Theory and Practice of Visual Languages and Description Logics

Volker Haarslev

Habilitationsschrift

Universität Hamburg
Fachbereich Informatik

September 2001

Contents

List of Figures

Foreword

I have written this monograph in a style reflecting American English. As a matter of taste I often form sentences in the plural form ('we' instead of 'I') in order to enhance the readability of this text. However, I am the only author of this text and I am solely responsible for any possibly remaining errors.

I am very grateful to Prof. Dr. Bernd Neumann, who always supported my research and gave me the freedom to pursue quite diverse research areas such as visual languages and description logics. His concern for practical applications inspired me to drive my theoretical work by the experience gained from the development and use of actual applications.

This work is based on a close collaboration with several colleagues and former students. I am particularly indebted to Dr. Ralf Möller, Michael Wessel, Carsten Lutz, and Anni-Yasmin Turhan. My colleague Dr. Ralf Möller was heavily involved in the research on description logics and gave essential advice on the work about visual languages. Michael Wessel implemented the GenEd editor and co-designed the VISCO query language and implemented the VISCO system. Carsten Lutz participated in the work on $\mathcal{ALCRP}(\mathcal{D})$ and helped to improve the presentation of the $\mathcal{ALCNH}_{R^+}$ calculus. Anni-Yasmin Turhan implemented prototype systems for the description logics $\mathcal{ALC}(\mathcal{D})$ and $\mathcal{ALCRP}(\mathcal{D})$ and co-developed optimization techniques for description logics with concrete domains. She also helped to test the RACE system and co-authored the RACE user manual.

I am also very thankful to Prof. Dr. Franz Baader and the members of his group at the RWTH Aachen for their detailed reviews that helped to improve many description logic papers which are now part of this monograph.

I am indebted to Prof. Dr. Kim Marriott and Dr. Bernd Meyer, both from Monash University, Australia. They always supported my logical approach to visual language theory and gave me plenty of feedback during discussions at workshops and conferences that helped to improve the visual language theory papers which became part of this monograph.

I would like to thank Prof. Dr. Bernd Neumann and Michael Wessel for their valuable comments on a draft of this monograph.

Part I

Introduction

Chapter 1

Motivation and Overview

This monograph reports on research carried out by the author over the last six years. The research comprises two major areas, visual language theory and description logics, which seem to be quite diverse. However, it will become clear to the reader in the following that both lines of research are intertwined with each other. The work on visual language theory gave important impetus to the author's research on description logics and the results from description logics motivated new approaches for visual language theory. Another basic principle underlying the research reported in this monograph is the coexistence of theory and practice. It is the author's firm belief that the advancement of theoretical work should be driven by shortcomings encountered in practical applications and, vice versa, results from theoretical work should be evaluated and possibly refined with the help of actual applications.

The following sections in this chapter will briefly introduce the above-mentioned research areas and relate the author's research to the body of work carried out in these areas. The chapter closes with a section giving a road map for the remainder of this monograph.

1.1 Theory and Practice of Visual Languages

Visual languages, such as diagrams or maps, have been important components of human communication over centuries. Unfortunately, the theoretical aspects of visual languages have only been analyzed to some extent but are in no way comparable to what we know today about sequential languages. One of the problems in visual language theory already starts with the definition of a visual language. In the context of this monograph we will regard a visual language as a collection of diagrams representing valid sentences in this language. A diagram is considered as a constellation of graphical symbols or icons in a two dimensional space. The meaning of a diagram (and a sentence in this language) is mainly defined by the meaning of its elements and their spatial relationships. Obviously, the work on visual language theory is closely related to research in diagrammatic reasoning (e.g. see [Glasgow et al., 1995] for an overview and [Olivier et al., 2000] for a recent survey).

The research on visual language (VL) theory is mainly concerned with the specification of visual languages. The body of work can be divided into three general approaches (see [Marriott and Meyer, 1998b, page 2]). The grammatical approach has its historical roots in formal language theory and theoretical linguistics. The logical approach is influenced by formalisms developed in the artificial intelligence community. The algebraic approach is derived from work in the formal specification of data types, programming languages and programming environments.

For sake of brevity we do not attempt to give an overview on research in visual language theory. There already exists an excellent book on visual language theory with a recent survey of visual language specification and recognition [Marriott et al., 1998].

The research on visual language theory presented in the next part of this monograph plays a major role in the logical approach. This research is concerned with the formal specification of visual (or diagrammatic) notations such as (aspects of) petri nets, entity-relationship diagrams, or visual programming languages. Our approach employs as a formal method the description logic $\mathcal{ALCQ}$ with so-called ABoxes. The approach is feasible for both the recognition and the specification of these notations. We used terminological (TBox) knowledge for expressing syntax and static semantics and assertional (ABox) knowledge for describing actual diagram examples. Our theory was practically applied to the formal design of visual notations with the help of the generic editor GenEd which supports editing of diagrams. GenEd offers the user a large variety of diagram elements and editing operations. The actual class of diagrams accepted by GenEd is specified with the help of description logic.

A parallel research effort is reported in the last two chapters of part two. It is concerned with the design of a visual spatial query language in the context of geographical information systems (GIS). A redesign of this query language has been implemented and exemplified with an application scenario using city maps of Hamburg. The effort clearly demonstrates the advantage of visual against textual query languages.

The fourth part of this monograph revisits our work on visual language theory by taking into account the results from our research on description logics where we developed mechanisms for integrating reasoning about spatial domains. The description logic $\mathcal{ALCRP}(\mathcal{D})$ is used to provide a first step towards specifying a semantics for visual spatial query languages.

Several requirements for the theory and practice of description logics were derived from our work on visual languages. Our theoretical work was based on the logic $\mathcal{ALCQ}$ and formed the basis for the design and implementation of the generic editor GenEd. Due to the unavailability of appropriate reasoners for $\mathcal{ALCQ}$, GenEd used the description logic system Classic [Brachman et al., 1991; Borgida and Patel-Schneider, 1994] which offers TBox and ABox reasoning for a language similar to $\mathcal{ALEN}$. Our experience with GenEd and Classic proved a strong demand for description logics with higher expressiveness and more efficient reasoners. Even with this relatively simple description logic, GenEd (i.e. the Classic system) was not able to analyze diagrams consisting of $\sim$100 elements

within several hours of runtime. The other requirement was the need to integrate reasoning about spatial domains, e.g. qualitative spatial relations, into description logics. Without this integration our specifications are incomplete w.r.t. the semantics of underlying spatial domains.

1.2 Theory and Practice of Description Logics

Description logic (DL) theories are based on the ideas of structured inheritance networks [Brachman and Schmolze, 1985]. In a DL, a factual world consists of named individuals and their relationships that are asserted through binary relations (roles). Hierarchical descriptions about sets of individuals (concepts) form the terminological knowledge (TBox). The assertional language of a DL is designed for stating constraints (in an ABox) for concept or role membership that apply to a particular domain or world. Most description logics are based on the basic logic $\mathcal{ALC}$ [Schmidt-Schauss and Smolka, 1991] offering standard constructors $(\neg, \sqcap, \sqcup, \exists, \forall)$ for composing descriptions. There exists a correspondence between several DLs and modal logics [Schild, 1991]. We refer to [Brachman et al., 1991; Woods and Schmolze, 1992; Borgida, 1995] for a general introduction into description logics.

As mentioned above we identified the need to integrate spatial reasoning into suitable description logics. Due to our experience with the DL system CLASSIC we first developed an approach to integrate reasoning about qualitative spatial reasoning into a language such as $\mathcal{ALEN}$. However, the need for high expressiveness resulted in a more general approach, the development of the DL $\mathcal{ALCRP}(\mathcal{D})$ offering a so-called role-forming predicate operator. This operator is based on the notion of concrete domains. A concrete domain offers a decision procedure for the consistency of a conjunction of concrete predicates ranging over a suitable domain. The feasibility of our $\mathcal{ALCRP}(\mathcal{D})$ approach is demonstrated in the context of GIS using a concrete domain based on the RCC theory [Randell et al., 1992]. In a parallel effort, we developed a prototype implementation for $\mathcal{ALCRP}(\mathcal{D})$ [Turhan, 1998]. Recently, we adapted two of the most effective optimization techniques to $\mathcal{ALC}(\mathcal{D})$ [Baader and Hanschke, 1991], the basic DL for reasoning with concrete domains, and to its extension $\mathcal{ALCRP}(\mathcal{D})$ [Turhan, 2000; Turhan and Haarslev, 2000].

The experience with this DL prover and the development of new optimization techniques for DL reasoners [Horrocks, 1997] were the main impetus for our research on the design and implementation of the DL $\mathcal{ALCNH}_{R^+}$. We advanced the theory of DLs by developing a calculus deciding the ABox satisfiability problem for the logic $\mathcal{ALCNH}_{R^+}$. This DL is already quite close to the above mentioned requirements from VL theory and seems to be suitable for developing an appropriate highly optimized ABox DL reasoner. This leads to our work on the practice of descriptions logics.

In the context of $\mathcal{ALCNH}_{R^+}$ which extends the logic $\mathcal{ALC}$ with number restriction, role hierarchies and transitive roles, we developed the ABox DL reasoner RACE. This DL system supports TBox and ABox reasoning for $\mathcal{ALCNH}_{R^+}$ and serves as a testbed for empirically evaluating appropriate optimization techniques.

RACE employs a set of nowadays "standard" and several novel optimization techniques. The techniques integrated into RACE can be classified into three main categories. The first category is the most general one and concerns so-called SAT techniques, i.e. techniques optimizing the satisfiability test for concept terms. A preprocessing phase transforms concept terms into negation normal form, removes duplicates, performs obvious simplifications, detects obvious clashes, flattens nested and/or expressions, normalizes the order of disjuncts and conjuncts, and ensures that all concepts, which are structurally equal, are also pointer equal. The pointer equality facilitates clash testing. For each concept, its negated counterpart is precomputed.

The tableaux expansion strategy of RACE incorporates the trace technique (see [Schmidt-Schauss and Smolka, 1991]) in an adapted version which considers the availability of role hierarchies and number restrictions in $\mathcal{ALCNH}_{R^+}$. This tableaux expansion strategy performs a depth-first search and allows one to reduce the memory requirements during a proof. Only assertions derived during a depth-first traversal of the search space need to be kept in memory at one time.

Disjunctions in concept terms are a major source of complexity in tableaux expansion. Two optimization strategies, semantic branching and dependency-directed backtracking, address this problem in particular. Caching the satisfiability status of independent "subtableaux" increases the memory consumption but is also very effective for many problems. Due to the logic $\mathcal{ALCNH}_{R^+}$, which allows for terminological cycles, lazy unfolding for concept terms has to be used. The tableaux expansion may be blocked using only a subset test instead of an equality test or other more expensive techniques. Blocking the tableaux expansion interacts with caching and may require the retraction of cache entries, i.e. making caching non-monotonic in RACE. The so-called signature calculus optimizes the handling of number restrictions. Domain and range restrictions for roles are efficiently dealt with by a special form of lazy unfolding.

The second category of optimization techniques is concerned with TBox reasoning, i.e. preprocessing of generalized concept inclusions (GCIs) and classification of named concepts in TBoxes. Usually, GCIs have to be represented as disjunctions. Therefore, GCIs are simplified and absorbed into concept definitions whenever possible. Concept definitions can be efficiently dealt with by the lazy unfolding technique. GCIs expressing domain and range restrictions for roles are also absorbed and addressed as mentioned above. RACE uses marking techniques controlled via the computation of so-called told subsumers and told disjoints in order to avoid unnecessary subsumption tests. The classification requires both a top and bottom search phase since axioms may be specified in arbitrary order and GCIs may cause subsumption relationships. Named concepts are ordered via topological sorting for safely bypassing the expensive bottom search phase during classification. A clustering technique tries to reduce the number of subsumption tests performed in the top search phase. The flat and deep model merging technique creates and caches pseudo models of concept terms and is a good heuristics to reduce the overall number of subsumption tests.

The third category comprises techniques especially designed for optimizing the ABox satisfiability test and ABox realization for individuals. The contraction technique transforms

ABoxes in order to maximize caching techniques. ABox realization also uses marking techniques controlled via told subsumers and told disjoints that avoid unnecessary ABox satisfiability tests. The individual model merging technique creates and caches pseudo models for ABox individuals. In analogy to the model merging techniques it reduces the overall number of ABox satisfiability tests.

All these techniques are already implemented in RACE and have been empirically evaluated with a set of either standard or newly created benchmark problems. Several of these problems were derived from knowledge bases used in actual applications.

1.3 Road Map

This monograph is organized in six parts. The first part contains this introduction and the German summary. The remaining parts describe the research outlined above in detail. Parts consisting of several chapters are usually introduced by chapter summaries.

The second part contains contributions to the theory and practice of visual languages. It starts with a theory for describing visual notations and describes the generic editor GenEd which is based on this theory. It continues with the design of a visual spatial query language motivated by geographical information systems. The last chapter of this part describes the VISCO system based on a redesign of the query language.

The third part is concerned with the integration of qualitative spatial reasoning into description logic theory. It begins with a first proposal for extending a logic similar to $\mathcal{ALEN}$. It ends with a chapter introducing our research on $\mathcal{ALCRP}(\mathcal{D})$, a logic supporting reasoning with concrete domains. This concrete domain mechanism is used to demonstrate the integration of spatial reasoning using examples from the GIS domain.

The fourth part revisits the theory presented in the second part and gives a first proposal for a semantics of visual spatial query languages using $\mathcal{ALCRP}(\mathcal{D})$.

The fifth part reports on the research about practical reasoning with expressive description logics. First, it introduces a calculus for the logic $\mathcal{ALCNH}_{R^+}$ deciding the ABox consistency problem. This calculus forms the basis for the RACE architecture. It continues with a chapter presenting optimization techniques that are either novel or significantly extend already known techniques. The effectiveness of these techniques is demonstrated with an empirical evaluation. This part concludes with new optimization techniques that proved to be effective in the context of "simple" but huge knowledge bases containing over 100,000 axioms and 60,000 hierarchical roles.

The last part gives a general summary of this monograph and tries to assess the achievements of the research presented here, followed by an outlook on future and ongoing research. The appendix consists of two sections. The first section summarizes the calculus for $\mathcal{ALCRP}(\mathcal{D})$ which is used in the second section where an extended reasoning example using $\mathcal{ALCRP}(\mathcal{D})$ in a GIS domain is given in full detail.

Kapitel 2

Deutsche Zusammenfassung

Diese deutsche Zusammenfassung ist als Ergänzung zum englischsprachigen Teil dieser Monographie zu sehen. Daher werden viele der im englischsprachigen Teil exakt eingeführten Begriffe und Probleme im deutschsprachigen Teil bewußt informell beschrieben oder motiviert. Damit sollen unangenehme Redundanzen vermieden werden. Eine präzise Einführung dieser Begriffe im deutschsprachigen Teil erscheint nur dann sinnvoll, wenn große Teile dieser Arbeit ins Deutsche übersetzt werden würden.

Einleitung

Diese Monographie stellt Forschungsarbeiten des Autors dar, die in den letzten sechs Jahren durchgeführt wurden. Die Arbeiten beschäftigen sich mit zwei wesentlichen Bereichen, der Theorie und Praxis visueller Sprachen und der Theorie und Praxis von Beschreibungslogiken. Beide Forschungsgebiete erscheinen auf den ersten Blick wenig Gemeinsamkeiten zu haben, es wird allerdings nachfolgend klar werden, daß die Arbeiten dazu miteinander verwoben sind und sich die Erkenntnisse in diesen Gebieten wechselseitig ergänzt haben. Die Arbeiten zu visuellen Sprachen haben die Forschung im Bereich der Beschreibungslogiken befruchtet, und umgekehrt haben die Erkenntnisse über Beschreibungslogiken neue Wege zur Weiterentwicklung der Theorie visueller Sprachen ermöglicht.

Ein wichtiges Prinzip bei der Durchführung dieser Arbeiten ist die Koexistenz von Theorie und Praxis. Arbeiten im Bereich der Theorie sollten immer durch bekannte Defizite in praktischen Anwendungen motiviert werden. In analoger Weise sollten Erkenntnisse aus der Theorie mithilfe praktischer Anwendungen evaluiert und möglicherweise verfeinert werden.

Die nachfolgenden Abschnitte geben über die vier wesentlichen Teile dieser Monographie einen Überblick. Für eine vertiefte Darstellung sei auf den englischsprachigen Teil dieser Monographie verwiesen.

Theorie und Praxis visueller Sprachen

Der erste Teil dieser Monographie präsentiert Arbeiten zur Theorie und Praxis visueller Sprachen. Visuelle Sprachen, z.B. Diagramme oder Karten, sind seit Jahrhunderten wichtige Bestandteile menschlicher Kommunikation. Allerdings wurden die theoretischen Aspekte visueller Sprachen erst in den letzten Jahren untersucht, so daß der derzeitige Erkenntnisstand nicht mit Arbeiten in anderen Bereichen, beispielsweise natürlichen Sprachen, vergleichbar ist. Das Problem mit visuellen Sprachen beginnt bereits mit der einheitlichen Charakterisierung oder Definition einer visuellen Sprache. In Kontext dieser Arbeit wird eine visuelle Sprache als eine Sammlung von Diagrammen verstanden, die gültige Sätze dieser Sprache repräsentieren. Ein Diagramm wiederherum wird als eine zweidimensionale räumliche Anordnung von graphischen Symbolen oder Piktogrammen angesehen. Die Bedeutung eines Diagramms und somit auch die Bedeutung eines Satzes der visuellen Sprache ist hauptsächlich durch die Bedeutung der Diagrammelemente und ihrer räumlichen Anordnung zueinander bestimmt. Die Arbeiten zur Theorie visueller Sprachen stehen in enger Beziehung zu den Arbeiten im Bereich des Schließens über Diagramme (siehe beispielsweise [Glasgow et al., 1995] für einen guten Überblick und [Olivier et al., 2000] für eine aktuelle Darstellung dieses Gebietes).

Die einschlägigen Forschungsarbeiten zu visuellen Sprachen beschäftigen sich hauptsächlich mit der Spezifikation derartiger Sprachen. Die Arbeiten lassen sich in drei grundsätzliche Ansätze unterteilen (siehe die Darstellung in [Marriott and Meyer, 1998b, Seite 2ff]). Ein Bereich beschäftigt sich mit der Spezifikation durch Grammatiken. Dieser Ansatz hat seine historischen Wurzeln in der Theorie der formalen Sprachen und der Linguistik. Der zweite Bereich untersucht die Verwendbarkeit von logischen Formalismen, die oft dem Gebiet der Künstlichen Intelligenz zugeordnet werden können. Im letzten Ansatz werden algebraische Formalismen favorisiert, die aus den Gebieten der abstrakten Datentypen und der formalen Spezifikation von Programmiersprachen und Programmierumgebungen stammen.

Es wird in dieser Monographie bewusst darauf verzichtet, einen umfassenden Überblick über den Bereich der visuellen Sprachen zu geben, da bereits eine exzellente und äußerst umfassende Darstellung zur Spezifikation und zum Parsen visueller Sprachen in [Marriott et al., 1998] zu finden ist.

Eine Theorie zur Beschreibung visueller Notationen

Kapitel 3 stellt einen logischen Ansatz zur Spezifikation visueller Sprachen oder Notationen vor. Dabei ist von Bedeutung, daß die Semantik eingesetzter Sprachmittel in der Theorie visueller Sprachen klar definiert ist. Dies bedeutet, daß die Bedeutung von Sprachelementen eindeutig durch ihre graphischen Repräsentanten gegeben ist, deren Semantik wohldefiniert ist, damit entsprechende Verfahren entwickelt werden können, die in Übereinstimmung mit der Semantik der Repräsentanten arbeiten, ohne daß ad-hoc Provisorien eingeführt werden müssen. Deshalb basiert die hier vorgestellte Theorie auf folgenden Komponenten. Objekte und ihre Beziehungen untereinander sind mithilfe der Topologie von offenen und

geschlossenen Punktemengen definiert. Es werden drei repräsentative Ansätze zur Definition qualitativer räumlicher Relationen vorgestellt [Egenhofer, 1991; Clementini et al., 1993; Randell et al., 1992], die letztendlich eine ähnliche Definition von Objekten und ihren Beziehungen gestatten.

Beschreibungslogik wird eingesetzt, um topologische Beziehungen und qualitative räumliche Relationen mit konzeptuellen Beschreibungen zu kombinieren. Beschreibungslogiken basieren auf Ideen zu strukturierten Vererbungsnetzen [Brachman and Schmolze, 1985]. In einer Beschreibungslogik wird eine faktische Welt als eine Menge von benannten Individuen verstanden, deren Beziehungen untereinander üblicherweise durch binäre Relationen beschrieben werden. Hierarchische Beschreibungen (oder auch Axiome) über Mengen von Individuen bilden sog. konzeptuelle oder terminologische Beschreibungen. Solche Beschreibungen (oder auch Terme) über Individuen werden als *Konzepte* bezeichnet, während die binären Relationen *Rollen* genannt werden. Die Elemente der rechten Seite einer Rolle werden als Füller dieser Rolle bzgl. eines Individuums bezeichnet. *Attribute* sind Rollen, die höchstens einen Füller haben dürfen.

Beschreibungen können aus Namen bestehen, die Konzepte, Rollen (Attribute) und Individuen denotieren, sowie aus Operatoren zur Kombination von Beschreibungen. Die modellorientierte Semantik der Sprachelemente kann auf Prädikatenlogik erster Ordnung oder auf einer kompositionalen Axiomatisierung durch Mengentheorie basieren. Die meisten Beschreibungslogiken basieren auf der „Basislogik“ $\mathcal{ALC}$ [Schmidt-Schauss and Smolka, 1991], die Standardoperatoren $(\neg, \sqcap, \sqcup, \exists, \forall)$ zur Konstruktion von Beschreibungen anbietet. Es existiert eine wohlbekannte Korrespondenz zwischen verschiedenen Beschreibungslogiken und Modallogiken [Schild, 1991]. Für eine weiterführende Einführung in das Gebiet der Beschreibungslogik wird auf [Brachman et al., 1991; Woods and Schmolze, 1992; Borgida, 1995] verwiesen.

Bei Beschreibungslogiken wird in der Regel zwischen zwei formalen Beschreibungsmitteln unterschieden. Eine Menge von Axiomen wird in einer TBox zusammengefasst, während eine ABox assertorische Beschreibungen zwischen benannten Individuen enthält. Eine grundlegende Inferenz, die mit TBoxen assoziiert wird, ist die Ableitung von Subsumptionsbeziehungen zwischen gegebenen Namen für Konzeptbeschreibungen. Dieser Inferenzdienst wird auch Klassifikation genannt. Die Subsumptionsbeziehung kann wiederherum auf den Konsistenzbegriff eines Konzeptterms zurückgeführt werden. Bei ABoxen ist deren Konsistenz eine grundlegende Inferenz. Die ABoxkonsistenz kann benutzt werden, die Menge der speziellsten Konzeptnamen (bzgl. der Subsumptionsbeziehung in einer TBox) eines Individuums zu bestimmen, von denen dieses Individuum eine Instanz ist. Dieser Vorgang wird auch als Realisierung bezeichnet.

Die Beschreibungslogik wird für visuelle Sprachen folgendermaßen eingesetzt. Konzepte beschreiben Elemente der visuellen Sprache, wobei sog. primitive Konzepte grundlegende geometrische Elemente repräsentieren. Mithilfe des Klassifikation einer TBox können die Konzeptnamen in einer Taxonomie angeordnet werden. Diese Taxonomie wird benutzt, um die Realisierung von Individuen in einer ABox durchzuführen. Individuen bezeichnen Elemente aus Beispielsätzen einer visuellen Sprache. Somit kann die Realisierung als Par-

sen eines Satzes verstanden werden. Daher kann der Einsatz von Beschreibungslogiken als Grundlage dafür gedeutet werden, daß Spezifikationen visueller Sprachen „ausführbar“ und mithilfe von Beispielen (gegeben durch ABoxen) verifizierbar werden. Inkonsistente Beschreibungen werden automatisch erkannt. Diese Eigenschaften führen direkt zur Anwendung der Theorie zur Beschreibung visueller Sprachen.

Der generische, objektorientierte Editor GenEd bietet eine Vielzahl von graphischen Elementen, die in einer visuellen Sprache enthalten sein können. Die geometrische Schlußfolgerungskomponente basiert auf der oben skizzierten Theorie und kann entsprechende räumliche Beziehungen zwischen Diagrammelementen berechnen. Ein Beschreibungslogiksystem, welches die Klassifikation von TBoxen und die Realisierung von ABoxen unterstützt, wird zum Parsen der erstellten Diagramme verwendet.

Die Plausibilität der Theorie zur Beschreibung visueller Sprachen wird anhand dreier Beispiele untermauert, nämlich einigen Aspekten von Petrinetzen, Entity-Relationship-Diagrammen und der visuellen Programmiersprache Pictorial Janus [Kahn and Saraswat, 1990; Kahn et al., 1991].

Eine visuelle räumliche Anfragesprache und ihre Realisierung

Kapitel 4 und 5 beschreiben die Arbeiten zur Entwicklung einer visuellen räumlichen Anfragesprache, die im Rahmen des VISCO-Systems implementiert wurde. Diese Arbeiten basieren auf den im Kapitel 3 entwickelten Ideen, die im Kapitel 8 zur Beschreibung der Semantik visueller räumlicher Anfragesprachen erweitert werden.

Die von VISCO unterstützte visuelle räumliche Anfragesprache kann sowohl topologische als auch metrische Einschränkungen und Beziehungen zwischen Anfrageelementen ausdrücken. Die Anfragesprache wird mithilfe einer Metapher der „naiven Physik“ visualisiert. Die Eigenschaften von Sprachelementen werden als Gummibänder, Streichhölzer, Drehgelenke, Nägel, Murmeln, usw. interpretiert. Die physikalischen Eigenschaften der von den Visualisierungen repräsentierten Gegenstände sollen beim Benutzer das Verständnis der Semantik der Sprachelemente unterstützen. Beispielsweise können Gummibänder sich zusammenziehen oder gedehnt werden, oder im Gegensatz zu Streichhölzern um geometrische Elemente gewickelt werden. Weiterhin darf eine Murmel im Unterschied zu einem Nagel herumrollen und damit ihre räumliche Position verändern.

Im Gegensatz zu anderen relevanten Arbeiten (siehe bspw. [Egenhofer, 1996]), deren Fokus auf topologischen Beschreibungen liegt, verwendet VISCO einen Bottom-Up-Ansatz und berücksichtigt direkt die durch das Anfragediagramm gegebenen geometrischen Eigenschaften, aber ermöglicht auch das Annotieren durch Metainformation, die eine Spezifikation fast reiner topologischer Anfragen gestattet. Die Metainformation kann genutzt werden, um Relaxierungen, zusätzliche Einschränkungen oder Freiheitsgrade (don't care) zu spezifizieren.

Das Basiselement von VISCO's Anfragesprache ist die sog. Transparentfolie, die eine rechteckige Form hat und ein eigenes lokales kartesisches Koordinatensystem besitzt. Folien können skaliert, verschoben, rotiert und übereinander gelegt werden. Sie müssen immer

einen Fixpunkt bzgl. möglicher Transformationen besitzen. Weitere Sprachelemente können innerhalb von Folien erzeugt werden. Dies sind Fixpunkte, Murmeln, Nägel, Streichhölzer, Gummibänder und Umzäunungen. Fixpunkte, Murmeln und Nägel können auch Stützpunkte einer Polylinie repräsentieren, während Streichhölzer starre und Gummibänder flexible Segmente einer Polylinie darstellen. Damit ist es möglich, eine Variabilität in der möglichen Formveränderung von Polylinien zu beschreiben. Gleichzeitig wird mit diesen Sprachmitteln ein fließender Übergang zwischen geometrischen und topologischen Anfrageelementen ermöglicht. Ein weiteres wichtiges Element dafür ist eine Umzäunung, die aus einer sich nicht selbst schneidenden Polylinie besteht. Sie wird benutzt, um die Bewegungsfreiheit von Murmeln einzuschränken. Mithilfe dieser Elemente können komplexere Objekte aggregiert werden, wobei Gelenke sowohl als Murmeln wie auch als Nägel wählbar sind. Der integrierte Parser von VISCO erkennt qualitative räumliche Beziehungen zwischen Anfrageelementen. Eine Anfrage ist somit ein Diagramm, das eine Beispielkonstellation von gesuchten Objekten beschreibt, die unter Einhaltung der implizierten Einschränken zu diesem Beispiel passen sollen.

Drei repräsentative Anfrageszenarien im Kontext von Stadtkarten von Hamburg werden vorgestellt und erläutert. Der implementierte VISCO-Prototyp besteht aus einem graphischen Editor zur Erstellung von Diagrammen, die eine Anfrage repräsentieren. Der Prototyp enthält weiterhin eine Komponente zur Ausführung der Anfrage und der Inspektion der Ergebnisse. Diese Komponente basiert auf einem optimierenden Anfrageübersetzer.

Anforderungen an Beschreibungslogiken

Aus den bisher skizzierten Arbeiten zur Theorie und Praxis visueller Sprachen konnten in Bezug auf Beschreibungslogiken mehrere Anforderungen gewonnen werden. Aufgrund der Tatsache, daß entsprechende Beschreibungslogiksysteme für die Logik $\mathcal{ALCQ}$, die in Kapitel 3 verwendet wird, nicht zur Verfügung standen, verwendet der generische Editor GenEd das Beschreibungslogiksystem Classic. Die Erfahrungen im Umgang mit GenEd und Classic haben den Bedarf für ausdrucksstarke Beschreibungslogiken und entsprechende effiziente Systeme untermauert. Beispielsweise konnten ABoxen mit etwa 100 Elementen innerhalb von mehreren Stunden Rechenzeit von Classic nicht erfolgreich realisiert werden, obwohl Classic nur eine Logik ähnlich zu $\mathcal{ALEN}$ unterstützt. Weiterhin wurde deutlich, daß Beschreibungslogiken das Schließen über räumliche Domänen, insbesondere über qualitative räumliche Relationen, unterstützen müssen, um Spezifikationen zu vermeiden, die bzgl. räumlicher Domänen unvollständig sind.

Räumliches Schließen mit Beschreibungslogiken

In diesem Teil der Monographie werden Arbeiten zu Beschreibungslogiken vorgestellt, die u.a. aus der Erfahrung mit visuellen Sprachen inspiriert sind.

Qualitatives räumliches Schließen mit Beschreibungslogiken

Kapitel 6 bietet einen Lösungsvorschlag, gewisse Eigenschaften des Raums in die Semantik einer Beschreibungslogik zu integrieren, die ähnlich zu $\mathcal{ALEN}$ ist. Diese Integration basiert auf der Idee, eine räumliche Region als eine Menge von Punkten zu interpretieren und die Subsumptionsbeziehung zwischen Konzepten auf Subsumption zwischen Regionen zu erweitern. Eine Region subsumiert eine andere Region, wenn die erstere die zweite enthält. In Anlehnung an [Egenhofer, 1991] definieren wir acht qualitative Basisrelationen, die sich wechselseitig ausschließen, und fünf weitere Relationen, die als Disjunktion von Basisrelationen definiert sind. Diese 13 Relationen werden in einer Subsumptionshierarchie beschrieben.

Darauf aufbauend führen wir einen neuen Operator ein, der eine qualitative räumliche Beziehung zwischen zwei Individuen beschreiben kann. Die Semantik von $(\bigcirc\ \mathsf{sr}\,.\,\mathsf{C})$ kann folgendermaßen beschrieben werden. Dieser Term beschreibt die Menge aller Individuen, die mit anderen Individuen, die eine Instanz von dem Konzept C sind, über eine qualitative räumliche Relation sr in Beziehung stehen. Dabei gilt diese Beziehung nicht direkt für diese Individuen, sondern für Regionen, die mit diesen Individuen über ein Attribut assoziiert sind. Dieser Sachverhalt wird in Abbildung 6.4 visualisiert. Die Nützlichkeit dieses Operators wird anhand eines Landkartenbeispiels für Norddeutschland demonstriert, wo räumliche und terminologische Schlüsse kombiniert werden. Eine prototypische Implementation hat gezeigt, daß dieser neue Operator in das CLASSIC-System integriert werden konnte. Allerdings musste die Kombinierbarkeit mit anderen Operatoren syntaktisch eingeschränkt bleiben. Dieser Nachteil führte zur Entwicklung der Beschreibungslogik $\mathcal{ALCRP}(\mathcal{D})$, die im nächsten Abschnitt skizziert wird.

Beschreibungslogiken und konkrete Domänen

Kapitel 7 erweitert die in Kapitel 6 geschilderten Arbeiten zu einem wesentlich generelleren Ansatz. Es wird die neue Beschreibungslogik $\mathcal{ALCRP}(\mathcal{D})$ eingeführt, die als Grundlage zur Integration des Schließens über qualitative räumliche Relationen in Beschreibungslogiken anzusehen ist. $\mathcal{ALCRP}(\mathcal{D})$ basiert auf der Logik $\mathcal{ALC}(\mathcal{D})$ [Baader and Hanschke, 1991], die die Menge der Individuen in abstrakte und konkrete Objekte, z.B. reelle Zahlen, aufteilt. Abstrakte Individuen werden mit konkreten über Attribute in Beziehung gesetzt. Beziehungen zwischen konkreten Individuen können durch eine Menge sog. konkreter Prädikate ausgedrückt werden. Diese werden durch einen konzeptbildenden Prädikatoperator miteinander assoziiert. Ein Paar bestehend aus einer Menge konkreter Individuen und einer Menge konkreter Prädikate wird auch als konkrete Domäne bezeichnet. $\mathcal{ALCRP}(\mathcal{D})$ erweitert $\mathcal{ALC}(\mathcal{D})$ um einen sog. rollenbildenden Prädikatoperator, der die Einführung von definierten Rollen anhand konkreter Prädikate erlaubt. Nur sog. zulässige konkrete Domänen sind gestattet. Ein wichtiges Kriterium für die Zulässigkeit ist die Forderung, daß das Konsistenzproblem für endliche Konjunktionen von Prädikaten entscheidbar sein muß.

Leider ist das ABoxkonsistenzproblem für die Logik $\mathcal{ALCRP}(\mathcal{D})$ unentscheidbar. Allerdings läßt sich eine Syntaxrestriktion für Konzeptterme in TBoxen formulieren, sofern

diese Terme vollständig aufgefaltet werden, d.h. alle Namen werden durch ihre Definition ersetzt und das Negationszeichen darf nur noch vor nicht mehr auffaltbaren Namen vorkommen. Die Syntaxeinschränkung verhindert beispielsweise Terme der Art $\exists\, \mathsf{R}_1 \,.\, \forall\, \mathsf{R}_2 \,.\, \mathsf{C}$, falls R_1 und R_2 definierte Rollen sind. Mit dieser Syntaxeinschränkung kann gezeigt werden, daß das ABoxkonsistenzproblem für die eingeschränkte Logik $\mathcal{ALCRP}(\mathcal{D})$ entscheidbar ist. Im nachfolgenden verstehen wir unter $\mathcal{ALCRP}(\mathcal{D})$ immer die Variante mit den Syntaxeinschränkungen, ohne dies explizit zu erwähnen.

$\mathcal{ALCRP}(\mathcal{D})$ ist für räumlich-terminologisches Schließen sehr interessant, da sich beispielsweise bekannte Ansätze zum qualitativen räumlichen Schließen mithilfe einer entsprechenden konkreten Domäne integrieren lassen. Dies wird mit der konkreten Domäne $\mathcal{S}_2$ gezeigt, die auf dem RCC-Ansatz [Randell et al., 1992] basiert. Mithilfe der Domäne $\mathcal{S}_2$ wird das sog. Hamburg-Beispiel (siehe Abschnitt 7.4.1) erneut betrachtet. Es lassen sich nun räumlich-terminologische Schlüsse aufzeigen, die sich in $\mathcal{ALC}(\mathcal{D})$ oder mit dem in Kapitel 6 vorgestellten Ansatz nicht ableiten lassen. Diese Beispiele sind auch leicht auf das Schließen mit ABoxen übertragbar. Die Eigenschaften von $\mathcal{ALCRP}(\mathcal{D})$ machen diese Logik nun wieder für die Theorie visueller Sprachen interessant, da die Bedeutung von Sätzen einer visuellen Sprache oft von räumlichen Beziehungen abhängen. Ein erster Vorschlag zur Semantik von visuellen räumlichen Anfragesprachen wird im nächsten Abschnitt vorgestellt.

Eine Semantik visueller räumlicher Anfragesprachen

In Anlehnung an den Anwendungsbereich der geographischen Informationssysteme wird im Kapitel 8 ein erster Vorschlag zur Spezifikation der Semantik visueller räumlicher Anfragesprachen beschrieben. Diese Ideen werden anhand von Beispielszenarien für Geo-Informationssysteme illustriert. Als konkrete Domäne wird die Vereinigung zweier konkreter Domänen verwendet, nämlich $\mathcal{S}_2 \cup \mathcal{R}$. Dabei sei $\mathcal{R}$ definiert über der Menge $\mathbb{R}$ aller reeller Zahlen mit Prädikaten, die aus Ungleichungssystemen gebildet werden können, die wiederum aus ganzzahligen Polynomen mit mehreren Unbestimmten bestehen dürfen [Tarski, 1951]. Die Domäne $\mathcal{S}_2$ enthält alle nicht-leeren, regulär geschlossenen Teilmengen von $\mathbb{R}^2$ (siehe Kapitel 7.3 für eine genaue Definition).

Unter Annahme dieser Voraussetzungen werden die Logiken $\mathcal{ALC}$, $\mathcal{ALC}(\mathcal{D})$ und $\mathcal{ALCRP}(\mathcal{D})$ bzgl. ihrer Ausdruckskraft verglichen. Ein wohl überzeugendes Beispiel für räumlich-terminologische Schlüsse in $\mathcal{ALCRP}(\mathcal{D})$ ist die Beschreibung eines sog. Paradiesferienhauses ($\mathsf{paradise\_cottage}$), das zum Angeln geeignet sein muß ($\mathsf{fishing\_cottage}$), aber in einem „mückenfreien“ Wald ($\mathsf{mosquito\_free\_forest}$) liegen soll. Ein zum Angeln geeignetes Ferienhaus soll der Einfachheit halber einen Fluß „berühren“, während ein mückenfreier Wald räumlich disjunkt zu einem Fluß sein muß. Die folgenden Axiome beschreiben diese Begriffe mithilfe der Beschreibungslogik $\mathcal{ALCRP}(\mathcal{D})$. Zum intuitiven Verständnis wird hier die informelle Bedeutung in Form von Prädikatenlogik beschrieben, wobei Konzepte als einstellige und Rollen als zweistellige Prädikate gedeutet werden. Für eine präzise Definition der Sprachelemente von $\mathcal{ALCRP}(\mathcal{D})$ sei auf Kapitel 7.1 verwiesen.

$\mathsf{fishing\_cottage} \doteq \mathsf{cottage} \sqcap \exists\, \mathsf{is\_touching}\,.\,\mathsf{river}$

$(\mathit{fishing\_cottage}(x) = \mathit{cottage}(x) \wedge \exists\, y : \mathit{is\_touching}(x, y) \wedge \mathit{river}(y))$

$(\mathit{is\_touching}(x, y) = \exists\, z_1, z_2 : \mathit{has\_area}(x, z_1) \wedge \mathit{has\_area}(y, z_2) \wedge \mathit{touching}(z_1, z_2))$

$\mathsf{mosquito\_free\_forest} \doteq \mathsf{forest} \sqcap \forall\, \mathsf{is\_connected}\,.\,\neg\mathsf{river}$

$(\mathit{mosquito\_free\_forest}(x) = \mathit{forest}(x) \wedge \forall\, y : \mathit{is\_connected}(x, y) \Rightarrow \neg\, \mathit{river}(y))$

$(\mathit{is\_connected}(x, y) = \exists\, z_1, z_2 : \mathit{has\_area}(x, z_1) \wedge \mathit{has\_area}(y, z_2) \wedge \mathit{connected}(z_1, z_2))$

$\mathsf{paradise\_cottage} \doteq \mathsf{fishing\_cottage} \sqcap \exists\, \mathsf{is\_g\_inside}\,.\,\mathsf{forest} \sqcap \forall\, \mathsf{is\_g\_inside}\,.\,\mathsf{mosquito\_free\_forest}$

$(\mathit{paradise\_cottage}(x) = \mathit{fishing\_cottage}(x) \wedge \exists\, y_1 : (\mathit{is\_g\_inside}(x, y_1) \wedge \mathit{forest}(y_1)) \wedge$
$\forall\, y_2 : (\mathit{is\_g\_inside}(x, y_2) \Rightarrow \mathit{mosquito\_free\_forest}(y_2)))$

$(\mathit{is\_g\_inside}(x, y) = \exists\, z_1, z_2 : \mathit{has\_area}(x, z_1) \wedge \mathit{has\_area}(y, z_2) \wedge \mathit{g\_inside}(z_1, z_2))$

Aufgrund der durch die qualitativen räumlichen Relationen modellierten Beziehungen kann gezeigt werden, daß das Konzept paradise_cottage nicht erfüllbar ist. Dies liegt in der den Relationen zugrundeliegenden Semantik begründet:

> Eine Situation, wo eine Region r_1 (Ferienhaus) *innerhalb* (g_inside) einer anderen Region r_2 (Wald) liegt und die Region r_1 eine dritte Region r_3 (Fluß) *berührt* (touching), impliziert, daß r_2 mit r_3 *verbunden* (connected) ist, d.h. $\mathsf{g\_inside}(\mathsf{r}_1, \mathsf{r}_2) \wedge \mathsf{touching}(\mathsf{r}_1, \mathsf{r}_3) \Rightarrow \mathsf{connected}(\mathsf{r}_2, \mathsf{r}_3)$

Damit muß ein Paradiesferienhaus sowohl in einem Wald liegen, der mit einem Fluß räumlich verbunden ist, als auch in einem mückenfreien Wald, der nicht mit einem Fluß verbunden sein darf. Dies ist in $\mathcal{ALCRP}(\mathcal{D})$ ein Widerspruch.

Eine Semantik für Anfragen wird nun folgendermaßen definiert. Es wird angenommen, daß eine Anfrage als Diagramm formuliert ist, das eine Beispielkonstellation beschreibt. Die durch die Elemente des Diagramms repräsentierten semantischen Entitäten sind als Konzepte definiert. Das Anfragediagramm wird nun in eine ABox übersetzt (z.B. durch ein System wie VISCO). Dabei stellen Individuen Elemente des Diagramms dar. Die Datenbankeinträge seien ebenfalls als ABoxindividuen repräsentiert. Mit einem Standardabstraktionsverfahren können ABoxen bestimmter Form auf einen Konzeptterm reduziert werden. Die Beantwortung einer Anfrage besteht dann in der Berechnung aller Individuen, die Instanz dieses Konzeptterms sind. Die Anwendbarkeit des Abstraktionsverfahrens hängt von der Ausdruckskraft der zugrundeliegenden Beschreibungslogik und von der Struktur der betrachteten ABoxen ab. Die zulässigen ABoxen dürfen keine Gabeln (joins) oder Zyklen enthalten, falls diese Strukturen nicht auf Konzeptterme abbildbar sind. Weiterhin können keine beliebigen n-ären Anfragen behandelt werden. Dafür werden sog. ABoxmuster vorgeschlagen, die es erlauben, beliebige ABoxen um den Preis einer höheren Komplexität zu behandeln.

Eine prototypische Implementierung von $\mathcal{ALCRP}(\mathcal{D})$, die keine Optimierungstechniken einsetzt, wird in [Turhan, 1998] beschrieben. Experimente mit dieser Implementation haben gezeigt, daß der Einsatz von Optimierungstechniken unabdingbar ist, um das System in der Praxis realistisch einsetzen zu können. Basierend auf den in [Horrocks, 1997; Horrocks, 1998; Horrocks and Patel-Schneider, 1999] und im folgenden Kapitel vorgestellten Ergebnissen wurden erste Arbeiten zur Optimierung von Beschreibungslogiken mit konkreter Domäne durchgeführt [Turhan, 2000; Turhan and Haarslev, 2000].

Praktisches Schließen mit Beschreibungslogiken

Die Erfahrungen im Umgang mit dem Prototypsystem für $\mathcal{ALCRP}(\mathcal{D})$ machten bereits Defizite im Entwurf von Beschreibungslogiksystemen für Teilmengen von $\mathcal{ALCRP}(\mathcal{D})$, z.B. für die Basislogik $\mathcal{ALC}$, deutlich. Insbesondere der Umgang mit Disjunktionen während des Beweisverfahrens zur Prüfung der Konzepterfüllbarkeit und der ABoxkonsistenz war zu ineffizient, um praktische Probleme handhaben zu können. Weiterhin ergaben sich aus den Anforderungen zur Theorie visueller Sprachen, daß ein ABox-Inferenzsystem für Beschreibungslogiken mit Sprachkonstrukten wie Rollenhierarchien und Anzahlrestriktionen unterstützt werden sollte. Aus dieser Motivation heraus wurde ein ABoxkalkül für die Logik $\mathcal{ALCN}\mathcal{H}_{R^+}$ entwickelt, da die dafür anwendbaren Optimierungsverfahren teilweise auch auf $\mathcal{ALCRP}(\mathcal{D})$ übertragen werden können. Die Logik $\mathcal{ALCN}\mathcal{H}_{R^+}$ erweitert $\mathcal{ALC}$ um Anzahlrestriktionen, Rollenhierarchien und transitive Rollen. Kapitel 9 führt die Logik und ihr Kalkül ein und gibt einen Beweis für die Entscheidbarkeit des ABoxkonsistenzproblems. Aus Gründen der Entscheidbarkeit wird die Kombinierbarkeit von Anzahlrestriktionen und transitiven Rollen eingeschränkt. Anzahlrestriktionen sind nur für nicht-transitive Rollen gestattet, die weiterhin keine transitiven Unterrollen haben dürfen. Die Ausdruckskraft von $\mathcal{ALCN}\mathcal{H}_{R^+}$ ist bereits recht hoch, da das Vorhandensein von Rollenhierarchien und transitiven Rollen die Spezifikation von allgemeinen Konzeptinklusionen gestattet, d.h. es sind auch Axiome der Form $\mathsf{C} \sqsubseteq \mathsf{D}$ möglich, wobei C und D beliebige Konzeptterme sein dürfen. Eine Beispielmodellierung ist in Kapitel 9.3 zu finden.

Der Beweis zur Entscheidbarkeit des ABox-Konsistenzproblems basiert auf sog. Vervollständigungsregeln (und deren Eigenschaften), die gegebene Zusicherungen solange expandieren, bis entweder keine Regel mehr anwendbar ist und auch kein Widerspruch entdeckt wurde, d.h. die gegebene ABox ist konsistent, oder alle ableitbaren ABoxen einen Widerspruch enthalten, d.h. die gegebene ABox ist inkonsistent. Im Falle der Konsistenz einer ABox kann dann mithilfe einer kanonischen Interpretation ein Modell konstruiert werden. Man könnte daher den Vervollständigungsprozeß auch als den Versuch einer Modellkonstruktion deuten. Um die Vollständigkeit des Verfahrens zu garantieren, wird als Teil des Beweises eine Strategie zur Regelanwendung definiert, die eine Bearbeitung von Rollennachfolgern gemäß einer Breitensuche sicherstellt. Dies wird benötigt, um die Korrektheit der kanonischen Interpretation zu gewährleisten. Nach dem Beweis der lokalen Korrektheit der Regeln werden die möglichen Widerspruchsarten definiert. Mithilfe der kanonischen Interpretation wird der entscheidende Teil des Beweis, nämlich die Korrektheit der Verfah-

rens, gezeigt. Die Terminierung des Verfahrens kann über die Angabe einer Oberschranke für die Menge aller ableitbaren Zusicherungen begründet werden.

Sowohl allgemeine Konzeptinklusionen als auch transitive Rollen erfordern eine Behandlung von Zyklen während der Anwendung von Vervollständigungsregeln. Um die Terminierung der Anwendung der Vervollständigungsregeln zu gewährleisten, muß eine sog. *Blockierungsbedingung* eingeführt werden, die die Vervollständigung (oder auch Tableauxexpansion) genau dann blockiert, wenn keine „neue“ Information abgeleitet werden kann. Dies sei mit dem folgenden Konzeptterm illustriert, der auf Konsistenz geprüft werden soll (has_ancestor sei eine transitive Rolle): $\exists_{\geq 2}$ has_ancestor $\sqcap \forall$ has_ancestor . ($\exists_{\geq 2}$ has_ancestor). Dieser Konzeptterm beschreibt folgenden Sachverhalt „*Jemand hat mindestens zwei Vorfahren und für alle diese Vorfahren gilt, daß sie ebenfalls mindestens zwei Vorfahren haben müssen*“. Ohne eine Blockierungsbedingung würde die Anwendung entsprechender Vervollständigungsregeln nicht terminieren, da bei der Modellkonstruktion durch die Transitivität von has_ancestor jeder Füller dieser Rolle ebenfalls wieder Nachfolger für has_ancestor haben muß. Da die Logik $\mathcal{ALCNH}_{R^+}$ jedoch die Eigenschaft besitzt, daß für alle Konxeptterme und ABoxen endliche Modelle existieren, ist es leicht möglich, eine Terminierung und die Existenz entsprechender Modelle sicherzustellen.

$\mathcal{ALCNH}_{R^+}$ ist gleichzeitig die Grundlage für die Entwicklung des Beschreibungslogiksystems RACE, das TBox- und ABox-Schließen für $\mathcal{ALCNH}_{R^+}$ implementiert. RACE ist ein prototypisches System, das u.a. der Evaluierung bekannter und neuartiger Optimierungstechniken zum Schließen mit $\mathcal{ALCNH}_{R^+}$ dient. Diese Techniken und ihre Evaluierung werden in den Kapitel 10 und 11 ausführlich dargestellt.

Verfahren zum Konsistenztest

Die von RACE eingesetzten Optimierungstechniken lassen sich in drei grundsätzliche Kategorien unterteilen. Die erste Kategorie betrifft die „Basismaschine“ von RACE, den sog. SAT-Tester, der den Konsistenztest für Konzepte und ABoxen realisiert. Eine Vorverarbeitungsphase transformiert Konzepte in ihre Negationsnormalform, entfernt redundante Elemente, führt offensichtliche Vereinfachungen durch, behandelt leicht erkennbare Widersprüche, verflacht geschachtelte Konjunktionen und Disjunktionen, normalisiert die Reihenfolge von Disjunkten und Konjunkten, und stellt sicher, daß alle strukturell gleichen Konzepte durch dieselbe Datenstruktur repräsentiert werden. Dies vereinfacht die Realisierung des Tests auf Widersprüche. Weiterhin werden aus Gründen der Einfachheit und Effizienz zu allen Konzepten ihre Negation vorberechnet und gespeichert.

Die Strategie zur Anwendung der Vervollständigungsregeln arbeitet in Anlehnung an die sog. Trace-Technik (siehe [Schmidt-Schauss and Smolka, 1991]), die eine Reduzierung des Speicherbedarfs im Mittel ermöglicht. Die Trace-Technik mußte für die Logik $\mathcal{ALCNH}_{R^+}$ derart angepaßt werden, daß die Interaktionen mit Rollenhierarchien und Anzahlrestriktionen berücksichtigt werden. Dabei findet die Vervollständigung im Gegensatz zum theoretischen Verfahren, das für den Beweis der Entscheidbarkeit entwickelt wurde (s.o.), in Form einer Tiefensuche statt, damit nur abgeleiteten Zusicherungen entlang eines Pfades

(trace) gleichzeitig gespeichert werden müssen.

Disjunktionen in Konzepttermen sind eine wichtige Ursache für die Zeitkomplexität von entsprechenden Vervollständigungsverfahren. Zwei generelle Optimierungstechniken, semantisches Verzweigen und abhängigkeitsgesteuertes Backtracking, können eine Effizienzsteigerung im Mittel bewirken. Ihre Anwendung auf Beweissysteme für Beschreibungslogiken wird in [Horrocks, 1997] ausführlich dargestellt und empirisch bewertet. Ohne den Einsatz dieser beiden Verfahren ist ein praktikables Beschreibungslogiksystem heute nicht mehr denkbar. Beim semantischen Verzweigen wird der Suchraum „semantisch“ halbiert, da davon ausgegangen wird, daß ein Disjunkt C entweder *wahr* (d.h. C führt zu keinem Widerspruch) oder *falsch* (d.h. $\neg\mathsf{C}$ führt zu keinem Widerspruch) sein kann. Allerdings ist es möglich, daß in einem Teilbaum beide Alternativen, C oder $\neg\mathsf{C}$, zu einem Widerspruch führen, d.h. dieser Teilbaum ist nicht erfüllbar. Derartige Situationen können beim Einsatz blinder Suche dazu führen, daß einmal erkannte Widersprüche, die unabhängig von der lokalen Entscheidung sind, u.U. immer wieder in anderen Teilbäumen erneut entdeckt werden. Dies bedeutet in der Regel, daß die Unerfüllbarkeit eines Teilbaumes von Entscheidungen abhängt, die in übergeordneten Teilbäumen getroffen wurden. Ein derartiges *Widerspruchsflattern* (clash thrashing) kann beim Einsatz des abhängigkeitsgesteuerten Backtracking vermieden werden. Bei diesem Verfahren werden die Abhängigkeiten von Disjunktionen vermerkt, so daß im Falle eines Widerspruchs beim Backtracking alle diejenigen verbleibenden Alternativen übergangen werden, deren Auswahl das Ableiten desselben Widerspruchs nicht verhindern kann.

Eine weitere Technik, das Speichern des Ergebnisses eines sog. Subtableauxtests (subtableaux caching), benötigt zwar im schlimmsten Fall für $\mathcal{ALCNH}_{R^+}$ exponentiell viel Speicher, ist aber für viele Anwendungsprobleme nötig, um diese lösen zu können. Als Beispiel sei eine Anwendung im Telekommunikationsbereich genannt, wo es darum geht, Interaktionen bei der Konfiguration von Telefonanlagen zu entdecken (siehe [Areces et al., 1999]). Aus der Modellierung von möglichen Interaktionen ergibt sich beispielsweise eine TBox, die RACE in weniger als 10 Sekunden klassifizieren kann, wenn die Ergebnisse der Subtableauxtests gespeichert und wiederverwendet werden. Ohne den Einsatz dieser Optimierungstechnik kann diese TBox dagegen nicht in 10000 Sekunden klassifiziert werden. Im Rahmen von RACE werden zwei Cache-Arten zum Einsatz gebracht. Der erste Cache-Typ vergleicht den Suchschlüssel mit den Schlüsseln der Einträge auf *Gleichheit*. Der zweite Cache-Typ unterscheidet zwischen Einträgen, die erfüllbar und unerfüllbar sind. Für die erfüllbaren Einträge werden deren Schlüssel mit dem Suchschlüssel auf eine Obermengenbeziehung verglichen, d.h. ist bereits ein erfüllbarer Eintrag für eine Obermenge bekannt, so gilt die Erfüllbarkeit ebenfalls für eine Untermenge. In analoger Weise kommen bei unerfüllbaren Einträgen der Test auf die Untermengenbeziehung zum Einsatz, d.h. ist bereits ein unerfüllbarer Eintrag für eine Untermenge bekannt, so gilt die Unerfüllbarkeit ebenfalls für eine Obermenge. Durch den zweiten Cache-Typ kann in der Regel eine Effizienzsteigerung in der Laufzeit und eine Reduktion des Speicherbedarfs für den Cache erreicht werden.

Die Ausdruckskraft von $\mathcal{ALCNH}_{R^+}$ erzwingt die Behandlung von terminologischen Zyklen.

Beispielsweise könnte ein Mensch beschrieben werden als jemand, der mindestens einen Menschen als Vorfahren besitzt ($\mathsf{human} \sqsubseteq \exists\, \mathsf{has\_ancestor}\,.\,\mathsf{human}$). Zwei Standardtechniken werden zur Behandlung von Zyklen eingesetzt. Zum einen werden Konzeptnamen nur nach Bedarf aufgefaltet (lazy unfolding), zum anderen darf die Tableauxexpansion blockiert werden (s.o.), wenn sichergestellt ist, daß keine neue Information erzeugt werden kann. Diese Technik interagiert allerdings mit dem Speichern von Subtableauxergebnissen, d.h. möglicherweise müssen bereits gespeicherte Subtableauxergebnisse widerrufen werden. Dies wird durch eine in RACE eingebaute Dependenzverwaltung für Cache-Einträge ermöglicht. Durch das Zurücknehmen von Einträgen wird der Cache in RACE nicht-monoton.

Durch Axiome der Form $\exists_{\geq 1}\, \mathsf{has\_ancestor} \sqsubseteq \mathsf{human}$ kann der „Urbildbereich“ (domain) und durch $\top \sqsubseteq \forall\, \mathsf{has\_ancestor}\,.\,\mathsf{human}$ der „Bildbereich“ (range) von Rollen eingeschränkt werden. Die erste Form der Axiome erfordert die Behandlung von Disjunktionen der Art $\neg\exists_{\geq 1}\, \mathsf{has\_ancestor} \sqcup \mathsf{human}$, die unbedingt zu vermeiden sind, da derartige Axiome sehr zahlreich sein können. In RACE ist ein erweitertes Tableauxverfahren integriert, das diese Art von Axiomen durch eine spezielle Form des Auffaltens nach Bedarf behandeln kann.

Eine weitere wichtige Ursache für die Zeit- und Speicherkomplexität ist die Behandlung von Anzahlrestriktionen in Kombination mit Rollenhierarchien. Ein naiver Umgang mit diesen Restriktionen führt in Tableauxverfahren dazu, daß sehr viele Individuen erzeugt werden müssen (z.B. unter Umständen 50000 Individuen für das Konzept ($\exists_{\geq 50000}\, \mathsf{hat\_sitz}$)). In Kapitel 10.4 wird ein geändertes Tableauxverfahren vorgestellt, daß nur notwendige Stellvertreter erzeugt, die über Signaturen (signatures) beschrieben werden und die von der verwendeten Zahl unabhängig sind. Beispielsweise würde für das Konzept $\exists_{\geq 50000}\, \mathsf{hat\_sitz}$ genau eine Signatur statt 50000 Individuen als Rollenfüller von $\mathsf{hat\_sitz}$ generiert werden. Empirische Untersuchungen haben gezeigt, daß damit eine Effizienzsteigerung in der Laufzeit um mehrere Größenordnungen erreicht werden kann.

Verfahren zur Behandlung von TBoxen

Die zweite Kategorie von Optimierungstechniken dient der effizienten Behandlung von TBoxen, insbesondere der Subsumptionstest soll optimiert werden. Eine wichtige Technik ist die Transformation bzw. Absorption von allgemeinen Konzeptaxiomen, die für Beschreibungslogiken erstmals in [Horrocks, 1997] eingeführt wurde. Dabei wird versucht, die Anzahl solcher Axiome zu verringern und sie in eine möglicherweise einfachere Form zu transformieren. Beispielsweise können die beiden Axiome $\{\mathsf{A} \sqsubseteq \mathsf{E} \sqcup \mathsf{F}, \mathsf{A} \sqcap \mathsf{B} \sqsubseteq \mathsf{C} \sqcap \mathsf{D}\}$ in folgendes Axiom $\mathsf{A} \sqsubseteq (\mathsf{E} \sqcup \mathsf{F}) \sqcap (\neg\mathsf{B} \sqcup (\mathsf{C} \sqcap \mathsf{D}))$ transformiert werden, das durch die Technik des Auffaltens nach Bedarf sehr effizient behandelt werden kann. Nichtabsorbierbare allgemeine Axiome müssen als Disjunktionen repräsentiert werden, die den Suchraum während eines Beweises vergrößern. Kapitel 10.3 stellt ein in RACE integriertes Verfahren dar, das die in [Horrocks, 1997] vorgeschlagene Technik erweitert und im Mittel verbessert. Weiterhin können die o.a. Axiome zur Definition des Urbild- und Bildbereichs von Rollen eliminiert und durch die Behandlung des Auffaltens nach Bedarf ersetzt werden.

In Kapitel 11 werden die in [Baader et al., 1994a] eingeführten Markierungs- und Pro-

pagierungstechniken erweitert. Diese dienen dazu, die Anzahl der Subsumptionstests zu reduzieren, indem die während der Vorverarbeitung von Konzepten und deren Klassifikation gewonnene Information ausgenutzt wird. Dabei werden strukturell leicht erkennbare Subsumptions- und Disjunktheitsbeziehungen zwischen Konzepten abgeleitet. Aufgrund der Ausdrucksstärke von $\mathcal{ALCNH}_{R^+}$ muß sowohl eine „Top-Down-Suche“ als auch eine „Bottom-Up-Suche“ durchgeführt werden, um Konzeptnamen in die Taxonomie einzusortieren. Kapitel 11 stellt ein neues Verfahren vor, das mithilfe einer topologischen Sortierung die benannten Konzepte in einer TBox in eine derartige Reihenfolge bringt, daß für primitive Konzepte die aufwendige Bottom-Up-Suche u.U. weggelassen werden kann. Diese Technik wird weiterhin durch eine Gruppierungstechnik unterstützt, die bei Konzeptnamen mit vielen direkten Nachfolgern (in der Taxonomie) diese Nachfolger in Gruppen einteilt, um ebenfalls die Anzahl von Subsumptionstests zu reduzieren. Statt die Subsumptionstests mit allen Gruppenmitgliedern einzeln zu überprüfen, wird nur ein korrekter aber unvollständiger Vortest auf Nicht-Subsumption (s.u.) durchgeführt. Ist dieser erfüllt, brauchen die Gruppenmitglieder nicht einzeln überprüft werden. Eine empirische Untersuchung für sehr große Wissensbasen belegt, daß sich durch die Kombination beider Techniken eine Rechenzeitersparnis von bis zu einer halben Größenordnung erreichen läßt.

Mithilfe sog. Pseudomodelle, die während eines Subtableauxtests erzeugt werden, kann die Anzahl der notwendigen Subsumptionstests ebenfalls erheblich reduziert werden. Dabei ist ein Pseudomodell für ein Konzept als eine Datenstruktur anzusehen, die Informationen enthält, die aus einem vollständig expandierten Tableau eines Konzeptkonsistenztests extrahiert wurde. Aus der Erkenntnis heraus, daß nur ein kleiner Prozentsatz (üblicherweise weniger als 5%) aller möglichen Subsumptionstests während der Klassifikation einer TBox tatsächlich eine Subsumptionsbeziehung ergibt, wird ein unvollständiger aber korrekter Vortest eingeführt. Statt sofort mit dem Tableauxverfahren zu prüfen, ob ein Konzept C ein Konzept D subsumiert, d.h. ob $\neg\,\text{satisfiable}(\{\mathsf{a}:\neg\mathsf{C}\sqcap\mathsf{D}\})$ gilt, wird getestet, ob die Pseudomodelle von $\neg\mathsf{C}$ und D verschmelzbar sind. Ist dies der Fall, so kann daraus abgeleitet werden, daß das Konzept C das Konzept D nicht subsumiert. Diese Technik wurde in [Horrocks, 1997] vorgestellt und analysiert. Im Kapitel 10.6 wird dieses Verfahren erstmals für die Logik $\mathcal{ALCNH}_{R^+}$ angepaßt und um eine Variante erweitert, die mit sog. tiefen Pseudomodellen arbeitet, d.h. die Pseudomodelle werden rekursiv traversiert und auf Verschmelzbarkeit überprüft. Die Korrektheit dieses Verfahrens wird formal untersucht. Empirische Untersuchungen belegen, daß mit der tiefen Variante der Pseudomodellverschmelzung eine Verringerung der Laufzeit während der Klassifikation um den Faktor 1.5-2 erreicht werden kann.

Verfahren zur Behandlung von ABoxen

Die dritte Kategorie umfaßt Techniken, die speziell zur Beschleunigung des ABox-Konsistenztests und der Realisierung von ABoxen entwickelt wurden. Die Kontraktionstechnik transformiert eine ABox derart, daß die Anzahl der vorhandenen Individuen verringert werden kann. Sie basiert auf der Idee, eine Menge von Individuen, die durch eine Rollenkette miteinander verknüpft sind, in einen geschachtelten Konzeptterm zu transformieren.

Dadurch können verstärkt Optimierungstechniken für Konzepte (z.B. Modellverschmelzung, Caching, etc.) eingesetzt werden. Empirische Untersuchungen für die Kontraktion (siehe Kapitel 10.5) belegen eine Steigerung in der Laufzeiteffizienz um eine bis mehrere Größenordnungen.

In Analogie zur Behandlung von TBoxen kommen bei der Realisierung von ABoxen auch entsprechende Markierungs- und Propagierungstechniken in Betracht. Aus den bekannten Zusicherungen für Individuen werden strukturell leicht erkennbare Instanz- und Disjunktheitsbeziehungen zu benannten Konzepten ermittelt. Dadurch können unnötige Instanztests vermieden werden. In Kapitel 10.7 wird erstmals die Technik der Pseudomodellverschmelzung für den Instanztest von Individuen entwickelt. Statt mit dem Tableauxverfahren zu überprüfen, ob ein Individuum a eine Instanz eines Konzepts C für eine gegebene ABox $\mathcal{A}$ ist, d.h. ob $\neg\,\text{satisfiable}(\mathcal{A} \cup \{\mathsf{a}:\neg\mathsf{C}\})$ gilt, wird getestet, ob die Pseudomodelle von a und $\neg\mathsf{C}$ verschmelzbar sind. Das Pseudomodell für ein Individuum a wird aus dem Konsistenztest der ABox $\mathcal{A}$ gewonnen, indem aus dem voll expandierten Tableaux alle Zusicherungen für das Individuum a weiterverarbeitet werden. Das Pseudomodell muß weiterhin noch derart erweitert werden, daß Rollennachfolger von a berücksichtigt werden. Auch dieser Vortest ist korrekt aber nicht vollständig, d.h. im Falle einer Verschmelzbarkeit der Pseudomodelle ist bekannt, daß das Individuum a keine Instanz des Konzepts C ist. Die Eleganz dieses Ansatzes liegt insbesondere darin begründet, daß die Pseudomodellverschmelzung für Individuen somit auf Algorithmen zur Pseudomodellverschmelzung für Konzepte zurückgeführt werden kann.

Die Effektivität der oben skizzierten Neuentwicklungen oder der Erweiterungen bekannter Verfahren wird anhand von TBoxen und ABoxen evaluiert, die entweder synthetisch generiert oder aus Anwendungen entstanden sind. Dabei gilt als Grundsatz für die hier dargestellten Arbeiten zur Optimierung, daß das Hauptaugenmerk in der Entwicklung von Verfahren zur Behandlung von TBoxen und ABoxen liegt, die idealerweise aus praktischen Anwendungen entstanden sind.

Zusammenfassung

Diese Arbeit hat sich mit zwei bisher weitgehend disjunkt voneinander betrachteten Forschungsgebieten befaßt, den visuellen Sprachen und den Beschreibungslogiken. Im Rahmen dieser Betrachtung konnte gezeigt werden, daß die durchgeführte Forschung beide Gebiete wechselseitig befruchtet hat. Der Einsatz von Beschreibungslogiken für visuelle Sprachen hat zu einem neuen logikbasierten Ansatz zur Theorie visueller Sprachen geführt. Die praktische Verwendung von Beweissystemen für Beschreibungslogiken hat die Notwendigkeit zur Entwicklung ausdrucksstärkerer Logiken und hochgradig optimierter Beweisverfahren bestätigt. Aus diesen Erkenntnissen heraus sind die Arbeiten zur Beschreibungslogik $\mathcal{ALCRP}(\mathcal{D})$ entstanden sowie die Entwicklung der Beschreibungslogik $\mathcal{ALCNH}_{R^+}$ initiiert worden. Der Entwurf und die Realisierung entsprechender optimierter Beweisverfahren im Rahmen des Systems RACE liegt darin begründet. Mit RACE wurde ein ABox-Beweissystem vorgestellt, das für fast alle Aspekte der Logik $\mathcal{ALCNH}_{R^+}$ angepaßte Optimierungs-

verfahren anbietet. Es ist zu hoffen, daß die in dieser Arbeit gewonnen und geschilderten Erkenntnisse weitere Fortschritte in verwandten Bereichen ermöglichen.

Part II

Contributions to Visual Language Theory

The first part of this monograph presents research about visual language theory. Chapter 3 addresses issues in visual language theory with the help of logic formalisms that were developed for reasoning tasks by the artificial intelligence and spatial databases communities, especially for spatial and diagrammatical reasoning. It describes an approach based on three formal components. *Topology* is used to define basic geometric objects. Theory about *spatial relations* from the domain of spatial databases is employed to define possible relationships between visual language elements. *Description logic theory* from the AI community is used to combine topology and spatial relations. The feasibility of this theory is demonstrated by describing three representative visual notations: entity-relationship diagrams, petri nets, and a pictorial language for concurrent logic programming. This chapter is based on [Haarslev et al., 1994; Haarslev, 1995; Haarslev, 1996a; Haarslev, 1996b; Haarslev and Wessel, 1996; Haarslev, 1998a].

Chapter 4 presents the design of the visual query system VISCO which offers a sketch-based query language for defining approximate spatial constellations of objects. VISCO smoothly integrates geometrical and topological querying with deductive spatial reasoning. It is based on a strong physical metaphor visualizing semantics of query elements. Approximate queries rely on combined topological and geometrical constraints enhanced with relaxations and "don't cares." Chapter 5 reports on the implementation of VISCO's spatial query language using city maps of Hamburg as example domain. Its innovative user interface consists of three interconnected components: a graphical (syntax-directed) query editor and visual language compiler, a browser for inspecting the query results, and a map viewer for browsing the spatial database. Chapter 5 also briefly reports on the process of compiling, optimizing, and executing VISCO's queries. Both chapters are based on [Haarslev and Wessel, 1997; Wessel and Haarslev, 1998]. The raster and vector maps used in both chapters were donated by the 'Amt für Geoinformation und Vermessung, Hamburg'.

Chapter 3

A Theory for Describing Visual Notations

This chapter reports on an approach to formalizing visual notations. We propose a *spatial logic* for describing syntax and static semantics of visual notations. This logic combines three components (topology, spatial/topological relations, description logic) that are themselves also formally specified with precise semantics. These components were derived from research communities that are related to visual language (VL) research: reasoning on diagrammatic representations and spatial databases. The goal of this chapter is to intensify the dialogue between these research communities and to "advertise" the benefits of this particular view of VL theory.

3.1 Introduction

The successful application of our theory to a completely visual language for concurrent logic programming, Pictorial Janus [Kahn and Saraswat, 1990; Kahn et al., 1991], has been reported in [Haarslev, 1995]. A revised and simplified version of the language specification of Pictorial Janus is also presented in this chapter. The experience with Pictorial Janus resulted in the development of the editor GenEd [Haarslev and Wessel, 1996] for designing visual notations. GenEd's generic semantics is based on and controlled by the theory described in this chapter.

GenEd is an object-oriented editor supporting the formal design and analysis of visual notations. Prominent features of GenEd are (1) it is *generic*, i.e. domain-specific syntax and semantics of drawings are specified by users; (2) it has a *built-in parser* for actual drawings, driven by our spatial logic; (3) it offers powerful *reasoning* capabilities about diagrams and their specification.

In principle, particular instances can be chosen for the components concerned with spatial/topological relations and description logic. This process depends on the nature of specific visual notations to be formally specified. For instance, the definition of Pictorial Janus

is mostly based on topological relations between lines, arrows, and regions. Therefore, we have selected corresponding definitions for primitive geometric objects, an appropriate theory on spatial (topological) relations [Clementini et al., 1993] that can deal with true 1D objects and regions, and a matching description logic. However, we like to emphasize that other visual languages or notations might require different definitions for objects and their possible relationships. The syntax and static semantics of particular classes of diagrams may be specified with description logic. This allows GenEd to support a large variety of diagrams.

This chapter is organized as follows. The next section discusses the theoretical foundation of our approach. Afterwards we describe the editor GenEd that implements our theory. This is followed by two sections presenting three representative visual notations and their formal specification. We conclude this chapter with a discussion of related work.

3.2 Theoretical Foundation

We believe that the semantics of representational devices used for VL theory should be well understood. That is, the meanings of represented language concepts should be unambiguously determined by explicit notational devices whose meanings (semantics) are understood, so that algorithms can operate on the representation in accordance with the semantics of the notation, without needing ad hoc provisions for specific VL domains. In the following we outline a fully formalized theory for describing visual notations that consists of several components. The definition of objects and relations is based on point-sets and topology. Description logic theory can be based on model-theoretic semantics appealing to first-order logic or on a compositional axiomatization with set theory. The next sections describe the components of the theory in more detail and briefly review alternative instances for these components.

3.2.1 Objects and Topology

The definition of basic geometric objects (the elementary vocabulary of a visual notation) usually relies on topology which is itself a basis for defining relationships between objects. In the following we assume the usual concepts of point-set topology with open and closed sets [Spanier, 1966]. The *interior* of a set λ_i (denoted by λ_i°) is the union of all open sets in λ_i. The *closure* of λ_i (denoted by $\overline{\lambda_i}$) is the intersection of all closed sets containing λ_i. The *complement* of λ_i (denoted by λ_i^{-1}) with respect to the embedding space $\mathbb{R}^n$ is the set of all points of $\mathbb{R}^n$ not contained in λ_i. The *boundary* of λ_i (denoted by $\partial\lambda_i$) is the intersection of the closure of λ_i and the closure of the complement of λ_i. It follows from these definitions that $\partial\lambda_i$, λ_i°, and $(\lambda_i^{-1})^{\circ}$ are mutually exclusive and $\partial\lambda_i \cup \lambda_i^{\circ} \cup (\lambda_i^{-1})^{\circ}$ is $\mathbb{R}^n$.

These definitions form the basis for Egenhofer's approach [Egenhofer, 1991] where the so-called *9-intersection* defines topological relations between objects. This method characterizes relations between two objects by nine set intersections (every pairwise combination of interior, boundary, and complement). The following restrictions apply to every pair of

sets. (1) Let λ_i, λ_j be n-dimensional sets with $\lambda_i, \lambda_j \subset \mathbb{R}^n$, (2) $\lambda_i, \lambda_j \neq \emptyset$, (3) all boundaries, interiors, and complements are connected, and (4) $\lambda_i = \overline{\lambda_i^o}$ and $\lambda_j = \overline{\lambda_j^o}$. A major drawback of this approach is the failure to describe true one-dimensional objects.

This was the motivation for the proposal by Clementini et al. [Clementini et al., 1993]. They extended Egenhofer's approach by introducing points and lines as additional object types and the dimension of intersections as new feature for discriminating more cases. Three types of geometric objects are modeled. *Regions* have to be connected and without holes. *Lines* and *arrows* must not be self-intersecting, are either circular or directed, and have exactly two end points. *Points* are elements of lines and describe their start or end points. The boundary of a point is an empty point-set, the boundary of a line is either an empty point-set (for a circular line) or a point-set consisting of its two end points (for a non-circular line). The boundary of a region is a circular line. The interior of an object is the object without its boundary. In case of points and circular lines their interior is identical to the object itself. Neither approach can deal with concave objects.

A third but different approach is based on the work of Clarke about "individuals and points" [Clarke, 1981; Clarke, 1985]. Clarke's calculus interprets individual variables as ranging over spatiotemporal regions and the two-place primitive predicate, "*x is connected with y,*" as a rendering of "*x and y share a common point*." Randell et al. [Randell et al., 1992] developed their RCC theory based on this single property of connectedness. The RCC theory is a superset of Egenhofer's theory. It can even describe relationships with concave objects by using a convex hull operator.

Of course, there exists a strong interdependency between the way of defining basic geometric objects and a set of corresponding spatial relations that can hold between these objects. Each of the above mentioned approaches defines a set of spatial (topological) relationships which are outlined in the next section.

3.2.2 Spatial Relations

Egenhofer's approach distinguishes eight mutually exclusive relations (out of $9^2 = 81$ different cases). The other cases can be eliminated since the above mentioned restrictions on sets have to hold. The remaining relations cover all possible cases. The 9-intersection is defined as a matrix.

$$I_n(\lambda_i, \lambda_j) = \begin{pmatrix} \partial\lambda_i \cap \partial\lambda_j & \partial\lambda_i \cap \lambda_j^o & \partial\lambda_i \cap \overline{\lambda_j} \\ \lambda_i^o \cap \partial\lambda_j & \lambda_i^o \cap \lambda_j^o & \lambda_i^o \cap \overline{\lambda_j} \\ \overline{\lambda_i} \cap \partial\lambda_j & \overline{\lambda_i} \cap \lambda_j^o & \overline{\lambda_i} \cap \overline{\lambda_j} \end{pmatrix}$$

With this definition the eight cases (disjoint, meet, overlap, equal, covers/coveredBy, contains/inside) can be easily characterized by the distinction between empty and non-empty intersections. For instance, the *contains* relation is specified by the 9-intersection as follows.

$$I_5(\lambda_i, \lambda_j) = \begin{pmatrix} \emptyset & \emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix}$$

Clementini et al. have to deal with $4^4 = 256$ different cases caused by taking into account the dimension of intersections. The number of cases can be reduced to a total of 52 real cases considering the restrictions on objects. They further reduced this still large number of possible relationships to five with the help of an object calculus. These five binary topological relations (touch, overlap, cross, in, disjoint) are mutually exclusive and cover all possible cases (see [Clementini et al., 1993] for a proof). For instance, the *in* relation is defined as follows: object λ_2 is *in* object λ_1 if the intersection between λ_1's and λ_2's region is equal to λ_2 and the interiors of their regions intersect. It is transitive and applies to every situation.

$$<\lambda_2, in, \lambda_1> \Leftrightarrow (\lambda_1 \cap \lambda_2 = \lambda_2) \wedge (\lambda_1^o \cap \lambda_2^o \neq \emptyset)$$

Randell et al. define nine spatial relations (that are similar to Egenhofer's set) in terms of a single primitive relation "C(x,y)" read as "x is connected with y." The authors also introduce an operator "conv(x)" which computes the convex hull of a possibly concave object. Its definition enables reasoning with concave objects. This approach is motivated by the idea that spatial databases might easily compute whether the single relation C(x,y) holds between two objects in the database. Further deductions could be based on this primitive relation. For instance, the relation "x is a part of y" (denoted as P(x,y)) is defined as follows.

$$P(x, y) \equiv \forall z : (C(z, x) \supset C(z, y))$$

3.2.3 Description Logic

This section gives a brief introduction to some aspects of description logic (DL) theory. We do not attempt to give a thorough overview and formal account of DL theory. However, we try to summarize the notions important for VL theory. We refer to [Brachman et al., 1991; Woods and Schmolze, 1992; Borgida, 1995] for more complete information about description logic theory.

DL theories are based on the ideas of structured inheritance networks [Brachman and Schmolze, 1985]. In a DL, a factual world consists of named individuals and their relationships that are asserted through binary relations. Hierarchical descriptions about sets of individuals form the terminological knowledge. Descriptions (or terms) about sets of individuals are called *concepts* and binary relations are called *roles*. Descriptions consist of identifiers denoting concepts, roles, and individuals, and of description constructors.

For instance, consider the following description with the intended meaning "a circle that touches only circles" that contains concept names (e.g. circle), role names (e.g. touching), and constructors (e.g. $\sqcap$ and $\forall$).

circle_touching_only_circles $\doteq \mathsf{circle} \sqcap \forall\, \mathsf{touching} \,.\, \mathsf{circle}$

The formal semantics of description terms is given denotationally. It uses a set $\Delta^{\mathcal{I}}$ of domain values and an interpretation function $\cdot^{\mathcal{I}}$mapping concept (or role) descriptions to subsets of $\Delta^{\mathcal{I}}$ (or $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$) (see below for more details).

Syntax

Let C be a set of concept names that is disjoint from the set R of role names. Any element of C is a *concept term*. If C and D are concept terms, $\mathsf{R} \in R$ is an arbitrary role, $n > 1$, and $m > 0$ ($n, m \in \mathbb{N}$), 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*)
- $\exists_{\leq m}\, \mathsf{R}$ (*at most number restriction*)
- $\exists_{\geq n}\, \mathsf{R}$ (*at least number restriction*)
- $\exists_{\leq m}\, \mathsf{R} \,.\, \mathsf{C}$ (*qualifying at most number restriction*)
- $\exists_{\geq n}\, \mathsf{R} \,.\, \mathsf{C}$ (*qualifying at least number restriction*).

The concept language of a DL is obtained recursively by starting from a set of names for concepts and roles, and forming more complex terms by applying description constructors. A knowledge base (also called terminology or TBox) contains definitions for concepts. Definitions characterize concepts as primitive or defined (operators $\doteq$ or $\sqsubseteq$, respectively). A specification of a *primitive* concept represents conditions that are necessary but *not* sufficient. The specification of a *defined* concept represents conditions that are both necessary *and* sufficient. Primitive and defined roles are similarly specified. A role R is considered as a binary relation and, loosely speaking, the elements of the "right-hand side" are called the *fillers* of the role R.

A considerable variety of description constructors is available, for example, unary (e.g. $\neg$) and binary operators (e.g. $\wedge$, $\vee$). A concept term can also be given as a restriction for role fillers. *Number restrictions* specify the maximum or minimum number of allowed fillers (e.g. $\exists_{\leq 5}\, \mathsf{touching}$, $\exists_{\geq 1}\, \mathsf{inside}$). *Value restrictions* allow only fillers that are individuals of a specific concept (e.g. $\forall\, \mathsf{touching} \,.\, \mathsf{arrow}$). Value and number restrictions may also be combined (e.g. $\exists_{\geq 1}\, \mathsf{touching} \,.\, \mathsf{arrow}$). Roles with an (implicit) '$\exists_{\leq 1}$' number restriction are called *attributes* or *features*.

In the DL considered in this chapter, the terminology must not contain cyclic definitions. Furthermore, a concept name must occur only once on the left-hand side in the definitions of a terminology. A terminology can also contain a set of disjointness assertions among concepts and among roles.

The assertional language of a DL is designed for stating constraints for concept or role membership that apply to a particular domain or world. The set of assertions (ABox) has to comply to the definitions declared in the TBox.

The language for representing knowledge about individuals is introduced. An *ABox* $\mathcal{A}$ is a finite set of assertional axioms which are defined as follows.

Let O be a set of individual names. If C is a concept term, R a role name, and $\mathsf{a}, \mathsf{b} \in O$ are individual names, then the following expressions are *assertional axioms*:

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

For instance, if we assume a TBox with the concept description of a "circle touching only circles" as given above, we can define individuals representing elements from our VL domain. The following conjunctive ABox assertions (denoted with the operator ':') use named individuals (e.g. circle_1) and concept names (e.g. circle) or concept expressions (e.g. $\exists_{\leq 1}$ touching).

$\mathsf{circle\_1} : \mathsf{circle},\ \mathsf{circle\_2} : \mathsf{circle},\ \mathsf{circle\_3} : \mathsf{circle},\ \mathsf{rectangle\_1} : \mathsf{rectangle},$

$\langle \mathsf{circle\_1}, \mathsf{circle\_2} \rangle : \mathsf{touching},\ \langle \mathsf{circle\_3}, \mathsf{rectangle\_1} \rangle : \mathsf{touching},$

$\mathsf{circle\_1} : \exists_{\leq 1}\ \mathsf{touching},\ \mathsf{circle\_2} : \exists_{\leq 1}\ \mathsf{touching}$

Based on the semantics explained below, a DL reasoner will infer that circle_1 and circle_2 are members of the concept circle_touching_only_circles while circle_3 is not a member of this concept.

Semantics

Let C be the set of concept names and R the set of role names. The model-theoretic semantics of a DL is based on the notion of an *interpretation*. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ (the domain) and an interpretation function $\cdot^{\mathcal{I}}$. The interpretation function maps each concept name C to a subset $\mathsf{C}^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name R to a subset $\mathsf{R}^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Let the symbols C, D be concept expressions, R be a role name, $n > 1$, and $m > 0$ ($n, m \in \mathbb{N}$). Then the interpretation function can be extended to arbitrary concept and role terms as follows ($\| \cdot \|$ denotes the cardinality of a set):

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

In the TBox the two special symbols '$\doteq$' and '$\sqsubseteq$' are used for introducing defined and primitive concepts, respectively. An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ iff it satisfies

- $\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$ for all terminological axioms $\mathsf{C} \sqsubseteq \mathsf{D}$ in $\mathcal{T}$, and
- $\mathsf{C}^{\mathcal{I}} = \mathsf{D}^{\mathcal{I}}$ for all terminological axioms $\mathsf{C} \doteq \mathsf{D}$ in $\mathcal{T}$.

Primitive concepts are introduced whenever there is no need or not enough knowledge for completely describing a concept, i.e. a primitive concept always 'signals' to users that there exists something that is not modeled and thus is outside of a DL.

The semantics of ABox assertions is defined analogously. Let the individual names a, b be elements of the set of individual names O. An interpretation $\mathcal{I}$ additionally satisfies an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ iff it satisfies

- $\mathsf{a}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$ for all assertional axioms in $\mathcal{A}$ of the form $\mathsf{a} : \mathsf{C}$, and
- $(\mathsf{a}^{\mathcal{I}}, \mathsf{b}^{\mathcal{I}}) \in \mathsf{R}^{\mathcal{I}}$ for all assertional axioms in $\mathcal{A}$ of the form $\langle \mathsf{a}, \mathsf{b} \rangle : \mathsf{R}$.

Reasoning Services

One of the basic reasoning services for a description logic formalism is computing the subsumption relationship between concept terms. This inference is needed in the TBox to build a hierarchy of concept names w.r.t. specificity.

The notion of a model is used to define the reasoning services that a DL inference engine has to provide, i.e. the engine can prove for concept specifications whether the following conditions hold:

- a term C *subsumes* another term D iff $\mathsf{D}^{\mathcal{I}} \subseteq \mathsf{C}^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$;

- a term C is *coherent/satisfiable* iff there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $\mathsf{C}^{\mathcal{I}} \neq \emptyset$;
- terms C and D are *disjoint* iff $\mathsf{D}^{\mathcal{I}} \cap \mathsf{C}^{\mathcal{I}} = \emptyset$ for all models $\mathcal{I}$ of $\mathcal{T}$;
- terms C and D are *equivalent* iff $\mathsf{D}^{\mathcal{I}} = \mathsf{C}^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$.

Proper DL systems (i.e. implementations of a DL) are guided by this semantics and implement these inference services. They usually distinguish two reasoning components. The *terminological reasoner* or *classifier* operates on the TBox and classifies named concepts with respect to subsumption relationships between them and organizes them into a taxonomy. The classifier automatically performs normalization of concept definitions as well as consistency checking operations and offers retrieval facilities about the status of the classification hierarchy. The forward-chaining *assertional reasoner* or *realizer* operates on the ABox in accordance with the definitions in the TBox and recognizes and maintains the concept and role membership of individuals. Assertional reasoners usually support a query language for accessing stated and deduced constraints. Some query languages offer the expressiveness of the full first-order calculus. The expressiveness and tractability of a particular DL depends on the variety of employed description constructors. Various complexity results for subsumption algorithms for specific description logics are summarized in [Woods and Schmolze, 1992].

3.2.4 Extension of Description Logic: Concrete Domains

Standard DL systems usually cannot deal with concepts defined with the help of arithmetic. For instance, it is not possible to specify a *defined* concept SmallCircle that describes every circle whose radius is less than 10mm. It is only possible to specify SmallCircle as a *primitive* concept (which can never automatically be recognized for an individual) and to assert this concept membership for an individual externally. Some DL systems offer extra-logical, user-defined test functions that may assert the property (radius less than 10mm) automatically. However, these functions and their related concepts escape the DL semantics and prevent any reasoning. For instance, a concept VerySmallCircle resembling circles with a radius less than 5mm should be recognized as a specialization (subconcept) of SmallCircle. The idea of incorporating concrete domains into DL theory is to extend semantics and subsumption in a corresponding way (see [Baader and Hanschke, 1991; Hanschke, 1996]). The concrete domain approach distinguishes between an *abstract* and a *concrete* part of a domain. The languages support operators for specifying predicates that apply to individuals from the concrete domain (e.g. circles in two-dimensional space).

The above mentioned concepts SmallCircle and VerySmallCircle could be easily specified with this extension as *defined* concepts. A reasoner would immediately recognize the subsumption relationship between these concepts. However, the concrete domain approach can only define concepts dependent on their own properties that are expressed with concrete predicates. Spatial relations cannot be adequately defined with the operators offered by "standard" DL languages. A solution for this problem is presented in Chapter 7.

3.2.5 Applying Description Logic to Visual Language Theory

We argue that the main characteristics of DL systems are directly applicable to VL theory (see also [Haarslev, 1995] and Section 3.4 for example applications):

- The TBox language is used to define VL elements as concept definitions. They are based on primitive concepts representing basic geometric objects (e.g. region, line, point). The primitive concepts form the roots of the taxonomy and are viewed as elementary lexical tokens. Defined concepts express (intermediate) semantic categories and are based on specializations of these primitive concepts.

- The classifier automatically constructs and maintains the specialization hierarchy of VL elements (defined as concepts). This hierarchy is used by the realizer to control the assertional reasoning process.

- Database-like assertion and query languages are used to state and retrieve spatial knowledge about individuals of VL sentences. Example sentences may be entered into the ABox by asserting primitive concept memberships for geometric objects and spatial relationships between objects (as role fillers).

- The forward-chaining realizer automatically recognizes the most specialized concept membership (i.e. the semantic category of VL elements) of individuals (e.g. input tokens). It is the main source for driving the recognition process and is utilized as a general visual parser.

- The automatic detection of inconsistent concept definitions or individuals is an important advantage of this approach. It is used to detect unsound (e.g. inconsistent) formal specifications (TBox) or erroneous parser input (e.g. errors in syntax or semantics).

Other (but still non-standard) characteristics are also very useful:

- The retraction of facts (stated in the ABox) is useful for supporting incremental and predictive parsing techniques in the editing process. Non-monotonic changes of users are automatically recognized and obsolete deductions retracted (e.g. in the DL system Classic).

- Default reasoning can make useful assumptions about parser input based on incomplete information.

- A DL extended to handle concrete domains could be very useful (see also Chapter 7). The definition of VL elements and the possible spatial relationships between them could be solely based on DL theory with a concrete domain over $\mathbb{R}$. The need for an extra-logical component that recognizes geometric features and asserts them to the DL system would be obsolete.

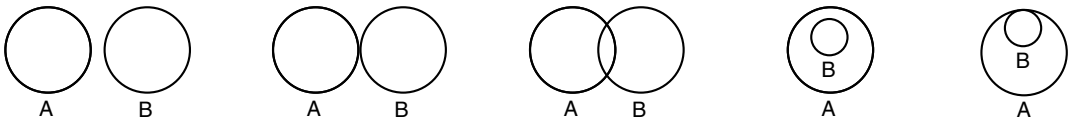


Figure 3.1: Primitive relations between A and B.

In the next sections, we apply the following characteristics of description logic theory to visual languages.

- TBox reasoning is used to design visual languages since VL elements are described by concept definitions.
- ABox reasoning is used to parse actual diagrams representing sentences of a visual language.

3.3 GenEd: Implementing the Theory

The logical framework described in the previous sections forms the basis for the generic object-oriented editor GenEd. The next sections describe GenEd's user interface and implementation in more detail.

3.3.1 Spatial Logic Implemented by Built-in Parser

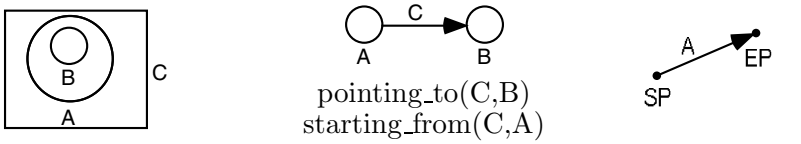
The implementation of geometric objects and recognition of spatial relations uses well-known computer graphics techniques for reasons of efficiency. The semantics of these algorithms is still specified within our theory (see [Wessel, 1996] for a complete treatment).

Geometric Objects

GenEd offers a set of predefined geometric objects (similar to other object-oriented graphic editors) that can be used to design examples of particular notations. For instance, supported primitive objects are points, (directed) line segments, line segment chains, (spline) polygons, circles, etc. These objects can be used to compose other objects (e.g. ovals).

Spatial Relations

GenEd recognizes seven primitive spatial relations (*disjoint, touching, intersecting, containing/inside, covering/covered_by*) which may hold between objects (see Figure 3.1). We deliberately omitted the 'equal' relation in the design of GenEd but it could be integrated rather trivially. GenEd also computes the dimension of the intersection of objects, if applicable. The semantics is defined in analogy to Clementini et al. [Clementini et al., 1993] (see Section 3.2.2). The relations have a parameterized 'fuzziness' compensating



directly_contains(C,A) pointing_to(C,B) starting_from(C,A) linked_with(A,B) part_of(EP,A)

Figure 3.2: Higher-level relations.

for inexact positioning of objects (caused by users or scaling factors) and floating-point arithmetic. The fuzziness is defined as a threshold value depending on the global scale factor and the size of the related objects. In contrast to several other approaches for spatial relations (e.g. see [Haarslev, 1996b]) GenEd can also deal with concave objects. The seven relations mentioned above can also be computed for arbitrary concave objects using standard algorithms from computational geometry. Additionally, an arbitrary collection of objects may be grouped together and treated as a *composite* object. A semantics for composite objects has been defined in analogy to [Clementini and Di Felice, 1997].

The following higher-level relations (that are also applicable to composite objects) have been implemented with the help of the above mentioned seven relations (e.g. see Figure 3.2).

- The relation *directly_contains/inside* is a subset of the containing/inside relation. A region λ_1 directly contains a regions λ_2 iff λ_1 contains λ_2 and there exists no other region in λ_1 that contains λ_2.

  $$\mathsf{directly\_contains}(\lambda_1, \lambda_2) \equiv \mathsf{containing}(\lambda_1, \lambda_2) \wedge \neg\, \exists\, \lambda_x : \mathsf{containing}(\lambda_1, \lambda_x) \wedge \mathsf{containing}(\lambda_x, \lambda_2)$$

- The relation *linked_with* resembles the connectivity of two-dimensional objects such as circles, rectangles, etc. It is computed for any two-dimensional object touching a line or an arrow that eventually leads (possibly via a chain) to another two-dimensional object. The formal specification is given in [Wessel, 1996].

- The direction of line segments is recorded in the relations *starting_from* and *pointing_to* which only apply to arrows.

- Partonomies are handled with the relation *has-part/part-of*. GenEd automatically asserts part-of relationships for end points of line segments and arrows and for components of composite objects.

3.3.2 User Interface

Figure 3.3 shows a screen shot of the user interface. It contains several (scrollable) panes and a menu bar at the top. The three horizontal panes below the menu bar offer the

selection of object types and the setting of drawing attributes for elements and text. The left vertical pane shows a variety of modifiable parameters for controlling display options and scaling factors. The center pane (workspace) contains a petri net for the reader-writer problem (see Section 3.4.1 for explanations). It displays the petri net elements (place as circle, transition as rectangle, edge as (spline) arrow, token as bullet, capacity label as gray number). The elements are also labeled with the concept names as computed by the classification phase of the spatial parser. The right pane is used to inform the user about computed concept memberships, role fillers, etc. The horizontal pane below the three vertical panes is the command pane. Users have the choice whether they enter commands as gestures (mouse movement, clicks) or as text commands. The pane at the bottom always shows object-sensitive documentation about available gestures.

Users can always select a collection of elements in the workspace with an enclosing box and aggregate them into a composite object. The contents of the workspace can be zoomed in or out (see Figure 3.4b for a magnified selection of the petri net). In general, GenEd offers many operations on objects that are also available in commercial graphic editors (create, delete, copy, move, scale, rotate, hide, show, inspect, arrangement, save, restore, undo list).

The general procedure for working with GenEd is as follows. The user loads a DL specification of a visual notation into GenEd. This specification has to comply with GenEd's built-in spatial logic. A new drawing may be created in the workspace (e.g. center window in Figure 3.3) or an existing one loaded. The built-in spatial parser analyzes a drawing in accordance with the spatial logic and creates ABox individuals and assertions resembling the elements of the drawing and their spatial relationships. Afterwards GenEd invokes the DL system. A protocol of the classification process can be displayed in GenEd's rightmost vertical window. GenEd optionally shows the concept membership of drawing elements and several other useful information (see center window in Figure 3.3).

GenEd supports two reasoning modes. While GenEd is in *incremental* mode, it records differences to previous states and reports these differences to the ABox. The reasoning process is invoked to automatically analyze drawings after every modification and to give the user an immediate feedback. If the *batch* mode is set, drawings are always analyzed from scratch and the user has to start the reasoning process manually. It is worth noting that users may attach special handles to arbitrary objects. These handles can be used to fix relative positions between objects or to define stretchable lines whose end points might be fixed to objects. Primitive and composite objects may be stored in and retrieved from a user-defined library. Figure 3.4a shows a submenu displaying visualizations of petri net places stored in the library. The workspace can be saved in and loaded from a file.

3.3.3 Implementation

GenEd is implemented in Common Lisp using the Common Lisp Object System (CLOS) and the Common Lisp Interface Manager (CLIM) as interface toolkit. The classification of concepts and the parsing of actual drawings take place by using CLASSIC [Brachman et al., 1991; Brachman, 1992] as DL system. CLASSIC provides a sound and complete

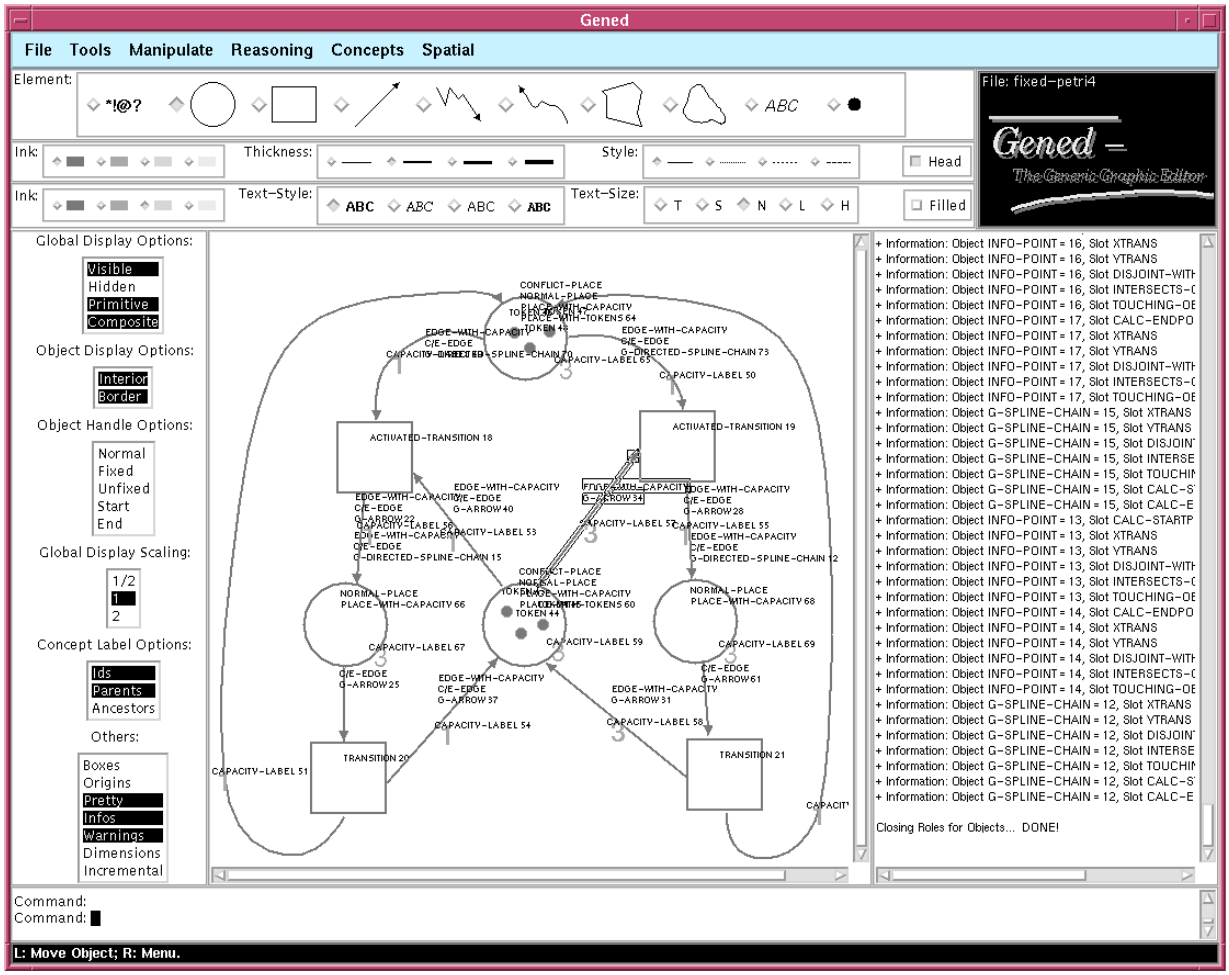
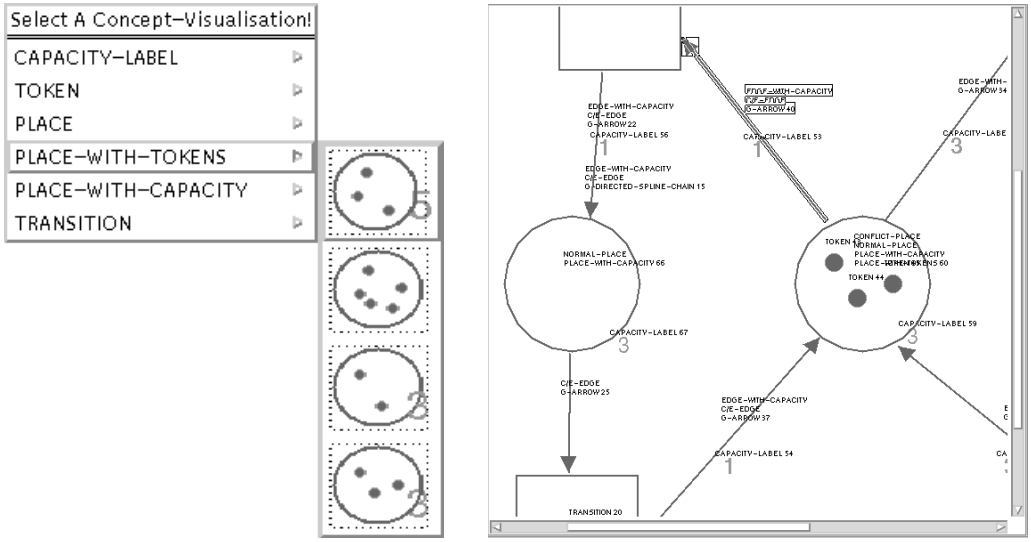


Figure 3.3: GenEd: petri net for reader-writer problem (simplified).

inference algorithm with polynomial time complexity (see [Borgida and Patel-Schneider, 1994] for comments on the semantics of the full CLASSIC language and for an explanation of its inference procedure). Besides a stable inference engine, the CLASSIC implementation provides also an explanation framework which is important for practical work. CLASSIC is also implemented in Common Lisp. GenEd consists of 28 modules with a total of about 300 KB source code (without CLIM, CLOS, and CLASSIC). GenEd is fully implemented with the features described in this chapter.

3.4 Examples: Diagrammatic Notations

We demonstrate two visual notations whose specifications were created with GenEd. Place-transition petri nets are used as first notation. The second notation defines simple entity-relationship (ER) diagrams. We present these examples in order to demonstrate the expressiveness of our specification language and the reasoning capabilities of GenEd. We



(a) Library menu with submenu. **(b)** Magnified selection of a petri net.

Figure 3.4: Two figures: library menu and zoom of a petri net.

assume a few primitive and mutually disjoint concepts such as rectangle, circle, diamond, line, and text. We also rely on spatial relations (touching, containing, linked_with, text_value) representing relationships between geometric objects. These examples have been developed within CLASSIC's DL but are presented in the following sections with a more powerful DL for reasons of brevity and simplicity.

Throughout the next sections, we use a top-down approach for developing and presenting diagram specifications, i.e. they get more and more complex. We start with specifications of basic geometric objects and compose these specifications stepwise to more specialized specifications describing higher-level elements of a diagram.

3.4.1 Petri Nets

In the following we give the reader an idea how a specification of visual language for fragments of petri nets might be developed. A *petri net* (e.g. see Figure 3.5) is a triple $N = (P, T, E)$ with P a set of places, a set $T \neq \emptyset$ of transitions, and a relation $E \subset (P \times T) \cup (T \times P)$ representing edges.

A tuple $N = (P, T, E, C, W, M)$ defines a *place-transition net* if the following conditions hold. The tuple (P, T, E) is a petri net with places P and transitions T. The capacity for each place is defined by $C : P \to \mathbb{N} \cup \{\omega\}$. $W : E \to \mathbb{N} - \{0\}$ specifies the weight of every edge. The initial marking is defined by $M : P \to \mathbb{N} \cup \{\omega\}$, with $\forall p \in P : M(p) \leq C(p)$.

We only outline the design of the specification for place-transition nets. We define concepts representing legal constellations for places, transitions, and edges. A petri net is specified as a composite object consisting of at least five parts.

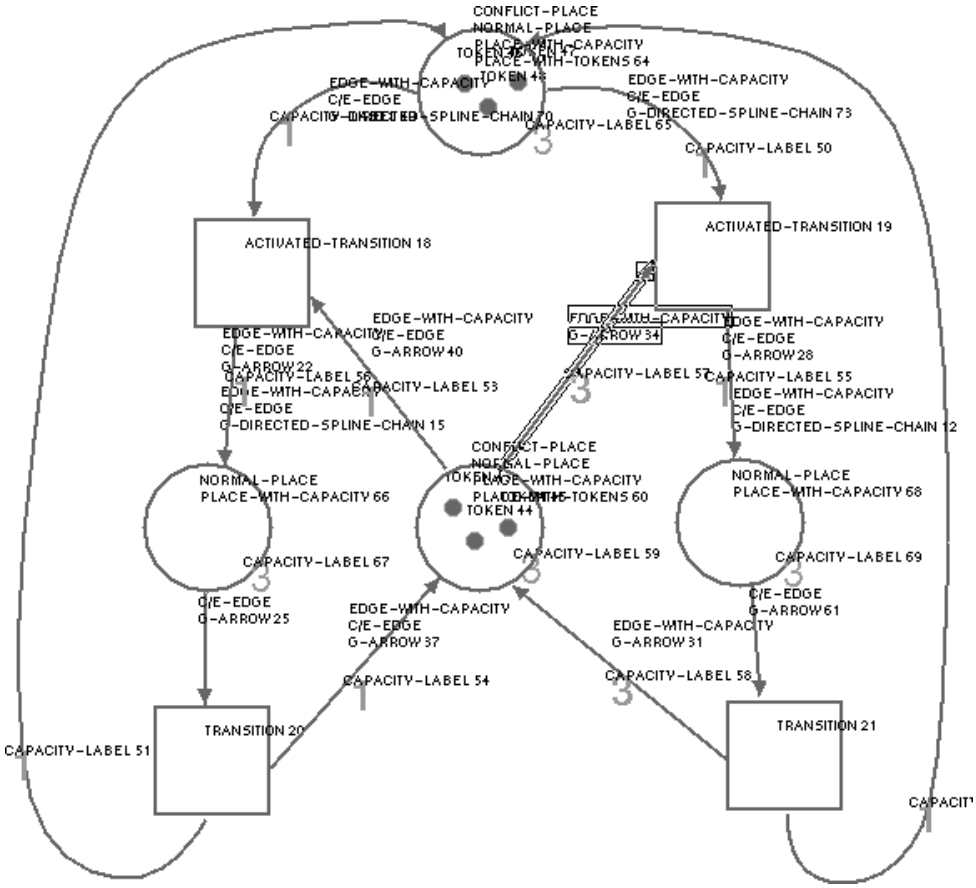


Figure 3.5: Zoom of the petri net shown in Figure 3.3.

$$\textbf{petri\_net} \doteq \mathsf{composite\_thing} \sqcap \exists_{\geq 5}\, \mathsf{has\_parts} \sqcap \exists_{\geq 1}\, \mathsf{has\_parts}\,.\,\mathsf{place} \sqcap \forall\, \mathsf{has\_parts}\,.\,(\neg\mathsf{rectangle} \sqcup \mathsf{transition}) \sqcap \forall\, \mathsf{has\_parts}\,.\,(\neg\mathsf{arrow} \sqcup \mathsf{edge})$$

It is important to note that a term such as $\forall\, \mathsf{has\_parts}\,.\,(\neg\mathsf{circle} \sqcup \mathsf{place})$, that one might have expected, would overconstrain the definition of the concept petri_net since tokens are currently also represented as circles. We could rectify this problem by splitting circles into filled (token) and transparent (place) circles. Petri nets are specialized to place-transition nets after defining capacity labels, places with capacity, tokens, places with tokens, edges with capacity, and active transitions.

$$\textbf{place\_transition\_net} \doteq \mathsf{petri\_net} \sqcap \exists_{\geq 1}\, \mathsf{has\_parts}\,.\,\mathsf{place\_with\_token}$$

An interesting special case of a place-transition net is a *predicate-event* net. All places and edges have a maximal capacity of one.

$$\forall p \in P\colon C(p) = 1 \wedge \forall (x, y) \in E : W(x, y) = 1$$

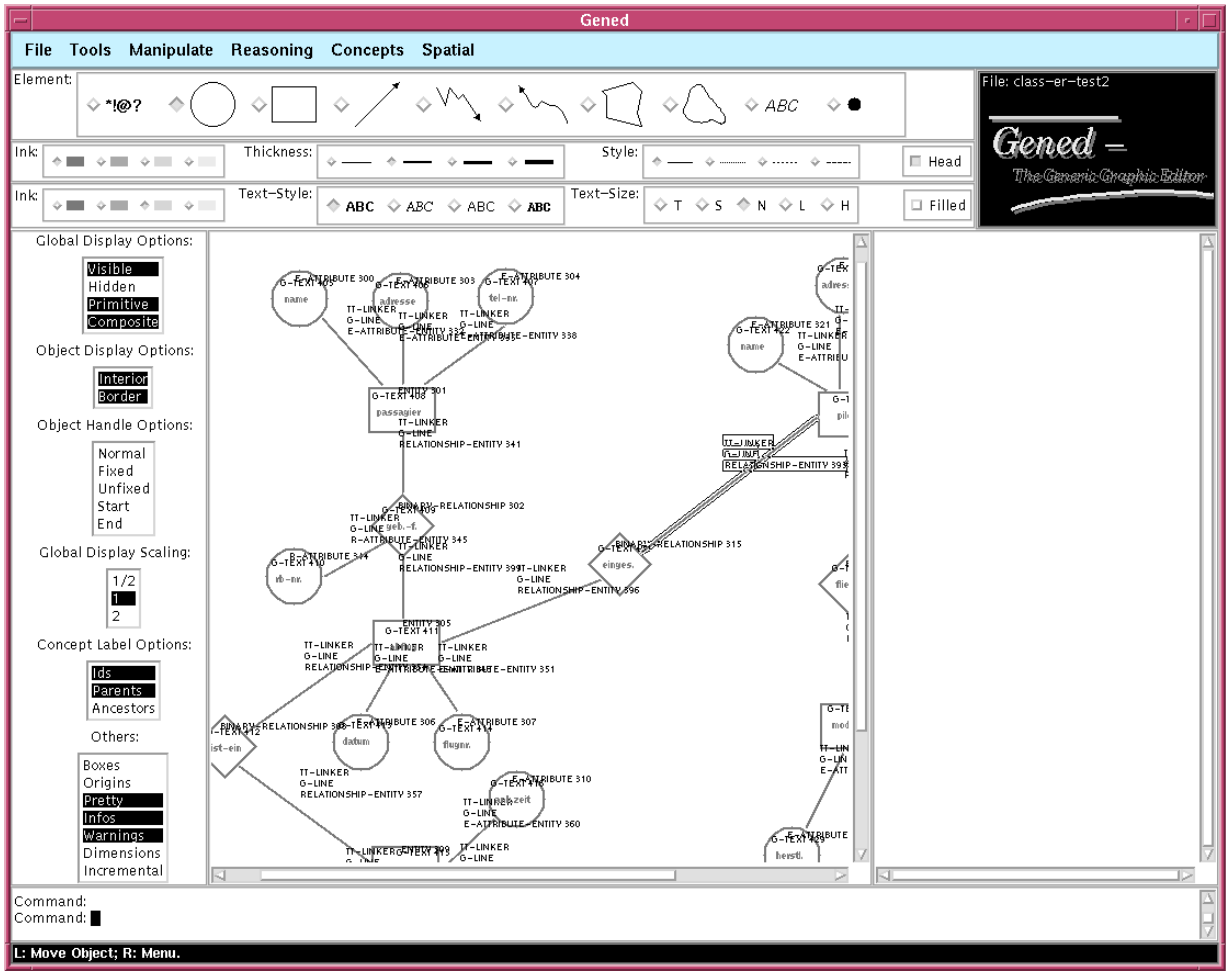


Figure 3.6: An ER diagram modeling airlines.

$$\textbf{predicate\_event\_net} \doteq \mathsf{place\_transition\_net} \sqcap \forall\, \mathsf{has\_parts}\,.\,(\neg\mathsf{place} \sqcup \mathsf{predicate\_event\_place}) \sqcap \forall\, \mathsf{has\_parts}\,.\,(\neg\mathsf{arrow} \sqcup \mathsf{predicate\_event\_edge})$$

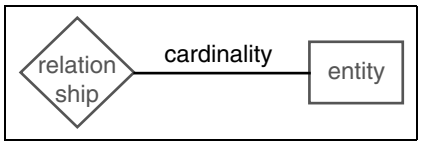
The definition of predicate-event places and transitions are omitted. There are still other interesting concepts characterizing special petri net elements that are left out due to brevity. However, the next section discusses entity-relationship diagrams which are specified in almost full detail.

3.4.2 Entity-Relationship Diagrams

Our definition of a subset of entity-relationship (ER) diagrams was inspired by [Serrano, 1995]. Figure 3.6 shows a part of a petri net modeling relationships in an airline company. Since this modeling does not employ concrete domains, we replace the reference to "text strings" (e.g. 1) by corresponding disjoint atomic concept names (e.g. $\mathsf{value\_1}$). We use the term $\exists_{=n} \ldots$ as an abbreviation for $(\exists_{\leq n} \ldots) \sqcap (\exists_{\geq n} \ldots)$.

Connectors

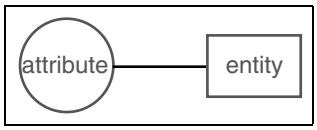
A *relationship-entity* connection is a line that touches exactly one text label (expressing cardinality) and two other regions (rectangle or diamond). A *cardinality* is a text string with values chosen from the set $\{1, m, n\}$.



$$\textbf{relationship\_entity} \doteq \mathsf{line} \sqcap \exists_{=3}\, \mathsf{touching} \sqcap \exists_{=1}\, \mathsf{touching} \,.\, \mathsf{text} \sqcap \exists_{=1}\, \mathsf{touching} \,.\, \mathsf{rectangle} \sqcap \exists_{=1}\, \mathsf{touching} \,.\, \mathsf{diamond}$$

$$\textbf{cardinality} \doteq \mathsf{text} \sqcap \forall\, \mathsf{text\_value} \,.\, (\mathsf{value\_1} \sqcup \mathsf{value\_m} \sqcup \mathsf{value\_n}) \sqcap \exists_{=1}\, \mathsf{touching} \sqcap \forall\, \mathsf{touching} \,.\, \mathsf{relationship\_entity}$$

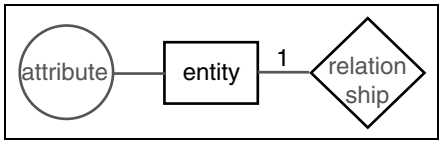
An *attribute-entity* connection is a line that touches only two regions (circle or rectangle) and no text string.



$$\textbf{attribute\_entity} \doteq \mathsf{line} \sqcap \exists_{=2}\, \mathsf{touching} \sqcap \forall\, \mathsf{touching} \,.\, (\mathsf{circle} \sqcup \mathsf{rectangle}) \sqcap \exists_{=1}\, \mathsf{touching} \,.\, \mathsf{rectangle} \sqcap \exists_{=1}\, \mathsf{touching} \,.\, \mathsf{circle}$$

Entities

An *entity* is a rectangle that contains its name. It touches one relationship-entity and optionally some attribute-entity connectors. It is linked with a diamond.

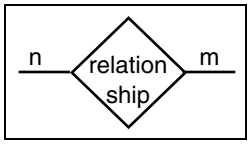


$$\textbf{named\_region} \doteq \mathsf{region} \sqcap \exists_{=1}\, \mathsf{containing} \sqcap \forall\, \mathsf{containing} \,.\, \mathsf{text}$$

$$\textbf{entity} \doteq \mathsf{rectangle} \sqcap \mathsf{named\_region} \sqcap \exists_{\geq 1}\, \mathsf{linked\_with} \,.\, \mathsf{diamond} \sqcap \forall\, \mathsf{linked\_with} \,.\, (\mathsf{circle} \sqcup \mathsf{diamond}) \sqcap \exists_{\geq 1}\, \mathsf{touching} \,.\, \mathsf{relationship\_entity} \sqcap \forall\, \mathsf{touching} \,.\, (\mathsf{attribute\_entity} \sqcup \mathsf{relationship\_entity})$$

Relationships

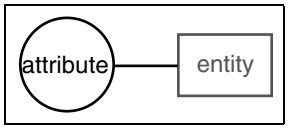
A *relationship* is a diamond that contains its name. It touches one relationship-entity and optionally some attribute-entity connectors. It is linked with two entities.



$$\textbf{relationship} \doteq \text{diamond} \sqcap \text{named\_region} \sqcap \exists_{=2}\, \text{linked\_with} \sqcap \forall\, \text{linked\_with} \,.\, \text{entity} \sqcap$$
$$\exists_{=2}\, \text{touching} \sqcap \forall\, \text{touching} \,.\, \text{relationship\_entity} \sqcap$$
$$\exists_{\leq 2}\, \text{touching} \,.\, \forall\, \text{touching} \,.\, \forall\, \text{text\_value} \,.\, \text{value\_1} \sqcap$$
$$\exists_{\leq 1}\, \text{touching} \,.\, \forall\, \text{touching} \,.\, \forall\, \text{text\_value} \,.\, \text{value\_m} \sqcap$$
$$\exists_{\leq 1}\, \text{touching} \,.\, \forall\, \text{touching} \,.\, \forall\, \text{text\_value} \,.\, \text{value\_n}$$

Attributes

An *attribute* is a circle that contains its name. It touches one attribute-entity connector and is linked with an entity.



$$\textbf{attribute} \doteq \text{circle} \sqcap \text{named\_region} \sqcap \exists_{=1}\, \text{linked\_with} \sqcap \forall\, \text{linked\_with} \,.\, \text{entity}$$

3.5 Example: Programming Language Pictorial Janus

In addition to the previous examples, we present the specification of the visual programming language Pictorial Janus (PJ). It is deliberately intended to demonstrate to the reader that the specification of PJ (and of similar notations with a realistic complexity) is a non-trivial task that requires automatic reasoning mechanisms.

Pictorial Janus [Kahn and Saraswat, 1990; Kahn et al., 1991] is a completely visual language for the domain of parallel programming. It is defined on purely pictorial terms. Figure 3.7 shows a simple PJ program for concatenating two lists. The next sections informally describe PJ's computational model and specify a subset of PJ's language elements. The informal description of PJ emphasizes the computational model since the language elements can be better described by their formal specification.

3.5.1 Computational Model of Pictorial Janus

PJ's syntax and static semantics are defined through topological relations which have to hold between language elements. Language elements are either represented as closed contours (agents, rules, ports, primitive functions, constraints, constants, arrays, bags) or as (directed) lines (links, channels, call arrows). Since PJ's syntax is purely based on topological relationships its language elements can have any shape, size, color, etc. provided the required relationships are still holding. These graphical features are available for the programmer for application-specific purposes. PJ's dynamic (operational) semantics can be defined by graphical transformation rules.

A PJ computation can be considered as a network of concurrently executing agents asynchronously communicating over point-to-point directional channels. The behavior of an

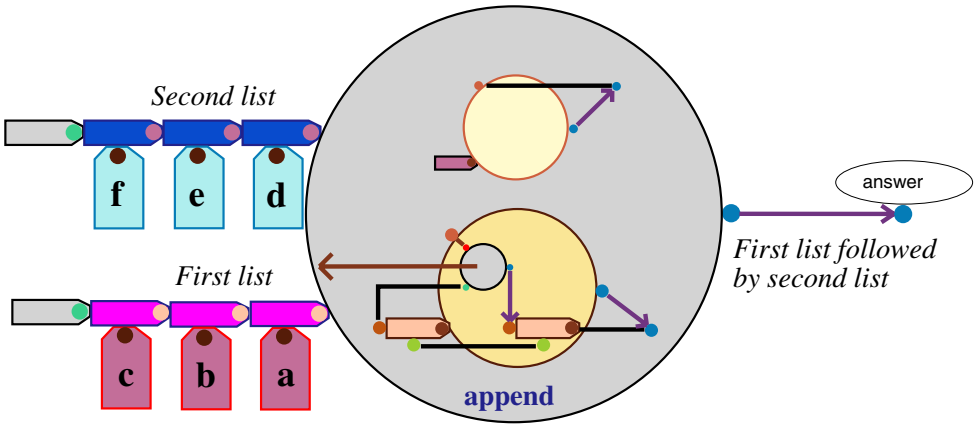


Figure 3.7: Append of two lists in Pictorial Janus (original art by Ken Kahn).

agent is defined by a collection of rules. *Rules* describe conditions (guards) under which their agent will read incoming messages from channels and may place new messages on channels for which it has *tell* rights. They also define how an agent replaces itself by a new subnetwork of messages, channels, and agents. Messages may be channels or may contain channels.

Preconditions of a rule are denoted by elements that are (transitively) connected to the ports of this rule and that are allocated outside of this rule. These elements are called *askers* of this rule since they represent what is "asked" of the corresponding ports of the enclosing agent. A rule is indeterministically selected if the agent's arguments match the askers of several rules.

The body of a rule consists of all elements that are inside of this rule. They have to be (transitively) connected to either a port or asker of this rule. We call these elements *tellers* since they post (tell) new constraints on shared data structures. A so-called *tell right* is required for posting new constraints to non-local data structures. At most one tell right may exist for any data structure. Tell rights are first-class elements since they may be communicated over channels.

The execution of a PJ program (i.e. of the corresponding agents) can be specified by purely graphical transformation rules.

1. The elements connected to the ports of an agent (later on referred to as input data) must graphically match with the askers of an agent's rule. As the result of a successful match every link of the askers is connected to the corresponding input data of the agent.

2. An agent replaces itself with the body of the selected rule. The askers, ports, and contour of the rule disappear. Only the input data of the agent, now connected to elements of the rule body, and all elements of the rule body remain.

3. A rule may assign a value to a channel if the rule has a channel as asker (this proves that it has the tell right to this channel). An assignment takes place by connecting the value and the head of the channel with a link. Eventually links will shrink to length zero joining the assigned or matched value and its recipient.

The selection of rules—by matching their askers to input data of their agent—takes place as long as runnable agents exist. A PJ program either terminates or suspends depending on the state of the agents.

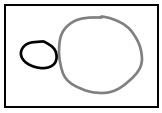
Keeping in mind these informal descriptions, we briefly explain the example program in Figure 3.7. The 'append' program concatenates two lists (displayed on the left side) and constructs a new list as result (its placeholder and the tell right are displayed on the right side). The big gray circle represents the append agent. It contains two mutually exclusive rules handling the iteration and the termination case. Connecting lines represent equality between elements. The agent is based on the following algorithm for concatenation. The first list is processed (i.e. copied) to the result placeholder until the end of the list is reached. The iteration is implemented by the bottom rule that concurrently removes one element from the first list, pushes it onto the result list, and recursively invokes its enclosing agent with the new arguments. The top rule fires if the list end (leftmost gray list element) is reached. It concurrently discards the list end, sets the list end of the result list equal to the list start of the second list (i.e. the concatenated new list is constructed), and afterwards terminates.

3.5.2 Language Elements of Pictorial Janus

This section defines language elements of PJ. We omit several language elements but present a selection of elements sufficient to understand the example program in Figure 3.8. It is important to note that the 'beautified' version of the append program in Figure 3.7 differs from the 'normalized' version in Figure 3.8. Any 'beautified' PJ program can be transformed to a 'normalized' version that is semantically equivalent and vice versa. The normalized version makes connections between ports and their arguments explicit. In the following the formal specification of PJ relies on this normalized version which simplifies the specification process. The formal specifications are illustrated with (sometimes slightly modified) quotations from the original PJ specification [Kahn and Saraswat, 1990] and with example configurations. However, the original PJ specification is somewhat ambiguous and contains cyclic definitions. This has been resolved in our formal specifications by deliberately deviating from the informal quotations. We either relax restrictions in order to prevent cycles or add additional restrictions in order to clarify ambiguities.

Ports

A port is an empty region which touches at most one other region.



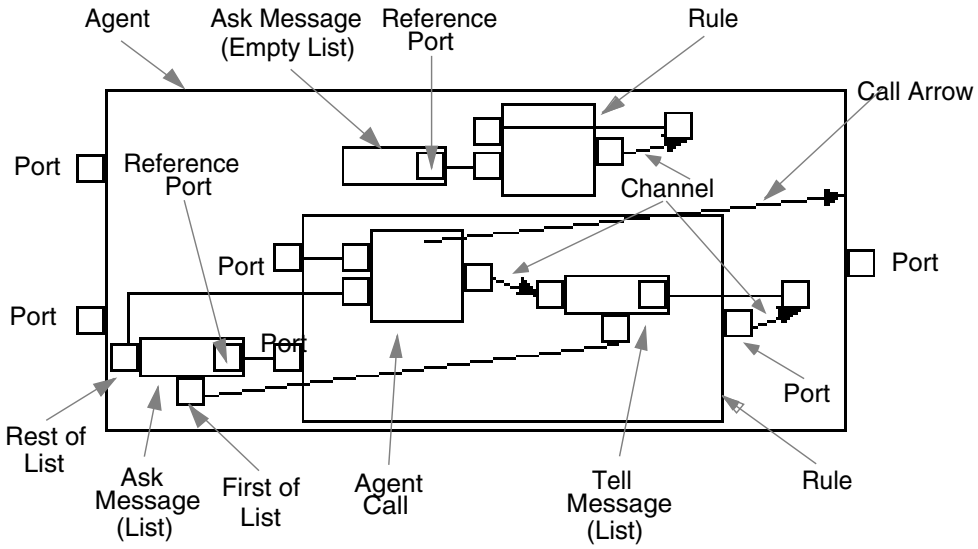


Figure 3.8: The normalized append program equivalent to the version in Figure 3.7. Regions are displayed as rectangles and annotated with DL concept names.

A region R is classified as empty_region iff there exists no PJ element inside of R. A port may touch any number of elements which are not regions but at most one other region. Ports serve as docking place for lines.

$$\textbf{empty\_region} \doteq \mathsf{region} \sqcap \exists_{\leq 0}\, \mathsf{containing}$$
$$\textbf{port} \doteq \mathsf{empty\_region} \sqcap \exists_{\leq 1}\, \mathsf{touching} \,.\, \mathsf{region}$$

We distinguish ports with respect to their relationship to other elements. A port may serve as a

- *reference* port identifying data structures
- *argument* port representing arguments of data and rules
- *single* port.

$$\textbf{reference\_port} \doteq \mathsf{port} \sqcap \exists_{=1}\, \mathsf{covered\_by} \sqcap \forall\, \mathsf{covered\_by} \,.\, \mathsf{term} \sqcap \exists_{\leq 0}\, \mathsf{touching} \,.\, \mathsf{region} \sqcap \exists_{\leq 1}\, \mathsf{touching} \,.\, \mathsf{segment} \sqcap \exists_{\leq 1}\, \mathsf{touching} \,.\, \mathsf{point}$$
$$\textbf{argument\_port} \doteq \mathsf{port} \sqcap \exists_{=1}\, \mathsf{touching} \,.\, \mathsf{term} \sqcap \exists_{\leq 0}\, \mathsf{covered\_by}$$
$$\textbf{single\_port} \doteq \mathsf{port} \sqcap \exists_{\leq 0}\, \mathsf{touching} \sqcap \exists_{\leq 0}\, \mathsf{covered\_by} \,.\, \mathsf{region}$$

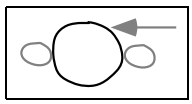
Any port can be linked or unused (i.e. not linked). However, an unused reference port indicates a semantic error since the element owning this reference port can never be referenced. Therefore, we restrict unused ports to be also argument ports.

$$\textbf{unused\_port} \doteq \mathsf{argument\_port} \sqcap \exists_{\leq 0}\, \mathsf{touching} \,.\, \mathsf{segment}$$

$$\textbf{linked\_port} \doteq \mathsf{port} \sqcap \exists_{\geq 1}\, \mathsf{linked\_with} \,.\, \mathsf{region}$$

Data Terms

A term is a (possibly empty) region. It touches only ports or tips of arrows.



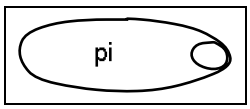
Terms are building blocks for defining rules, agents, and data structures. Terms can be divided into data and rule terms. A *data term* does not touch any arrow or link but has one reference port (i.e. it is covering this port). End points are only defined for arrows.

$$\textbf{term} \doteq \mathsf{region} \sqcap \exists_{\leq 0}\, \mathsf{covered\_by} \sqcap \forall\, \mathsf{touching} \,.\, (\mathsf{port} \sqcup \mathsf{arrow} \sqcup \mathsf{end\_point})$$

$$\textbf{data\_term} \doteq \mathsf{term} \sqcap \exists_{=1}\, \mathsf{covering} \sqcap \forall\, \mathsf{covering} \,.\, \mathsf{reference\_port} \sqcap \exists_{\leq 0}\, \mathsf{touching} \,.\, (\mathsf{arrow} \sqcup \mathsf{end\_point})$$

Constants

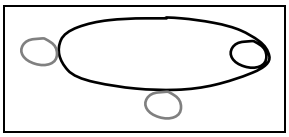
A constant is a data term that has no argument port and contains a constant value (represented as text).



$$\textbf{constant} \doteq \mathsf{data\_term} \sqcap \exists_{\leq 1}\, \mathsf{containing} \,.\, \mathsf{region} \sqcap \exists_{=1}\, \mathsf{containing} \,.\, \mathsf{text} \sqcap \exists_{\leq 0}\, \mathsf{touching} \,.\, \mathsf{port}$$

List elements

A list element is a data term with at most two argument ports. The empty list is a list element that has no argument ports.



$$\textbf{list} \doteq \mathsf{data\_term} \sqcap \exists_{\leq 2}\, \mathsf{touching} \,.\, \mathsf{region} \sqcap \exists_{\leq 1}\, \mathsf{containing} \,.\, \mathsf{region}$$

$$\textbf{empty\_list} \doteq \mathsf{list} \sqcap \exists_{\leq 0}\, \mathsf{touching} \,.\, \mathsf{region}$$

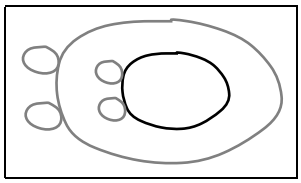
Rule Terms

Rule terms may not cover any region (and thus any port). Rule terms are building blocks for agent calls, rules, and agents.

$\textbf{rule\_term} \doteq \mathsf{term} \sqcap \exists_{\leq 0}\, \mathsf{covering} \,.\, \mathsf{region}$

Rules

A rule is a rule term with any number of argument ports. It has to be inside of an agent.

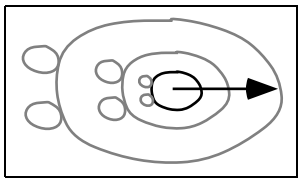


$\textbf{rule\_body} \doteq \mathsf{rule\_term} \sqcap \exists_{\geq 1}\, \mathsf{touching} \,.\, \mathsf{port} \sqcap \forall\, \mathsf{touching} \,.\, (\neg \mathsf{region} \sqcup \mathsf{argument\_port})$

$\textbf{rule} \doteq \mathsf{rule\_body} \sqcap \exists_{=1}\, \mathsf{inside} \,.\, \mathsf{rule\_body}$

Agent calls

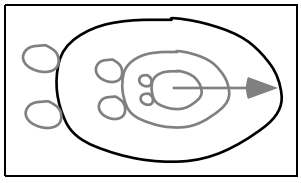
An agent call is a rule term with any number of argument ports. It has to be inside of a rule and has to contain the start point of a call arrow.



$\textbf{agent\_call} \doteq \mathsf{rule\_body} \sqcap \exists_{=1}\, \mathsf{inside} \,.\, \mathsf{rule} \sqcap \exists_{=1}\, \mathsf{intersecting} \,.\, \mathsf{call\_arrow} \sqcap \exists_{=1}\, \mathsf{containing} \sqcap$
$\forall\, \mathsf{containing} \,.\, \mathsf{end\_point} \sqcap \forall\, \mathsf{containing} \,.\, \forall\, \mathsf{part\_of} \,.\, \mathsf{call\_arrow}$

Agents

An agent is a rule term at the top level with any number of argument ports. It has to contain some rules.



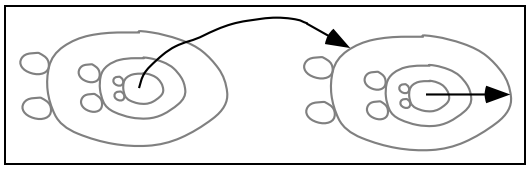
$\textbf{agent} \doteq \mathsf{rule\_body} \sqcap \exists_{\geq 1}\, \mathsf{containing} \,.\, \mathsf{rule\_body} \sqcap \exists_{\leq 0}\, \mathsf{inside} \,.\, \mathsf{region}$

Call Arrows and Channels

Arrows are used to denote an agent to be called or as representation for a tell right. A *tell right* is necessary for sending data to another agent, i.e. writing to a channel.

Call Arrows

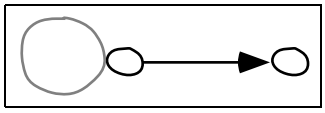
A call arrow is an arrow starting inside of an agent call and pointing to the outline of an agent.



$$\textbf{call\_arrow} \doteq \text{arrow} \sqcap \exists_{\geq 1}\ \text{intersecting} \,.\, \text{rule}$$
$$\textbf{recursive\_call\_arrow} \doteq \text{call\_arrow} \sqcap \exists_{=1}\ \text{covered\_by} \,.\, \text{agent}$$
$$\textbf{other\_call\_arrow} \doteq \text{call\_arrow} \sqcap \exists_{=1}\ \text{touching} \,.\, \text{agent}$$

Channels

A channel is an arrow connecting an argument port with another port.



$$\textbf{channel} \doteq \text{arrow} \sqcap \exists_{=2}\ \text{touching} \sqcap \forall\, \text{touching} \,.\, \text{port} \sqcap \exists_{=1}\ \text{touching} \,.\, \text{argument\_port}$$

3.5.3 Other Semantic Issues

The formal semantics given in the previous sections is mostly dealing with PJ's language elements. For the sake brevity, we left out many conditions specifying more complex semantic issues. In the following we roughly sketch out important notions concerned with these issues:

- **Connectivity**: With the exception of argument ports of agents or rules, every port has to be linked to another port. This feature is expressed by GenEd's higher-level relations (e.g. linked_with).
- **Reachability**: List elements (i.e. their reference ports) that are outside of rules have to be reachable from argument ports of rules via a (possibly empty) chain of other list elements. Of course, connectivity is a necessary condition for reachability.
- **Askers**: Channels, links, and data terms which are reachable parts of preconditions of rules are classified as askers.

- **Tellers**: Channels, links, and data terms which are reachable parts of rule bodies are classified as tellers.
- **Rules**: Ports of rules are either unused or linked to reference ports of askers or tellers.
- **Agent calls**: The number of argument ports of agent calls and their denoted agents have to be equal.

The complete set of specifications is sufficient to describe semantics of static PJ programs. This property is verified through the analysis of example programs resulting in a semantic network which can be used to create executable Textual Janus programs. Every Pictorial Janus program can be translated into an equivalent textual representation (Textual Janus) based on flat guarded horn clauses.

3.6 Related Work

There exist many approaches to specifying syntax (and to some degree semantics) of visual languages. Mostly, these are based on extensions of string grammar formalisms. A complete and recent overview is out of the scope of this chapter. However, we like to mention a few approaches: generalizations of attributed grammars (e.g. picture layout grammars [Golin, 1991]), positional grammars (e.g. [Costagliola et al., 1991]), and graph grammars (e.g. [Göttler, 1989; Najork and Kaplan, 1993; Rekers and Schürr, 1995]). Other approaches closely related to this one use (constraint) logic or relational formalisms (e.g. [Crimi et al., 1991; Helm and Marriott, 1991; Meyer, 1992; Wittenburg et al., 1991; Wittenburg, 1993; Marriott, 1994]) to represent spatial relationships. Wittenburg [Wittenburg, 1993] reports that some grammar approaches have limitations such as no arbitrary ordering of input is supported, only special relations are allowed, connected graphs are required, no bottom-up parsing is provided, no ambiguous grammars, etc. These limitations are sometimes unacceptable for particular application domains. We refer to [Marriott et al., 1998] for an extensive review of related work.

Helm and Marriott [Helm and Marriott, 1991] developed a declarative specification and semantics for VLs. It is based on definite clause logic and implemented with the help of constraint logic programming. Marriott's recent approach is based on these ideas but utilizes constraint multiset grammars [Marriott, 1994]. This is further explored in [Marriott and Meyer, 1997; Marriott and Meyer, 1998a] where a classification of visual languages by grammar hierarchies is presented on the basis of *copy-restricted* constraint multiset grammars. The decidability versus expressivity trade-off is used to shape the hierarchy in analogy to the Chomsky hierarchy in formal language theory. An advantage of our approach is the taxonomic hierarchy of concept definitions and the capabilities to reason about these specifications and their subsumption relationships.

Cohn and Gooday [Cohn and Gooday, 1994; Gooday and Cohn, 1996] applied the 'Region-Connection-Calculus' (RCC theory) to the VL domain and developed formal static and

procedural semantics for Pictorial Janus. However, their specifications use the first-order theory of RCC that is known to be undecidable (see [Cohn, 1997]). As far as we know, they do not support the graphical construction (e.g. editing and parsing) of diagrammatic representations or mechanical verification processes (e.g. consistency checking of specifications). Of course, due to the undecidability of RCC's first order theory a decision procedure for consistency checking cannot exist.

Citrin et al. [Citrin et al., 1994] also present work on formal semantics of completely visual languages. They developed formal operational semantics for control in the object-oriented language VIPR but their specification framework is not very formal and appeals to intuition.

Another approach to reasoning with pictorial concepts is based on a different, type-theoretic framework [Wang and Lee, 1993a; Wang and Lee, 1993b; Wang et al., 1995]. An important distinction is that our theory is more expressive with respect to concept definitions. For instance, in [Wang and Lee, 1993a] the authors suggest to extend their type-theoretic approach by notions such as parameterization for construction of generic concepts and type dependency for describing pictures consisting of parts of other pictures. Our DL theory already handles the intended effects of parameterization and type dependency since its reasoning component automatically maintains a taxonomy of subsuming concept definitions which may share common subparts.

The logical status of (extended) Venn diagrams is analyzed by Shin [Shin, 1994]. She gives axioms for well-formed Venn diagrams and a semantics using first-order predicate logic. However, Shin's formal account is not based on a spatial logic and not supported by reasoning mechanisms comparable to DL systems.

The understanding of diagrams can be also considered as a subproblem of image interpretation and is related to similar approaches in this area. The first treatment in this area was the MAPSEE approach [Reiter and Mackworth, 1989]. It is based on specifications with full first-order predicate logic. Another approach for the logical reconstruction of image interpretation [Lange and Schröder, 1994; Schröder and Neumann, 1996; Schröder, 1998] uses DL theory as framework.

In comparison to other logic-based approaches, we argue that DL notation—featuring concept and role definitions with inheritance and with a possible extension to concrete domains—is much more suitable for human and even mechanical inspection. This is an important issue since theories about VLs are still designed by humans. Another principal advantage of our approach is the use of necessary and sufficient descriptions, i.e. *defined* concepts.

The framework introduced in this chapter is suitable for recognizing (parsing) visual notations as well as constructing examples from specifications (without addressing the layout problem). Parsing can even hypothesize unknown information about notation elements. This can be accomplished with the help of ABox reasoning and the underlying model-theoretic semantics. The ABox reasoner verifies a notation example by creating a corresponding model and can automatically prove whether this model is still satisfiable if further

assumptions about elements were made. A similar reasoning scheme is proposed in Section 8.3.1 with the help of default reasoning.

The approach presented in this chapter also supports multi-level reasoning and can thus avoid problems with a combinatorial explosion of alternatives in specifications. For instance, imagine the specification of a triangle based on unordered sets of points (representing lines). We can avoid this problem because reasoning can take place about connectedness of points (low-level reasoning) as well as undirected lines (higher-level reasoning).

3.7 Summary

We like to emphasize that our approach has no restrictions about the ordering of input and the type of allowed relations if we incorporate concrete domains (see Chapter 7 for a DL with concrete domains). We do not rely on special parsing techniques because our approach is purely declarative. We can even deal with ambiguous grammars since the DL realizer can compute every model satisfying the specifications. A problem with our approach could be the worst-case time complexity of the underlying classification algorithms. However, almost every logical or constraint approach with an interesting expressiveness has to deal with tractability and decidability. It is also important to note that complexity issues of DLs are very well understood and analyzed.

The description logic used in this chapter is called $\mathcal{ALCQ}$ [Hollunder and Baader, 1991]. It extends the basic logic $\mathcal{ALC}$ with qualified number restrictions. The study reported in this chapter deliberately neglected the consideration of role hierarchies and inverse roles due to unknown decidability results at that time the study was conducted. Recently, the decidability of the ABox consistency problem for a superlogic of $\mathcal{ALCQHI}$ was proven [Horrocks et al., 1999]. $\mathcal{ALCQHI}$ extends $\mathcal{ALCQ}$ with role hierarchies and inverse roles. With these new constructs it is possible to model the characteristics of spatial relations and relations for partonomies much more precisely.

Chapter 8 investigates how to integrate DL theory with spatial domains in analogy to the concrete domain approach. This is necessary since spatial relations cannot be adequately defined with the operators offered by the concrete domain DL $\mathcal{ALC}(\mathcal{D})$. In our current approach spatial relations are considered as uninterpreted (primitive) roles with respect to DL theory and we need an external geometric reasoner (built into GenEd) asserting spatial relationships. A visual sketch-based query language for geographical information systems that is based on these ideas is introduced in the next chapter.

Chapter 4

Querying GIS with Spatial Sketches

We present the design of the visual query system VISCO that offers a sketch-based query language for defining approximate spatial constellations of objects. VISCO smoothly integrates geometrical and topological querying with deductive spatial reasoning. It is based on a strong physical metaphor visualizing semantics of query elements. Approximate queries rely on combined topological and geometrical constraints enhanced with relaxations and "don't cares."

4.1 Motivation and Introduction

The need to develop new interface paradigms for interacting with spatial information systems or databases, especially geographical information systems (GIS), has already been noted elsewhere [Egenhofer, 1992; Egenhofer, 1996]. With respect to HCI an interface should be convenient, easy-to-use, and actively supporting the user. A strong metaphor can motivate users and guide them through the interaction with a system. In response to these considerations the visual query system VISCO (Vivid Spatial Constellations) is presented. It provides a sketch-based query language for defining approximate spatial constellations of objects. VISCO is designed with the goal to reduce the burden of the user's intuition about VISCO's language elements with the help of such a metaphor. Geometric VISCO objects are associated with a metaphorical naive physics semantics that is intended to guide the user's interpretation of spatial aspects in a visual representation. The query language elements are visualized as rubber bands, (cross)beams, swivel joints, nails, marbles, etc. The meaning of the language elements is immediately graspable from the physical properties of their visualizations, e.g. a rubber band may be stretched, shrunk and wrapped around in contrast to a (rigid) beam, a marble can roll around and change its position in contrast to a nail (see Figure 4.3). The physical properties of geometrical objects add extra semantics to VISCO's language elements. Therefore, the meaning of the aspect "position" of a visualized point object can be immediately discovered by a user because of the attached "naive physics" semantics.

VISCO offers several novel features that correspond to issues mentioned in a survey on visual

query systems for databases. Catarci et al. [Catarci et al., 1997] conclude this excellent survey with a list of most significant issues for the design of next generation visual query systems. VISCO's features incorporate solutions for several of these issues.

- Visualization is an essential part of VISCO and illustrates possible variations in user sketches.
- VISCO deals with spatial data types such as points, segments, polylines, polygons and their possible spatial relationships.
- VISCO offers powerful tools for approximate questioning that may be used for formulating queries about approximate spatial constellations of database objects.

In contrast to other relevant work [Egenhofer, 1996] which focuses on topological descriptions we adopt a bottom-up approach and parse the sketches and their geometry as drawn by the user. VISCO takes the geometry of query sketches seriously but supports the annotation of meta information which can be used to specify almost pure topological queries, i.e. the query language can express geometric as well as topological constraints. The user may add meta information to a sketched query. This meta information specifies relaxations, additional constraints or "don't cares" that define the interpretation of the query. The visibility of user-defined relaxations and "don't cares" is a major advantage of our approach. In our opinion this explicit meta information (which has to be supplied by the user) is important since drawings are always in a sense "overspecified" and their (relaxed) interpretation strongly depends on the application domain.

4.2 VISCO: Visual Spatial Constellations

The interpretation of a user's spatial query is a critical part for any spatial query system. The user's intuition about the interpretation should match with the system's implemented algorithms. For instance, the concept of a right angle has a strong significance in a CAD system but must be relaxed in a GIS system. This shows that the "correct" interpretation depends on the actual application domain. Thus, VISCO offers tools that specify meta information resembling the user's idea of the interpretation. This approach demands more skill from the user but makes the intended interpretation explicit. This is the reason why VISCO's user interface (that is derived from GenEd, see the previous chapter) offers active support through visualizations.

4.2.1 The Visual Query Language

In the following, we assume a topologically structured vector representation of the data of interest. Data models like the one assumed here can be found in advanced vector-based GIS (in contrast to raster-based GIS). The data in vector-based systems usually consists of nodes or vertices (points), edges (lines) and faces (simple polygons). Additionally,

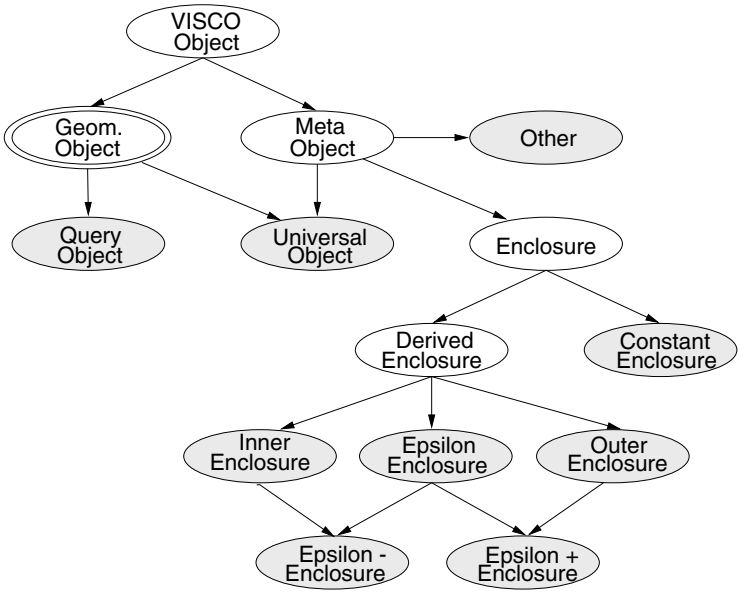


Figure 4.1: Query language elements supported by VISCO (non-shaded nodes represent auxiliary concepts; a refinement for the node "Geom. Object" is given in Figure 4.2).

polylines and arbitrary aggregates of these objects can be found. The "direct component of" relationship between these objects forms a DAG (Directed Acyclic Graph). In our case, the DAG has a maximum depth of 4 because aggregates may contain polygons (but never other aggregates) which are built from lines, and lines contain their end points. Together, object classes and the operations on them form the logical data model of the spatial information system. Like SQL, which is only suitable for the relational data model, VISCO's query language is only suitable for (topologically structured) vector-based spatial information systems.

The following elements are the building blocks of VISCO's query language (see Figure 4.1-4.3).

- **VISCO Objects:** A *VISCO object* is any element of the visual language VISCO.
- **Geometric Object:** *Geometric objects* are points, lines, simple (non self-intersecting) polylines, simple polygons and aggregates (see Figure 4.2). We distinguish two types of geometric objects: *query* objects and *universal* objects (see below).
- **Query Objects:** A *query object* is a geometric VISCO object that matches geometric objects in the spatial database. We can also say, that a query object *represents* a database object. Geometric database and geometric VISCO objects must be identically structured or very similar because VISCO's query execution is based on *graph matching* (see the next chapter).
- **Universal Objects:** Unlike geometric *query objects*, which must directly match objects in the spatial database, a *universal* or *auxiliary object* represents an object in

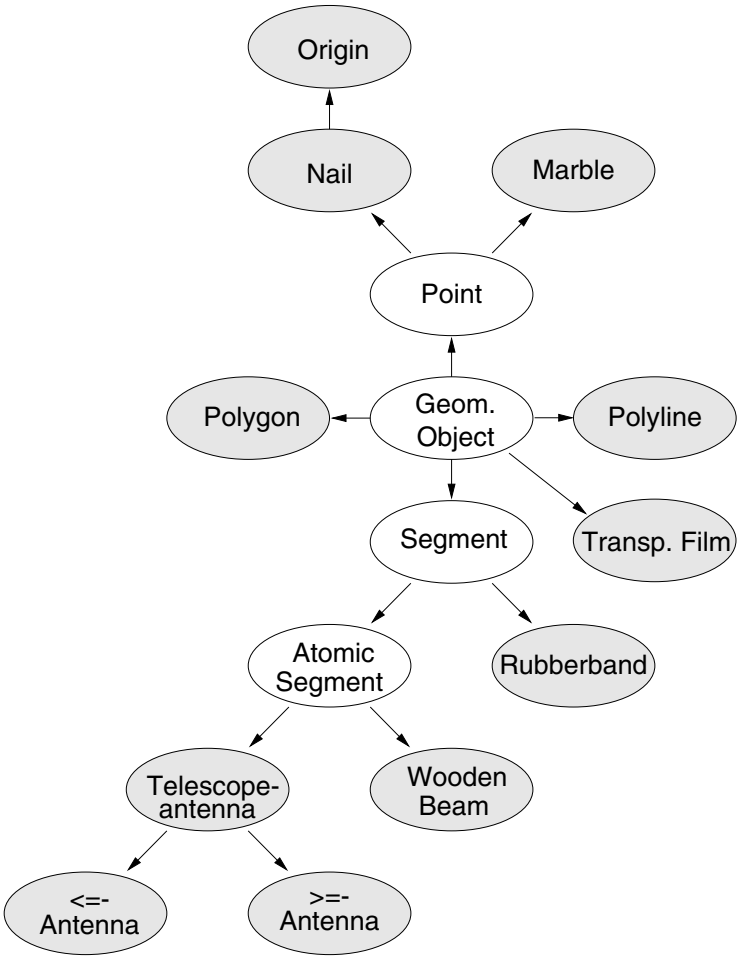


Figure 4.2: Geometric objects of VISCO.

the universe of all well-formed geometric objects. Universal objects are primarily used for expressing additional constraints on query objects, and therefore are considered as *meta objects*. It is required that universal objects can be instantiated by other objects (e.g., as an operator result or through its component objects – a universal segment could be instantiated by its end points provided they are query objects).

- **Meta Objects:** A *meta object* is a VISCO object that visualizes some additional conditions (constraints) on other VISCO objects and therefore makes statements *about* these other objects and their interpretation. Special meta objects are *enclosures*, other meta objects visualize other possible constraints (in form of arrows, text objects etc).

- **Enclosures:** An *enclosure* is a meta object representing a (connected) subset of $\mathbb{R}^2$. We distinguish *constant* (or *sketched*), *interior* and *exterior* (for polygons), and ε-enclosures (see also Figure 4.9). Enclosures may be *translucent* or *opaque*.

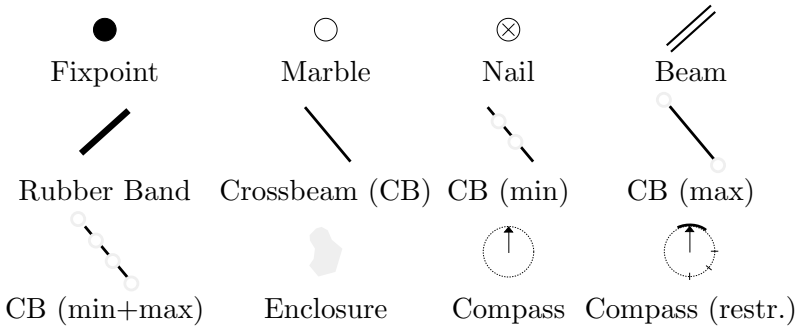


Figure 4.3: Basic language elements of VISCO.

Opaque enclosures are used to (partially) occlude other objects and to (partially) disregard their existence. With the help of opaque enclosures one can express spatial "don't cares" by ignoring spatial relationships with occluded objects. In order to compensate for this visual incompleteness, VISCO's language has to be integrated into a supporting environment because it might be impossible to reconstruct the semantics of a query only from the final drawing.

- **Derived Objects:** *Derived objects* comprise interior, exterior, and ε-enclosure, but also derived center points, calculated intersection points, etc.

In the following some of the geometric objects of VISCO (see also Figure 4.3) are described in detail.

Transparency Films

Keeping in mind the naive physics metaphor mentioned above, the applicability of VISCO's language elements is easily explained. The basic building block is a *transparency film* (of an overhead projector). Every transparency has its own local cartesian coordinate system and a rectangular shape. Users can interactively start to draw query language elements upon a transparency. A collection of drawn elements defines a (sub)constellation with relevant geometrical/topological relationships. The size of a transparency and the size and position of elements drawn on a transparency are taken seriously and do matter. Transparencies can be scaled, translated, rotated and stacked up like layers. Transparencies always have a fixpoint (with respect to transformations) which can be any nail (isolated or as vertex) on the transparency.

Figure 4.4 illustrates various examples. Figure 4.4a shows a simple unscalable transparency with its fixpoint in the center. The transparencies in Figure 4.4b-f may be scaled as follows: only vertically down (b), any direction (c), only proportionally (visualized by dashed guiding lines for the vertices) in any direction (d), only proportionally down with an uncentered fixpoint (e), only proportionally up (f). The transparencies in Figure 4.4g-j also constrain the upper and/or lower limit of scaling (visualized by dashed horizontal or vertical lines): proportionally (g) or arbitrary (h) with lower and upper limit, only

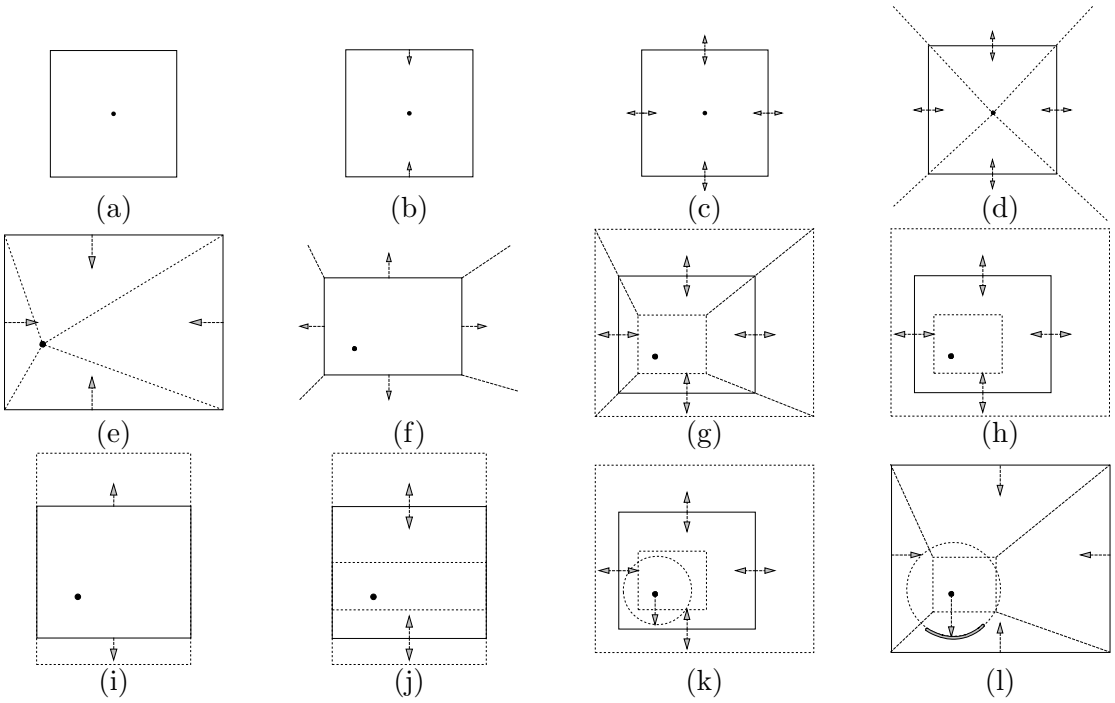


Figure 4.4: Various applications of transparency films.

vertically up with upper limit (i), only vertically with upper and lower limit (j). In Figure 4.4k-l we additionally allow rotation of the transparencies around their fixpoint. This is specified by the compass disk with the fixpoint as center and an arrow as hand. The compass in Figure 4.4k allows free rotation around the fixpoint, while that in Figure 4.4l constrains the possible rotation to an angle interval (visualized as bold arc). In general, a compass may constrain rotation to angle intervals and discrete angle values (see Figure 4.3). A compass allowing only one discrete angle may be abbreviated as a single arrow (its hand) in order to avoid visual clutter.

Enclosures and Points

Let us imagine, an enclosure is sketched on a transparency. An enclosure is represented as a simple polygon (with optional holes) whose boundary has to be a closed and not self-intersecting polyline. An *enclosure* is a meta object adding to its denoted area the semantics that all enclosed objects have to stay inside of this (fenced) area. Also, it specifies that the position of some of its enclosed points have to be relaxed: a point drawn inside an enclosure is by default a *marble* that can move around but has to stay inside. However, a *nail* cannot change its position (even inside of an enclosure). Enclosures can be *translucent* or *opaque* and are displayed with a gray texture. The associated semantics is described below (see Section 4.2.2). It is also possible to generate so-called ε*-enclosures* which are computed by operators (see Figure 4.9 for examples).

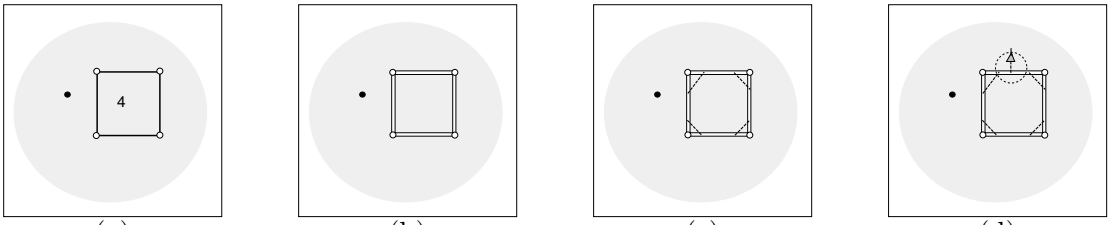


Figure 4.5: Various quadrilaterals.

Line Segments, Polylines and Polygons

Another basic language element is an (undirected) line *segment* with its two end points. Each point can be either a marble or a nail. It should be noted that all query objects except marbles and nails are compound objects, which have component objects. These components are itself "first class" objects. Segments or edges (as components of polylines) act either as *beams* of fixed length or stretchable/shrinkable *(atomic) rubber bands*. A beam matches a line segment of fixed length (as specified) in the database. A rubber band represents a topological structure that matches an arbitrary polyline in the database. However, an *atomic rubber band* is defined as indivisible and therefore matches a single database line segment. In our physical metaphor it resembles somehow a *telescope antenna.* There are three types available: the $\leq$ ($\geq$) atomic rubber band represents a segment with a given maximal (minimal) length, and the "don't care" atomic $\star$ rubber band does not enforce any constraints on the length of segment. Vertices (points) connect edges and also play the role of *swivel joints* that are either marbles or nails. Polylines may be closed but not self-intersecting. Regions are defined as simple polygons with polylines as boundaries.

Enclosures affect directly the position of points and indirectly the position and/or shape of polylines (and thus polygons). Form variability of polylines and polygons is achieved by position (length) variability of their vertices (edges). Furthermore, number constraints can be associated with some query objects. An *at most constraint* for a rubber band, polygon or polyline specifies the maximal number of line segments of the database object that is matched against this query object. The number of segments of a query polygon or polyline always forms an implicit at least constraint. An atomic $\star$ rubber band is therefore equivalent to a rubber band with an 'at most one' constraint.

For instance, Figure 4.5a shows a transparency containing the specification of an arbitrary quadrilateral whose edges are rubber bands and whose vertices are marbles that can float inside of the enclosure (gray circle). If we discarded the *at most four* constraint for the polygon, this query would retrieve arbitrary polygons (with at least four line segments). Alternatively we could have defined the quadrilateral without the at-most constraint by using *atomic* $\star$ rubber bands instead of rubber bands. Figure 4.5b-d demonstrates various other definitions: arbitrary quadrilateral with edges of fixed length (b), floating square of fixed shape (c), floating square of fixed shape and upright orientation (d).

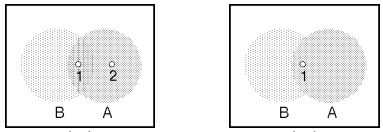


Figure 4.6: Translucent and opaque enclosures.

The examples in Figure 4.5c-d use another meta object: a *crossbeam* constrains the angle between two connected edges. A simple crossbeam freezes the angle as drawn. This can be relaxed with minimal and maximal angles indicated by bullets lying on the crossbeam (see Figure 4.3).

4.2.2 Relationships between Objects

At first it has to be noted that the user's query is constructed step-by-step using VISCO's interactive environment. VISCO's language is not designed to be visually complete because the construction history of the query affects its semantics. Parsing a VISCO query is a progressive process as the construction continues. However, at each construction step WYSIWYG is taken seriously. Whenever a new query object is introduced, various relationships to other (partially) *visible* objects are calculated. For instance, the 'contains' relationship (or constraint) is enforced between enclosures and their enclosed query objects. Components of complex query objects are considered as individual "first class" objects (e.g. the segments of a polygon). So, any discussion of query objects applies to component objects as well.

We already mentioned that enclosures can be translucent or opaque. An opaque, overlapping enclosure stacked in front can (partially) hide some underlying enclosures and also objects already drawn. If the topmost enclosure is translucent it will not hide the underlying elements. Any visible element stacked below can still be addressed since it is not hidden (WYSIWYG). Thus, we can enforce an object to be inside various (underlying) enclosures, if they are not hidden. This is an explicit 'and' constraint holding for all visible 'inside' relationships of this object. See Figure 4.6a-b: in case a), marble 1 has to be inside A and B, marble 2 has to be inside A. In case b), marble 1 only has to be inside A.

If an object Y is (partially or fully, see below) visible (with respect to occluding enclosures) at the time of creation of a query object X, we enforce some additional high-level spatial constraints between X and Y. These enforced spatial relations strongly affect the semantics of a query. The following relations (and their unlisted inverses) are recognized by VISCO:

- **Enclosure × query object:** *contains*
- **Point × segment:** either *intersects* (if the point lies on the line segment) or *disjoint*
- **Segment × segment:** either *intersects* (this includes any constellation where two segments have at least one common intersection point) or *disjoint*.

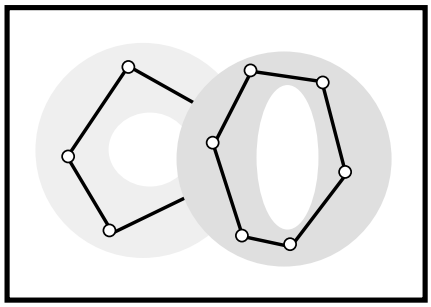


Figure 4.7: Unknown spatial relations between two objects in different enclosures (disjoint or touching or overlapping?).

There are no point × point relations, because the user interface does not allow the definition of two points at the same position. Two points are therefore always disjoint. A common vertex of e.g. two line segments of a polygon is assumed to be only one unique point object (and not two objects that are equal). This also implies that there exist no congruent query objects.

The *intersects* and *disjoint* relations are computed between points and lines, and lines and lines regardless whether the point or line objects are components (because of the "first classness" of component objects). It can be easily shown –by taking into account all component point-line relations– that the relations *intersects* and *disjoint* are expressive enough to define, for example, all interesting line-line relations (e.g. disjoint, crosses, touches, meets, overlaps, contains, inside, covers, covered_by).

However, no relation between a compound object and any of its components will be introduced. As an example, if we have two individual line segments that are crossing each other and completely visible with their two endpoints, the system will calculate 10 relation constraints (2 line × line tuples, 4 point × line tuples, 4 line × point tuples).

We also support relaxations of spatial relations. Relaxations can be achieved by abstracting from the identity of component objects and regarding them as a proxy of their compound object. As an example, the description of a query where a segment crosses a polygon might include the constraint $crosses(segment_1, poly_2\_segment_5)$. If we abstract from the unique identity of $poly_2\_segment_5$, we can rewrite this constraint to $crosses(segment_1, poly_2)$ and could get more matches. VISCO supports these relaxations through a graphical annotation method which is not discussed here in detail.

Generally speaking, WYSIWYG determines whether a *disjoint* or *intersects* relation is enforced. The visual information present at creation time has to unambiguously imply the relation constraint. For instance, if an object A is partially hidden by an enclosure containing another object B, one cannot visually decide at the time of creation of object B whether A and B are disjoint or not, and the visually present information does not accidentally imply ambiguous relationships (see Figure 4.7).

The concept of an enclosure can also be applied to stacked transparencies and their fix-

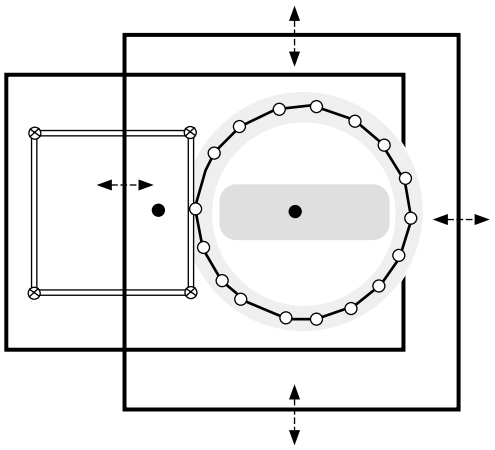


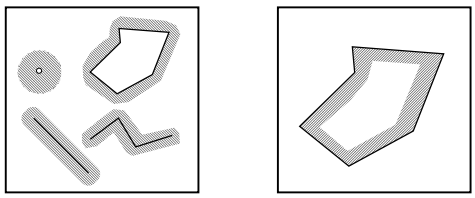
Figure 4.8: Scalable circle touching a rectangle.

points. The fixpoint of a transparency A can play the role of a marble enclosed on a lower-level transparency B. This implies that A's fixpoint can float inside of the enclosure on B (and thus the flotation of the whole transparency A is constrained). Note that the whole coordinate system of A is affected by B's actual transformation. Enclosures for fixpoints of other transparencies must be opaque. We restrict the visibility of fixpoints to at most one transparency on a lower-level, i.e. a hierarchy of stacked transparencies may result. The fixpoint of the top-level transparency (in the hierarchy) is implicitly constrained to be inside a "whole world area". See Figure 4.8 for an example: this query defines a rectangle of fixed size and shape that touches a scalable circle at the right.

4.2.3 Operators and Computed (Derived) Objects

VISCO's interface supports the application of operators to objects. When a new object is created, its component objects will be introduced as first-class objects. We call this "top-down" creation of objects. Conversely, a new object can be aggregated (what we call "bottom-up" creation) by choosing existing query objects as components of the new object (e.g. choose two existing nails as end points of a new rubber band). Operator-created objects are also first-class (*derived*) objects. For instance, the intersection point of two crossing line segments can be computed. Type and properties of derived objects are automatically computed but can always manually changed by the user through subsequent operator applications. Note that derived objects are visualized in a different color in order to distinguish them from meta and other query objects. Constraints can become redundant during the incremental construction process, e.g. a compass allowing free rotation for a beam that is fixed between two nails is superfluous and will be removed from the query.

For sake of brevity we will not discuss all operators of VISCO in detail. The most important ones are the following (of course, additional operators for changing object positions, parameters etc. are required). Note that we consider nails and marbles as points, and rubber bands, atomic rubber bands, and beams as line segments.



ε^+-enclosures ε^--enclosure

Figure 4.9: Examples of ε-enclosures.

- **NIL**: create transparency
- **Transparency:** create query object, create enclosure
- **Enclosure:** toggle enclosure's translucency, negate enclosure (precondition: enclosure has no holes), create at most constraint (specifying the maximal number of found database objects inside this enclosure)
- **Query object:** attach DL concept descriptor (e.g., "road") to object
- **Point:** create (derived) ε-enclosure with radius r
- **Nail:** declare as transparency fixpoint, declare as marble (precondition: nail inside an enclosure)
- **Marble:** declare as nail
- **Fixpoint $\times$ 'opaque enclosure of underlying transparency containing the fixpoint':** restrict possible translations of the fixpoint's transparency to the enclosure of the underlying transparency.
- **Fixpoint $\times$ 'underlying transparency that contains the fixpoint':** see above, with an enclosure degenerated to a point.
- **Point $\times$ point:** aggregate line segment from points
- **List of line segments:** aggregate polygon, aggregate polyline
- **Beam, atomic rubber band:** create compass
- **Polygon, polyline:** create compass, then inherit the compass to all beams and atomic rubber band segments
- **Line segment, polyline:** create ε-enclosure with radius r, create midpoint
- **Point, line segment, polyline:** declare object as a part of a searched object (e.g. a polyline as part of a polygon boundary)

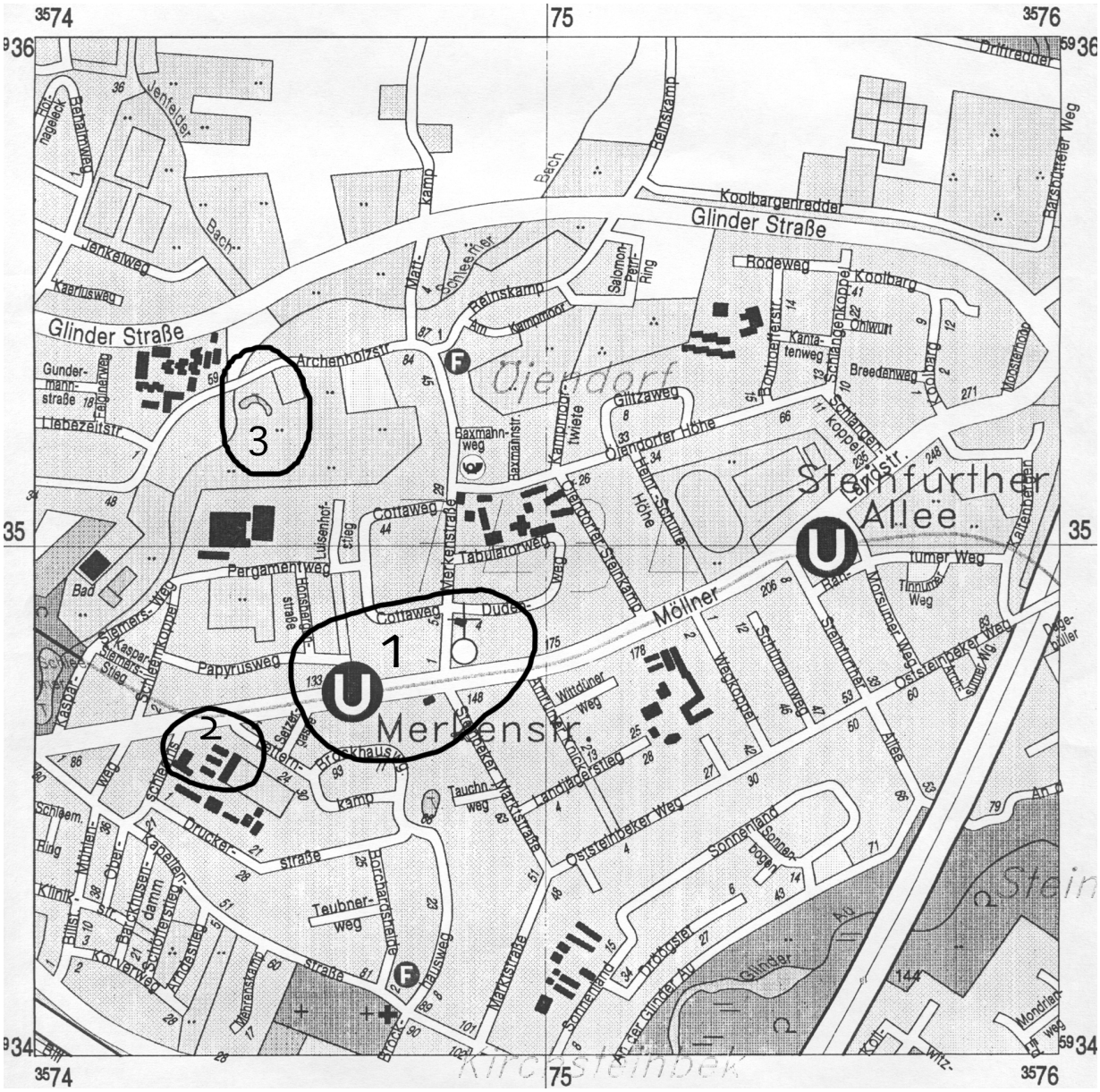


Figure 4.10: A subsection of the city of Hamburg. Intended query matches are marked and numbered.

- **Polygon:** create inner enclosure, create outer enclosure, create ε^+-, ε^--enclosure (see Figure 4.9, a radius r is always required), create midpoint
- **Beam:** declare as rubber band, declare as atomic rubber band
- **Atomic rubber band:** change type to $\leq$, $\geq$ or $\star$, declare as beam, declare as rubber band

- **Rubber band:** declare as beam, declare as atomic rubber band, create at most number constraint
- **(Beam ∨ atomic rubber band) × (Beam ∨ atomic rubber band):** create crossbeam between them
- **Line segment × line segment:** create intersection point (precondition: a *crosses* relation). The new point will be a marble (if inside any enclosure), or a nail otherwise.
- **Polygon or polyline with at least one non-atomic rubber band segment:** create at most number constraint

4.2.4 Extended Example: City Maps

In the following we present three extended examples with queries of increasing complexity. All examples are taken from a 'city map' domain. The queries describe constellations of objects represented in a city map. Figure 4.10 shows a raster image displaying a small subsection of the city of Hamburg, located in Northern Germany. The intended match for each query is marked and numbered.

There exists a corresponding vector version of this map that is represented in our spatial database. The data elements of the vector map consist of text, points, polylines, and polygons. All data elements have attributes describing their role in the city (e.g. lake, pond, street, church, building, etc). The attribute values correspond to DL concept descriptions organized in a predefined taxonomy. Data elements are represented as instances of concept descriptions that combine conceptual and relational (spatial) knowledge.

The first example describes a constellation where a subway station is in the vicinity of a church. The intended target of the query is displayed in Figure 4.10 (marked as number 1). A magnified selection is shown as the first element of Figure 4.11. The other elements show the three construction steps of the corresponding query.

1. Create a transparency with a fixed size of 300 × 300 meters. We assume that the subway station and the church are represented as points in our database.
2. Draw a nail on the transparency and attach the predefined concept 'subway station'. We declare the nail as the fixpoint of this transparency. The transparency itself is unrelated to any other transparency. This implies that the fixpoint may coincide with any point object in the database (i.e. on the map).
3. Generate with a predefined operator a circular ε-enclosure with a radius of 100 meters around the fixpoint. Afterwards, draw a point as a marble inside of the enclosure and attach the concept 'church' to the marble.

The second example describes a constellation where we search for three buildings aligned in parallel (the intended match for the query is marked as number 2 in Figure 4.10). The size of the buildings may vary individually. The first element of Figure 4.12 shows a zoom of intended match while the other elements illustrate the query construction process.

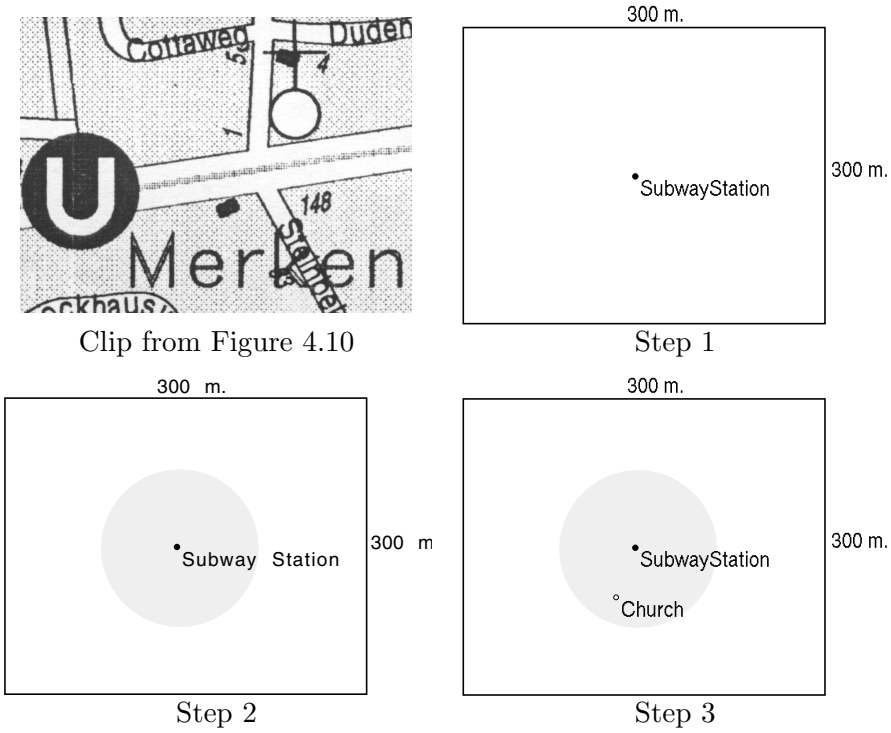


Figure 4.11: A subway station with a church in its vicinity.

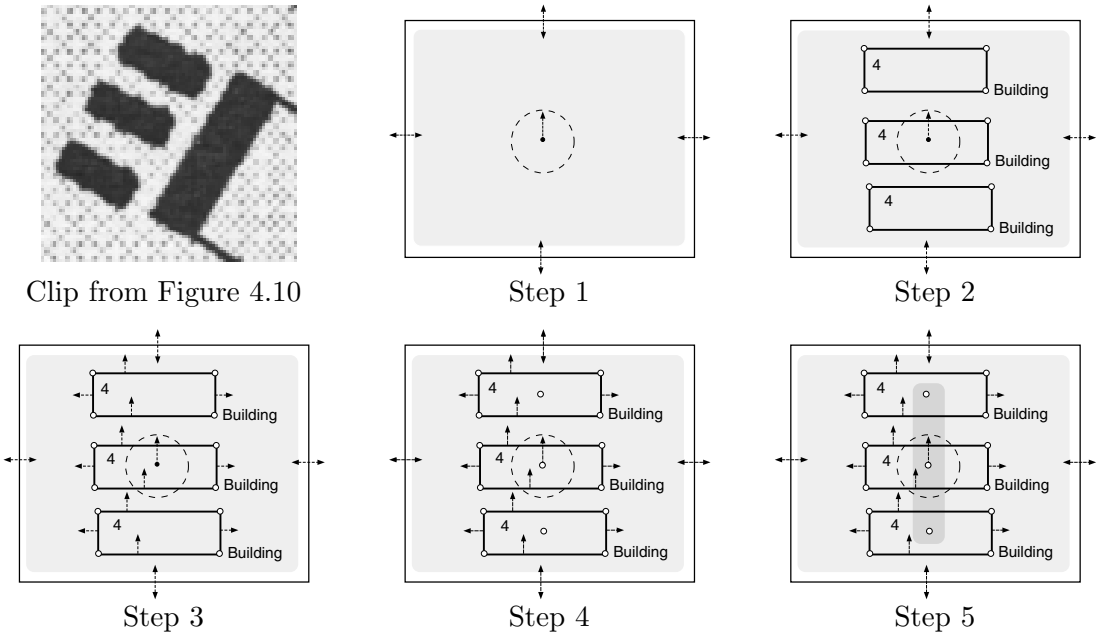


Figure 4.12: Three adjacent buildings aligned in parallel.

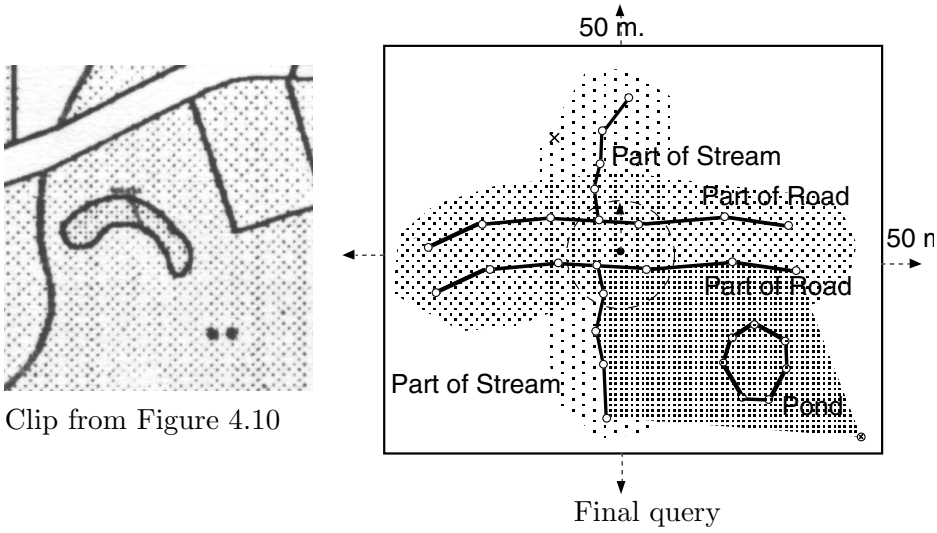


Figure 4.13: Constellation with brook, street, pond.

1. Create an arbitrarily scalable and rotatable transparency. Add a large rectangular enclosure to this transparency.

2. Create three rectangles. Their vertices are declared as marbles and their edges as rubber bands. Each rectangle has to match a database polygon with exactly four line segments. Please note that each rectangle defines a quadrilateral (see also Figure 4.5a). The quadrilaterals are enforced to stay disjoint from one another. We attach the concept 'building' to each quadrilateral.

3. Attach to each edge of the quadrilaterals an arrow constraining the edge's orientation to a single fixed angle (as visualized).

4. Apply to every quadrilateral the operator "create center point" resulting in three (derived or computed) marbles.

5. Sketch an enclosure inside of the large enclosure denoting admissible positions for the center points of the buildings. This area defines possible deviations for the alignment.

The third and last example (see Figure 4.13) shows the final query and its intended target (marked as number 3 in Figure 4.10). The construction steps are illustrated in Figure 4.14. The query defines a constellation where a brook crosses a street in underground and runs along a pond.

The elements of the query are represented as follows. The constellation is attached to a proportionally scalable and rotatable transparency specifying a rectangular area with a minimal size of 50×50 square meters. The brook and the street are contained by the transparency and represented as polylines. They have their own translucent enclosure

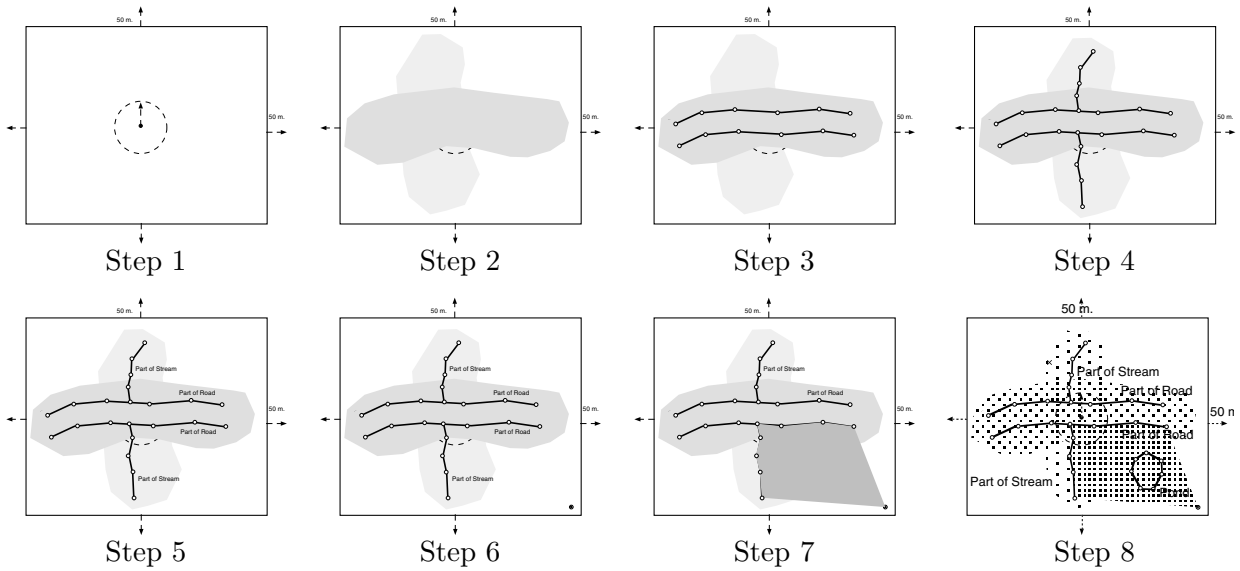


Figure 4.14: A brook crossing a street in underground and running along a pond.

overlapping each other. The pond's enclosure is attached to the lower right corner of the intersection of the enclosures of the brook and the street.

4.3 Related Work

VISCO can be classified as a visual query system for spatial information systems that uses sketched queries combined with deductive reasoning. A comprehensive survey of visual query systems for database systems handling conventional data can be found in [Catarci et al., 1997]. Other relevant work [Meyer, 1994; Egenhofer, 1996] reviews visual query systems for spatial information systems. A related approach that also uses spatial relations [Del Bimbo et al., 1994] deals with symbolic descriptions and retrieval in image databases. In the following we shortly review four approaches [Meyer, 1994; Calcinelli and Mainguenaud, 1994; Lee and Chin, 1995; Egenhofer, 1996] that come closest to the ideas and concepts behind VISCO. We especially focus our attention on the spatial properties of their query languages.

An iconic query language for GIS is presented in [Lee and Chin, 1995]. Icons represent geographic objects such as lakes, rivers, countries, etc. The icons are abstract and represent objects only symbolically. However, the dimensionality of objects (0D-2D) is directly represented by their icons. Possible positions and size of icons can be algebraically specified. Topological relationships are computed from the icons of a query. It is also possible to specify orientations or circular search areas. The system provides a "foreground" mode which is used to specify relevant relationships between objects. Other accidental relationships are interpreted as "don't cares". The user interface is very abstract with simple visualizations.

Geometrical aspects cannot be specified. Only a small set of relations without a formal model is allowed.

Cigales [Calcinelli and Mainguenaud, 1994] is a "query by visual example" system which also allows the user to generate a query. However, queries cannot be sketched but have to be created with the help of operators. For instance, two intersecting areas are created as the result of an intersect operator. Thus, the drawings are only visualizations created by the system. In contrast to other systems, Cigales derives the spatial layout of query elements from the application of predefined operators. The implied look-and-feel of Cigales' user interface appears to be quite tedious. Furthermore, the layout of queries can change dramatically after their reformulation and might confuse users. This may be caused by the automatic visualization process.

Sketch [Meyer, 1994] was the first language proposing a metaphor for drawing sketches on a blackboard. Sketch allows free-form elements as components of sketches but it strictly divides queries into propositional and spatial conditions. A sketch is parsed and translated to a formula in a corresponding logical calculus. The problem of "don't care" relations is solved by layers that can contain common objects. Topological relationships are only computed for elements of a layer. The language Sketch has formal semantics but the topological relationships have no mathematical foundation. Sketch does not support the integration of geometric properties into queries. It also requires that database objects are "pretyped", i.e. they cannot be recognized through their geometrical properties. For instance, one can think of a CAD drawing of a transistor that consists of a flat unordered and unstructured "spaghetti" collection of line segments.

Spatial-Query-by-Sketch [Egenhofer, 1996] is distinct to the previous approaches with respect to similarity matches. It uses conceptual neighborhoods of topological relations for relaxation of queries. For instance, a 'touches' relationship between two objects can be very similar to a 'disjoint' or 'intersects' relationship provided the (positive or negative) gap between the objects is below an appropriate threshold. Spatial-Query-by-Sketch allows multi-modal user input for specifying annotations or desired relaxations. It provides no facilities for specifying "don't cares" that apply to relationships between objects. We argue that metric information is existent in many domains and should not be discarded and then subsequently added.

Egenhofer states in his paper that "topology matters, metric refines". However, we argue that an important cognitive distinction for users is also the geometry (e.g. shape) of objects. For example, the distinction between a rectangle and a circle (although topologically equivalent) might be highly relevant for particular application domains.

4.4 Summary

The design of VISCO differs in several aspects from the four approaches mentioned above. It is expressive enough to define geometrical as well as almost pure topological queries or a combination of both. It yields high expressiveness by interpreting topological relations as

high-level qualitative constraints enriched with meta information. VISCO offers a powerful but still quite simple physical metaphor for defining queries as spatial constellations. It is possible to specify approximate or vague objects and constellations. VISCO provides a clear distinction between query and meta objects. The meta information (i.e. additional specifications guiding sketch interpretation) is directly visible to users. There exist no 'hidden' relaxations that might confuse the user's intuitions about query interpretation. The prototype implementation of VISCO is introduced in the next chapter.

Chapter 5

VISCO: Bringing Visual Spatial Querying to Reality

This chapter reports on the implementation of a subset of the spatial (sketch-based) query language VISCO (see also [Wessel, 1998] for further details). The design of VISCO's query language was presented in the previous chapter. The example domain, city maps of Hamburg, is revisited and used to describe VISCO's user interface. The user interface consists of three interconnected components: a graphical (syntax-directed) query editor and visual language compiler, a browser for inspecting the query results, and a map viewer for browsing the spatial database. We also briefly report on the process of compiling, optimizing, and executing VISCO's queries.

5.1 Introduction

We will mainly discuss the user interface which is composed of three main components: the *graphical query editor* offering "sketch"-based querying of spatial databases (e.g. containing digital vector maps of the city of Hamburg), a browser-like *query-result inspector*, and a powerful *map-inspection tool* for the spatial database. Another component of the system is the optimizing visual language query compiler. Our prototype successfully demonstrates the usefulness and feasibility of VISCO and its underlying language concepts. The current prototype is fully implemented as described in this chapter.

According to the point of view of many other authors in the field, we assume that the term *visual language* denotes a means of communicating with a *visual system* in a coherent and consistent way through *visual expressions* (e.g. see [Catarci et al., 1997]). In our opinion, a visual query language should not be considered in isolation, but in an integrated environment providing an easy-to-use visual query system, offering active support and feedback, strong metaphors etc. In fact, it often becomes obvious that the usefulness of a visual language heavily relies on an appropriate interface which therefore –at least for the user– *is* the language (see [Graf, 1990]).

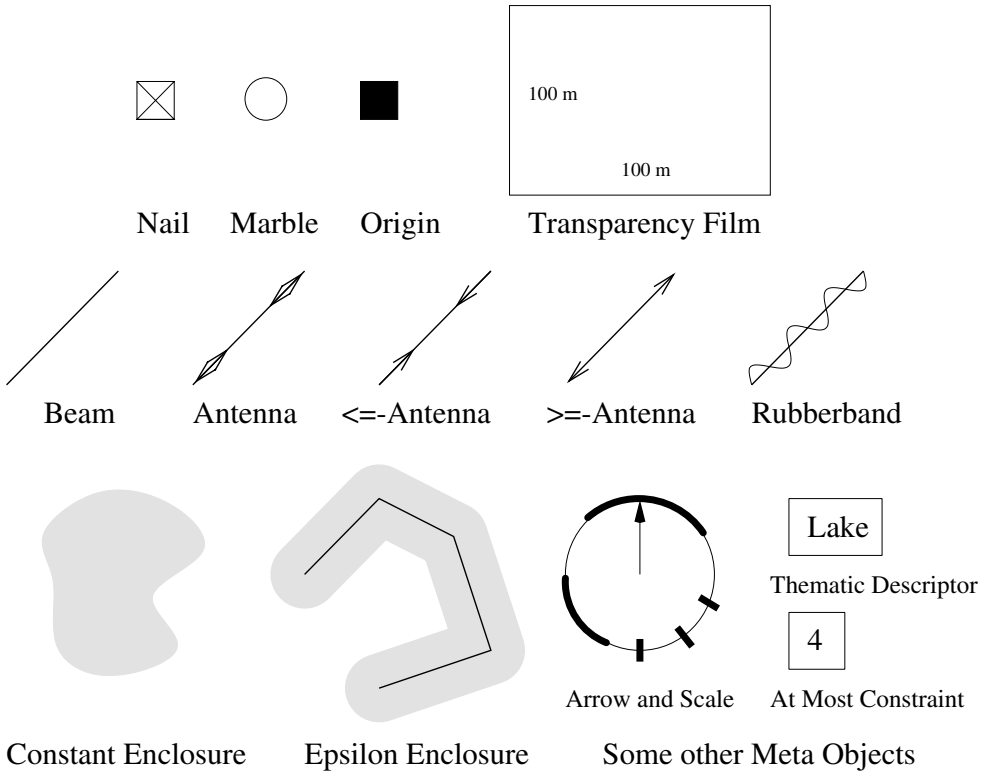


Figure 5.1: Visual appearance of various implemented VISCO objects.

The actual visual appearance of various VISCO objects in the implemented user interface is shown in Figure 5.1 (see Section 4.2.1 for a description of these elements and Figure 5.2 for an example query). For sake of efficiency and ease of implementation the appearance of some elements is changed compared to their first design reported in the previous chapter. Due to the high expressiveness of the full query language only a subset with slightly modified language elements has been implemented.

Constructing a VISCO query is a progressive process: at the time when a new object is created, various high level topological (spatial) constraints between the new object and already existing objects are established. Here, the notion of (partial) visibility becomes crucial. Also, each component object (e.g. a segment of a polygon) is considered as an individual object with its own identity – however, topological constraints involving components can be discarded if necessary, yielding more relaxed queries. The following topological relationships between a newly introduced object and any existing object that is not totally occluded by an enclosure are recognized by VISCO:

- **Disjoint** is established, if the other object is completely visible.
- **Intersects** is established, if the new object has at least one visible intersection point with the other object.

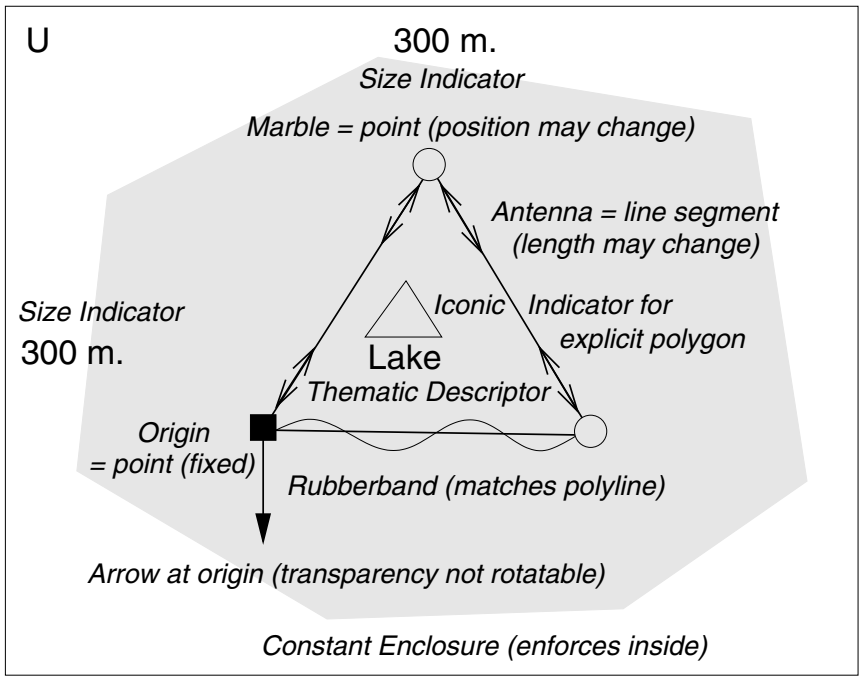


Figure 5.2: A simple VISCO query: "Search for a lake of (nearly) arbitrary form that is not bigger than 300×300 m." The elements used in the query are explained in the figure by annotations (in italic font).

- **Inside** is established, if the new object is completely visible inside of an enclosure.
- **Contains** is established between a new translucent enclosure and any other object that is completely contained within this new enclosure.

When editing a query it is important to distinguish explicit from implicit (emergent) polygons and polylines. Note that a polygon is by itself an implicit object. For instance, in the case of a triangle we only have visual indicators for the sides of the triangle, but not for the triangle *itself*. As a solution to this problem, we introduce for each explicit polygon (such as the triangle in Figure 5.2) and polyline an iconic sign in form of a "picture of the picture" (reduced to a small size).

5.2 The VISCO Prototype

The logical architecture of the VISCO prototype is shown in Figure 5.3 and could be termed as a "repository model" (see [Sommerville, 1995]). We think of VISCO as a component in the application layer of a spatial database working on an external data model (view) provided especially for VISCO. Of course, meta data plays a crucial role for answering questions like

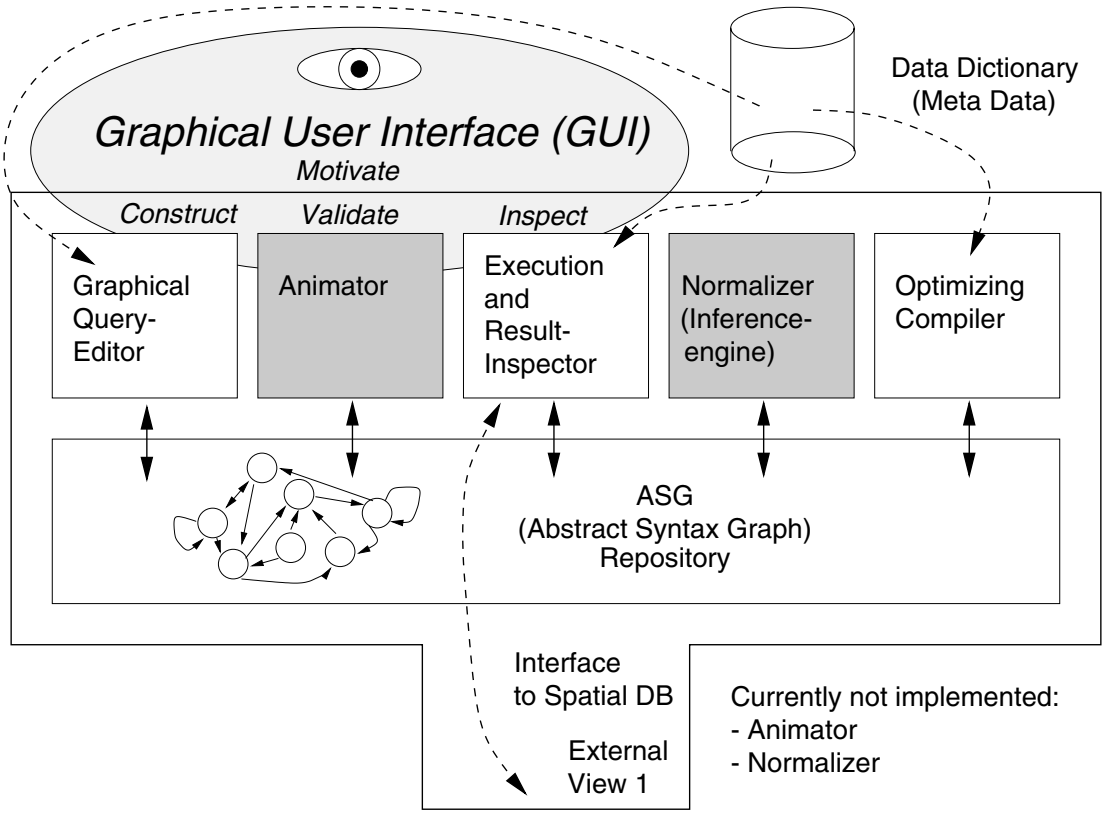


Figure 5.3: Logical architecture of VISCO.

"What types of objects are present in the database?", etc. The GUI is given through the following two components:

- a (syntax directed) query editor (a specialized graphical editor), and
- an execution control and query result inspector (including the map viewer, see below).

Another component is the optimizing compiler (see Section 5.3). The animator (intended to visualize parts of the extension of the current query through animations) and the query normalizer have not been implemented. These five components work on a common abstract syntactic representation of the current query, which is maintained and managed by the abstract syntax graph repository module (ASG module). The ASG is given in form of a directed multi-hypergraph. The syntax directed query editor enforces the construction of correct ASGs in this repository. Some meta data must also be reflected at the user interface — for instance, it does not make sense to allow the user to query a CAD-database for buildings located in the vicinity of lakes.

We already emphasized the importance of well-designed and easy-to-use GUIs for visual languages. In our case, the graphical query editor shown in Figure 5.4 is one of the most

Figure 5.4: The Graphical Query Editor of VISCO: the main window (left) and the "Buttons" window (right).

important parts of the VISCO GUI. The editor's user interface is composed of two main windows, labeled "VISCO" and "VISCO Buttons" (handled by two communicating concurrent processes). The "VISCO" window is the *working space*, allowing users to interactively construct (execute, load, etc.) graphical queries. It consists of four main areas: the biggest one is the "VISCO Query" pane showing the actual graphical query (which has been already presented in Figure 5.2), the "VISCO Infos" pane providing helpful information and explanations, the command line for entering textual commands, and the "VISCO Control" pane displaying the editable query construction history, which is automatically maintained by the system during the construction of a graphical query (see below).

The *current state* of the query editor is maintained and completely visualized by the "VISCO Buttons" window. For instance, by selecting the button for "rubberband" in the "VISCO Objects" pane, the next new line segment will be created as a rubberband. The buttons are named as follows (from top to bottom and left to right): transparency, constant enclosure, beam, $\leq$-antenna, origin, $\geq$-antenna, nail, antenna, polyline (chain), marble, rubberband, polygon). The (slightly set apart) block of 9 buttons labeled "DB DB-C U" determines whether the next new geometric VISCO object will be a *geometric*

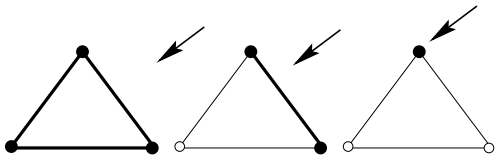


Figure 5.5: Vague gestures in VISCO.

universal object ("U"), or a *geometric query object* ("DB-C" or "DB" – the difference between these two can be ignored here). The two blocks of buttons in this pane must be considered columnwise; please note that the object icons of the buttons are labeled according to the actual selection (here, "DB-C", "DB-C", "DB").

Some other graphical presentation options can be selected with buttons in the "VISCO Options" pane, as well as an operator from an *iconic operator library* that pops up by pushing the currently active operator icon shown in the "VISCO Operators" pane. Then, the selected operator can be applied to a VISCO object (or a pair of VISCO objects in the case of a binary operator) visible in the "VISCO Query" area (prefix operator mode). Most commands can either be entered textually via the command line or chosen from the "Operators" menu in the menu line, as well as directly activated on an object by a key sequence. However, most of the construction (e.g. polygon "drawing" etc.) is done directly without the need to refer to these operators.

In summary, the query editor offers the following powerful features:

Parallel maintenance of an *editable construction history*: By selecting a construction step in the history, the main display is updated to reflect the state of the query construction at the time of this step. Entries can be deleted (but of course, the "delete" operation can also be applied directly to their graphical counter parts). The indentation of entries shown in the "VISCO Control" pane reflects the graph structure of the objects (e.g., lines having end points as parts).

Facilities that support the users's process of query understanding and formulation: For instance, the recognized topological relationships (see above) for a newly introduced object can be visualized by coloring the already visible objects, denoting the enforced corresponding spatial constraint (blue means disjoint, red means intersects, green means inside/contains). By changing the *focus* or *current object* (this can be done in the "VISCO Control" pane), every enforced spatial constraint for every object can be inspected in a stepwise manner.

Handling of and interaction with complex objects: For instance, in the case of a polygon, the polygon as a whole as well as its segments and their end points must be referenced and manipulated by mouse gestures through the user. VISCO offers two mechanisms for achieving this goal: *first,* the notion of a *focus* (or *current*) *object* (which can be selected by pointing at the graphical object or its counterpart in the construction history); and *second* the concept of *vague mouse gestures* (see Figure 5.5). The current selection (displayed in bold) depends on the distance between the mouse pointer and the

possibly targeted objects and their size.

"Top Down" and "Bottom Up" creation of complex objects: For instance, in order to create a polygon, a user must be able to select already present segments which afterwards become components of the freshly built polygon (what we call bottom up creation or aggregation), as well as to create some completely new segments and their end points (what we call top down creation). If a new object happens to be created, its type (e.g., rubberband, beam, antenna, etc.) and other attributes (should a new enclosure be opaque or translucent?) will be determined by the current state or mode of the query editor, which is maintained and visualized in the "VISCO Buttons" window. The editor's state can be changed concurrently *while* performing a complex operation (e.g. *while* creating a polygon, one segment could be defined as a rubberband between marbles, but the next segment could be a beam between two nails).

Handling of and interaction with emergent objects: For instance, there has to be a way to materialize the emergent rectangle formed by two overlapping rectangular polygons or the intersection point formed by two intersecting lines. In this example, the emergent rectangle can easily be made explicit by a twofold application of the operator "Create intersection point" and an aggregation of the four points to create a new polygon. This is also an example for the use of *derived or computed objects*: after repositioning one of the intersecting lines, an automatic reconstruction of the scene must follow.

Other features of VISCO, such as unrestricted multi-level "Undo" and "Redo" (in our query editor, even a "Load" can be undone), context-sensitive help facilities etc. are considered as *obligatory* nowadays.

Figure 5.6 shows the query execution and result inspection component, another important part of the VISCO GUI. Here, the result of the query displayed in Figure 5.4 is shown. Each tile represents a match (in this case, a lake). Another pane shows the Lisp code generated by the compiler for performing the search. Because components of polygons etc. are considered as individual objects, also permutations of "one and the same constellation" appear. Single tiles can be selected and further inspected, deleted, etc. Once a tile is selected, it can also be inspected more thoroughly with the advanced map inspection tool "Map Viewer" which is shown in Figure 5.7: here, also the neighborhood of a match can be inspected, neighborhood objects can be queried for their type by selecting them, their structure, etc. The map viewer also supports the generation of *layers* by selecting individual themes (each theme can be switched "on" or "off" in the scrollable list at the left side).

5.3 Representing and Compiling Queries

The ASG repository maintains an abstract syntactic representation of the current query in form of a directed multi-hypergraph. The nodes represent objects (e.g. marbles, rubberbands, etc.) with their properties (e.g. an object is a "lake"), the simple edges denote spatial relations and other constraints (e.g. direct component of). Hyperedges represent

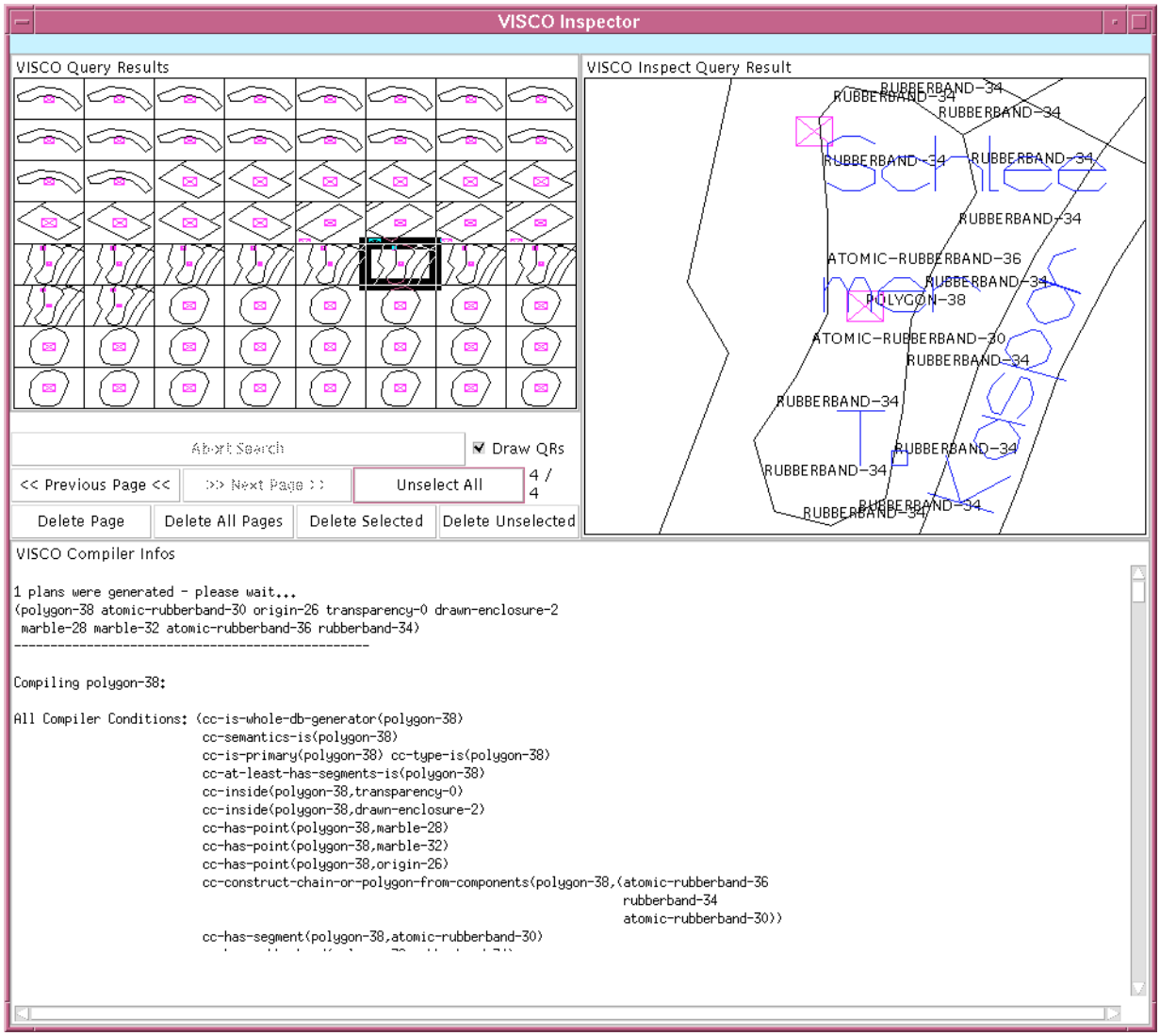


Figure 5.6: Execution and result inspection.

binary (ternary, ...) operators (e.g. a marble can be the derived intersection point of two beams). The ASG is constructed by a sequence of internal operator applications provided by the ASG module, each checking its applicability by a list of *preconditions*. By enforcement of these preconditions we ensure that only syntactically correct ASGs can be constructed. Most of the user's interactions can directly be mapped to sequences of these internal operators. No advanced parsing techniques from visual language theory are necessary; however, the above mentioned topological relationships between language elements are recognized by algorithms borrowed from computational geometry.

In fact, the graphical query editor maintains an internal construction or application sequence of these internal ASG operators (a beautified subset of this sequence is shown in the "VISCO Control" window). After a user operation, the internal history is updated:

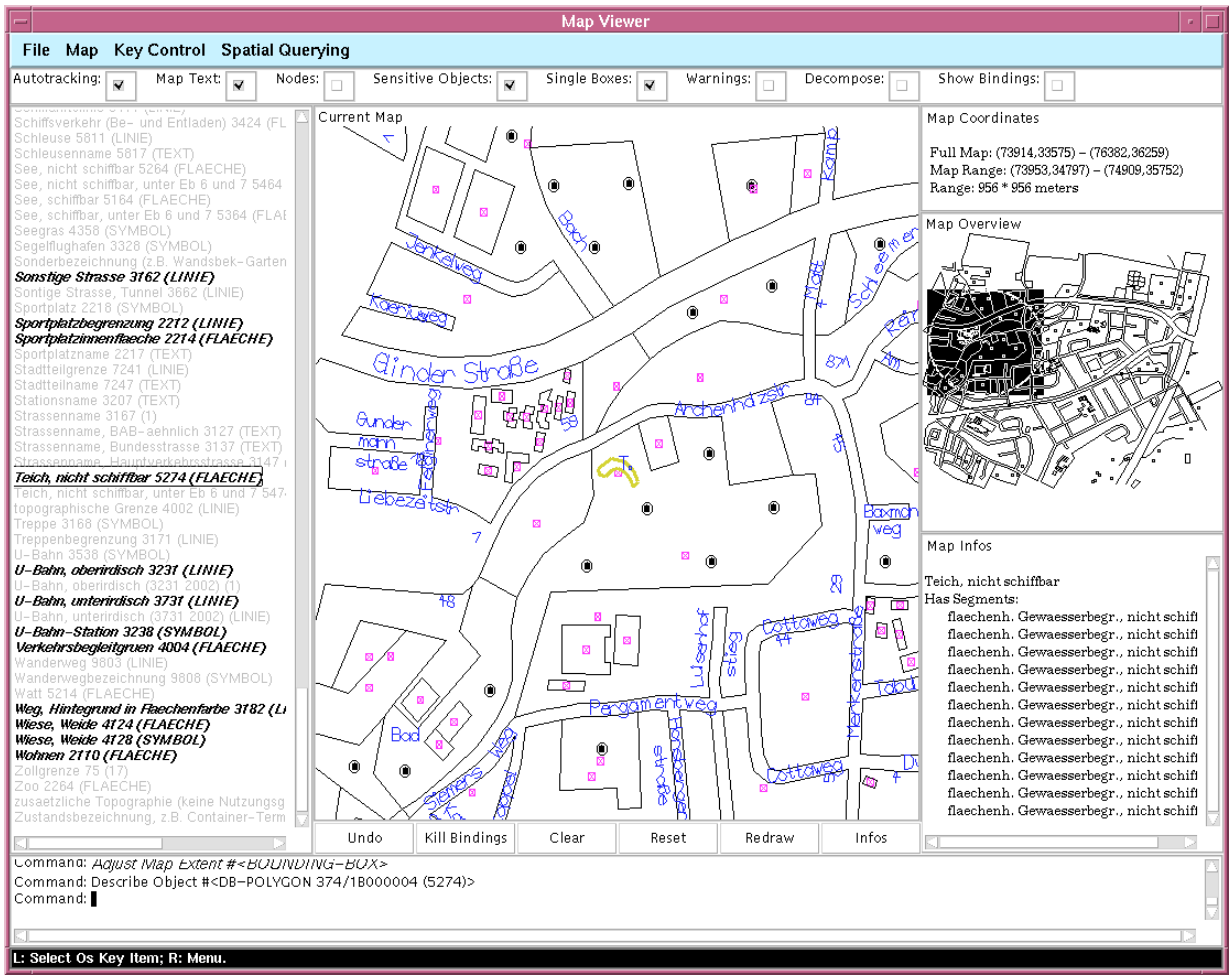


Figure 5.7: The Map Viewer.

an entry can be added, deleted or modified. In the case of the removal or modification of an entry of the history, the ASG is simply reconstructed by replaying the whole internal history. If an error is encountered during the reconstruction phase because of an unfulfilled precondition, the user's interaction is regarded as invalid and automatically undone by the system. However, in our experience this "frustrating situation" does not appear too often (the only critical interactions regarding this are "move" and "delete"). The reconstruction of the ASG is fast enough and therefore appears instantaneously to the user.

The process of *query compilation* can be easily explained by using a *petri net model* (in fact, the compiler can be viewed as a special petri net). A VISCO query has to be considered as highly declarative — many possible *execution plans* can be expected. The *optimizer* determines the best of these plans (by assigning cost weights to plans) and uses this plan to construct a Lisp program that will search the database simply in a depth-first manner (backtracking with "generate-and-test"). A plan itself is basically a sequence of nodes of the ASG representing the order of sequence in which query objects are matched to objects

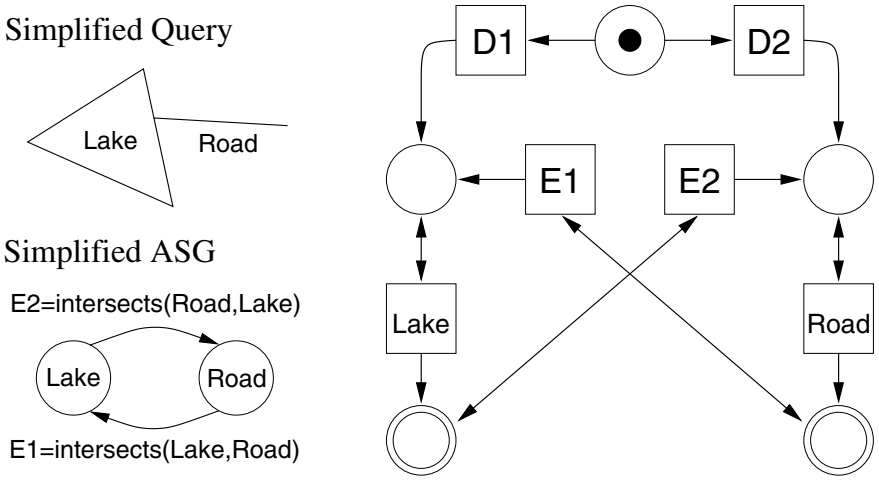


Figure 5.8: A simple ASG and its corresponding C/E net.

in the spatial database. However, to find the best of these potentially $n!$ number of plans is a very hard problem and can only be addressed heuristically.

In contrast to query objects, universal objects must be constructed or calculated and can not be searched for (either their operands have to be known and bound in the case of derived universal objects or their component objects in the case of complex universal objects). This demonstrates the possibility of having a large number of dependencies between objects that might reduce the number of possible execution (search) plans.

In Figure 5.8, a very simple ASG and its corresponding C/E petri net are shown.[1] The nodes "Lake" and "Road" (representing query objects) stand in an "intersects" relation (note that by taking the component relations for the – here not shown – endpoints of the road into account, we would really get something like "touching"). Two plans are possible: First, we could search for "Lake" and then for "Road" by using the edge E2 as a *generator* (we use a *spatial index* supporting spatial join and selection operations based on topological relationships). This requires that the edge E1 has to be *deferred.* But the reverse order is also possible. Multiple plans can be derived by traversing edges or their inverses. The *possible plans* are now given as *processes* (sequences of firing transitions) of this simple net (the transitions D1 and D2 stand for "defer E1" and "defer E2"): D1–Lake–E2–Road or D2–Road–E1–Lake.[2] A firing transition usually generates Lisp code for the search program. In a *complete* plan, each double-circled place must be marked. The quality of the generated plans can differ dramatically (factor 10000 or more in execution time). The heuristics used by the optimizer are not discussed here.

[1]In a *condition event (C/E) net*, each *place* has a maximal capacity of one – a *transition* is *activated*, if each of its in-places is marked with *tokens* and all its out-places are empty *after* removing all tokens from the in-places.

[2]The first plan is better since we have many roads intersecting other elements (e.g. roads) but only very few lakes.

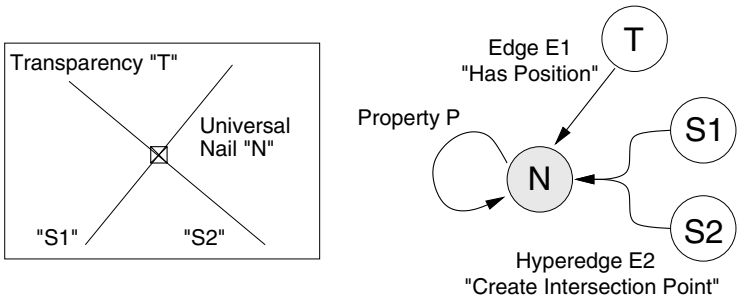


Figure 5.9: A node "N" of a more complex ASG.

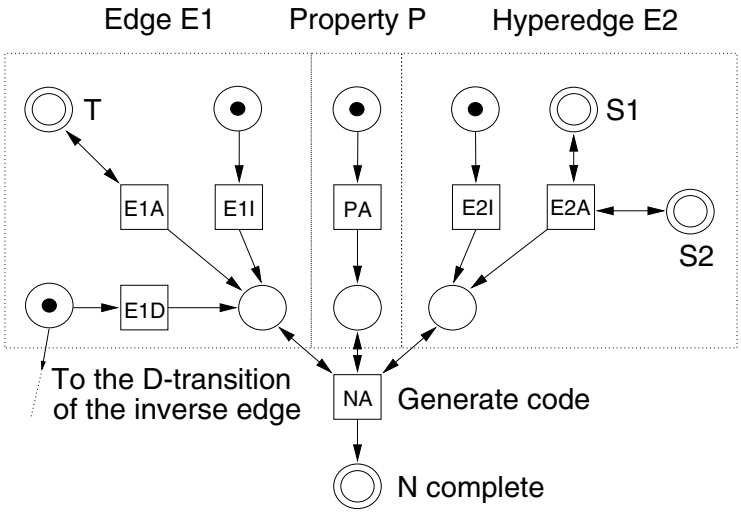


Figure 5.10: Corresponding C/E net for node "N".

A more complex example is shown in Figure 5.9 and Figure 5.10. Sometimes, additional complex dependencies that can not be modeled by simple C/E nets must be taken into account. Therefore, transitions might be annotated with additional predicates that must be fulfilled before firing (in addition to the firing rule). Edges might be *ignored* under certain circumstances if their corresponding condition is already *implied* by the pre-history (see the annotated transitions E1I, E2I).

5.4 Summary

In this chapter we presented an advanced prototype (fully implemented and operational) for a sketch-based visual spatial query language. A major advantage of this approach is the direct visibility of an object's meaning. The strong physical metaphors for language elements make the intended semantics and therefore the interpretation chosen by the system explicit. The built-in browser described here facilitates a step-wise focusing and understanding of the current query as a whole. Mismatches between the intended meaning of a query as posed by a user and its interpretation as computed by the system are therefore

mostly avoided. We argue that it can be very difficult or even impossible (at the current state of art in artificial intelligence) to correctly grasp the users intentions (the relevant aspects) from a *freestyle drawn sketch* (like the ones assumed in [Egenhofer, 1996]). Therefore, in order to get a practical system working *today*, the gap can not be bridged by the system alone. We still need the semantic input from users and we need systems offering active support, so that both are meeting halfway. The described GUI provides an embedding spatial querying environment, regarding querying as an incremental, step-wise process of selections, result inspections and further refinements.

A first approach to using description logics for specifying the semantics of visual spatial queries is presented in Chapter 8.

Part III

Spatial Reasoning with Description Logics

This part presents two contributions to description logics that were inspired by the need to integrate reasoning about conceptual descriptions and qualitative spatial relations. The major contribution of Chapter 6 is the incorporation of characteristics of space into the semantics of a description logic similar to $\mathcal{ALEN}$. The main idea is to treat a region as a set of points and to extend the subsumption relationship between concepts to subsumption between spatial regions. A map interpretation problem is considered as an example and used to demonstrate how conceptual background knowledge can be exploited for image interpretation tasks. The chapter is based on [Haarslev et al., 1994; Haarslev and Möller, 1997a; Haarslev and Möller, 1997b; Möller et al., 1998].

Chapter 7 extends the work of the previous chapter by a more general approach. It presents the description logic $\mathcal{ALCRP}(\mathcal{D})$ with concrete domains and a role-forming predicate operator as its prominent aspects. The feasibility of $\mathcal{ALCRP}(\mathcal{D})$ for reasoning about spatial objects and their qualitative spatial relationships is demonstrated by providing an appropriate concrete domain for spatial objects. The theory is motivated as a basis for knowledge representation and query processing in the domain of geographic information systems. In contrast to existing work in this domain, which mainly focuses either on conceptual reasoning or on reasoning about qualitative spatial relations, this theory integrates reasoning about spatial information with terminological reasoning. We conclude this chapter with examples for spatioterminological reasoning in the GIS domain. The chapter is based on [Haarslev et al., 1994; Lutz et al., 1997; Möller et al., 1997; Haarslev et al., 1998a; Haarslev et al., 1999b].

Chapter 6

Integrating Qualitative Spatial Reasoning into Description Logics

Qualitative relations play an important role in formal reasoning systems. We emphasize that inferences about spatial relations should not be considered in isolation but should be integrated with formal inferences about concepts (e.g. automatic consistency checking and classification). The semantics of qualitative relations should be grounded in a quantitative representation of spatial data. In our opinion, the abstractions provided by qualitative spatial relations can be interpreted as an interface between a conceptual model about the world and quantitative spatial data representing spatial information about domain objects.

6.1 Introduction

The combination of conceptual and spatial inference services can be used to solve important application problems. In the following we illustrate how terminological inferences with spatial relations can be used for image interpretation. The characteristic of these problems is that it is often very difficult to describe a fixed algorithm that defines an exact sequence of "interpretation steps" because several different "cues" have to be integrated. In other words: the solution must be computed by adequately integrating partial information about domain objects. The information about objects is given by conceptual background knowledge, the image itself and different kinds of intermediate interpretation results. According to the work of Schröder and Neumann [Schröder and Neumann, 1996] which was inspired by the MAPSEE approach [Reiter and Mackworth, 1989], image interpretation can be defined as a (re-)construction process of a specific possible world that is consistent with the given knowledge (see also Section 6.5).

In this chapter, we consider a map interpretation problem and demonstrate how conceptual background knowledge can be exploited for image interpretation tasks. In a geographical information system, queries like "search for a living area in a border district with recreation areas" might be defined. We assume that the necessary data are automatically gathered using image interpretation techniques. Note that in our setting image interpretation starts

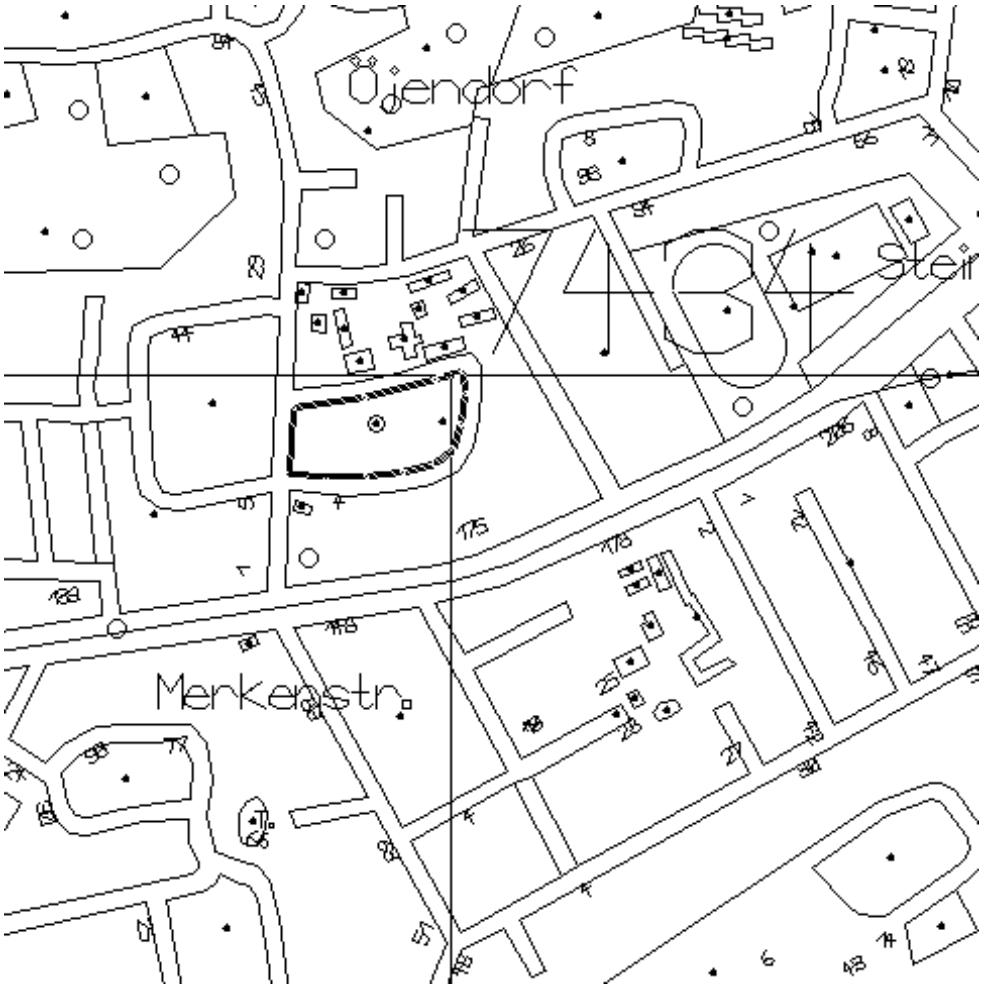


Figure 6.1: A subsection of Öjendorf (a district of the city of Hamburg).

with vector data, i.e. areas are defined by polygons. As an example, a subsection of a vector based map of the city of Hamburg is shown in Figure 6.1. Polygons from the image data are already annotated with labels like "living-area", "ordinary-road" etc. In order to correctly interpret the image, different kinds of world knowledge are required. For instance, with background knowledge one can infer that the large number 7434 in the upper right corner of Figure 6.1 cannot be a zip code nor can it describe the height of a mountain (not in Northern Germany). The required inference steps can be formalized by combining spatial and terminological reasoning. A proposal for a solution of this task is discussed in Section 6.4.

The spatial part of the theory is based on Egenhofer's set of topological relations. In contrast to [Haarslev et al., 1994] and Chapter 3 where topological relations are used as primitives in the sense of logic (i.e. they are semantically uninterpreted), we extend the treatment of topological relations by interpreting their semantic definition with respect to concept entailment (cf. the notion of subsumption: one concept is more general than

another) and demonstrate their influence on automatic concept classification.

Thus, the theory presented in this chapter allows one to detect both inconsistencies and implicit information in formal conceptual models for spatial domain objects. On the one hand, it can be shown that concept definitions and subsumption (or inheritance) relations restrict the set of possible relations between domain objects. On the other hand, definitions about topological relations might define implicit subsumption relationships which have to be automatically detected to capture all kinds of possible inferences that are sanctioned by the semantics of the representation formalism.

The major contribution of this chapter is the incorporation of characteristics of space into the semantics of the inference system. The main idea is to treat a region as a set of points and to extend the subsumption relationship between concepts to subsumption between spatial regions. A region R_1 can be defined to subsume another region R_2 when R_1 "contains" R_2 (see Section 6.2 for a formal definition of spatial relations). Basically, for spatial subsumption, the same set-inclusion semantics as for concept languages is used. For our application we consider spatial point sets defined by polygons. Qualitative relations between two-dimensional areas are defined by topological relations between polygons.

6.2 Qualitative Modeling

The previous section motivated the formalization (qualitative modeling) of space with the help of conceptual and spatial inference services. This section introduces the formal tools used for qualitative modeling. We define spatial regions and their qualitative relationships and combine them with a description logic framework extended by a spatial reasoner.

The definition of basic geometric objects usually relies on topology which is itself a basis for defining relationships between objects. In the following we assume the usual concepts of point-set topology with open and closed sets [Spanier, 1966]. The *interior* of a set λ_i (denoted by λ_i°) is the union of all open sets in λ_i. The *closure* of λ_i (denoted by $\overline{\lambda_i}$) is the intersection of all closed sets containing λ_i. The *complement* of λ_i (denoted by λ_i^{-1}) with respect to the embedding space $\mathbb{R}^n$ is the set of all points of $\mathbb{R}^n$ not contained in λ_i. The *boundary* of λ_i (denoted by $\partial\lambda_i$) is the intersection of the closure of λ_i and the closure of the complement of λ_i.

The following restrictions apply to every pair of sets. (1) λ_i, λ_j be n-dimensional and $\lambda_i, \lambda_j \subset \mathbb{R}^n$, (2) $\lambda_i, \lambda_j \neq \emptyset$, (3) all boundaries, interiors, and complements are connected, and (4) $\lambda_i = \overline{\lambda_i^\circ}$ and $\lambda_j = \overline{\lambda_j^\circ}$.

Using these definitions we can define 13 binary topological relations that are organized in a subsumption hierarchy (see Figure 6.2). The leaves of this graph represent eight mutually exclusive relations that cover all possible cases with respect to the restrictions mentioned above. The eight relations are also referred to as *elementary relations*. The elementary relations are equivalent to the set of eight relations defined by Egenhofer [Egenhofer, 1991] and others [Randell et al., 1992; Clementini et al., 1993]. Figure 6.3 illustrates five of these eight relations (the inverses and the relation "equal" are omitted). The 13 relations are

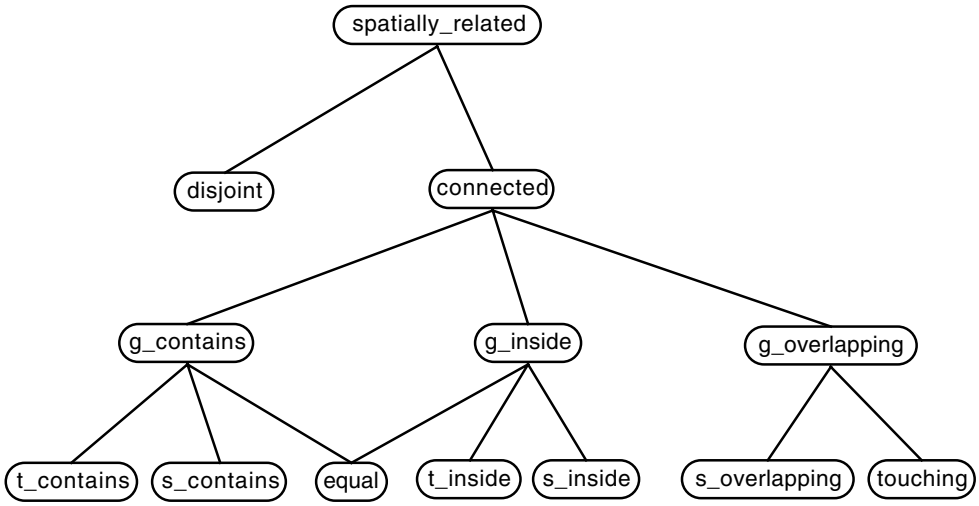


Figure 6.2: Hierarchy of spatial relations.

defined as follows:

- **spatially_related**: Two objects have a *spatial relationship* between each other. This relation is defined as the disjunction of its two mutually exclusive subrelations disjoint and connected.

  $\mathsf{spatially\_related}(\lambda_1, \lambda_2) \equiv \mathsf{disjoint}(\lambda_1, \lambda_2) \vee \mathsf{connected}(\lambda_1, \lambda_2)$

- **disjoint**: Two objects are *disjoint* if their intersection is empty; disjoint is symmetric.

  $\mathsf{disjoint}(\lambda_1, \lambda_2) \equiv \lambda_1 \cap \lambda_2 = \emptyset$

- **connected**: Two objects are *connected* if their intersection is non-empty; connected is symmetric.

  $\mathsf{connected}(\lambda_1, \lambda_2) \equiv \lambda_1 \cap \lambda_2 \neq \emptyset$

- **g_overlapping**: Two objects are *generally overlapping*. This relation is defined as the disjunction of its two mutually exclusive subrelations touching and s_overlapping; g_overlapping is symmetric.

  $\mathsf{g\_overlapping}(\lambda_1, \lambda_2) \equiv \mathsf{touching}(\lambda_1, \lambda_2) \vee \mathsf{s\_overlapping}(\lambda_1, \lambda_2)$

- **touching**: Two objects are *touching* if only their boundaries are intersecting; touching is symmetric.

  $\mathsf{touching}(\lambda_1, \lambda_2) \equiv \mathsf{connected}(\lambda_1, \lambda_2) \wedge (\lambda_1^{\circ} \cap \lambda_2^{\circ} = \emptyset)$

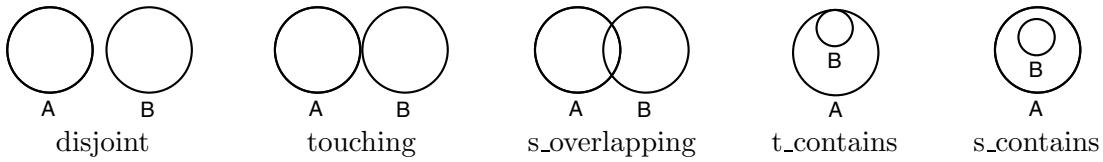


Figure 6.3: Spatial relations between A and B.

- **s_overlapping**: Two objects are *strictly overlapping* if they are connected and their intersection is not equal to either of them; s_overlapping is symmetric.
  $\mathsf{s\_overlapping}(\lambda_1, \lambda_2) \equiv$
  $\mathsf{connected}(\lambda_1, \lambda_2) \wedge (\lambda_1 \cap \lambda_2 \neq \lambda_1) \wedge (\lambda_1 \cap \lambda_2 \neq \lambda_2) \wedge (\lambda_1^{\circ} \cap \lambda_2^{\circ} \neq \emptyset)$

- **g_contains/g_inside**: An object λ_1 *generally contains* an object λ_2. This relation is defined as the disjunction of its three mutually exclusive subrelations equal, t_contains, and s_contains; g_inside is the inverse of g_contains; g_contains and g_inside are reflexive, antisymmetric, and transitive.
  $\mathsf{g\_contains}(\lambda_1, \lambda_2) \equiv \mathsf{t\_contains}(\lambda_1, \lambda_2) \vee \mathsf{s\_contains}(\lambda_1, \lambda_2) \vee \mathsf{equal}(\lambda_1, \lambda_2)$

- **equal**: The relation equal is symmetric and transitive.
  $\mathsf{equal}(\lambda_1, \lambda_2) \equiv \lambda_1 = \lambda_2$

- **t_contains/t_inside**: An object λ_1 *tangentially contains* an object λ_2 if their intersection is equal to λ_2 and the intersection of their boundaries is non-empty; the inverse of t_contains is t_inside; t_contains and t_inside are asymmetric.
  $\mathsf{t\_contains}(\lambda_1, \lambda_2) \equiv (\lambda_1 \cap \lambda_2 = \lambda_2) \wedge (\lambda_1 \cap \lambda_2^{-1} \neq \emptyset) \wedge (\partial\lambda_1 \cap \partial\lambda_2 \neq \emptyset)$

- **s_contains/s_inside**: An object λ_1 *strictly contains* an object λ_2 if their intersection is equal to λ_2 and only the interiors of their regions intersect; the inverse of s_contains is s_inside; s_contains and s_inside are asymmetric and transitive.
  $\mathsf{s\_contains}(\lambda_1, \lambda_2) \equiv (\lambda_1 \cap \lambda_2 = \lambda_2) \wedge (\lambda_1 \cap \lambda_2^{-1} \neq \emptyset) \wedge (\partial\lambda_1 \cap \partial\lambda_2 = \emptyset)$

6.3 Spatial Reasoning for Polygons

This section introduces a spatial reasoner that realizes inference services over 2D polygons. We demonstrate that the reasoning services provided by the description logic $\mathcal{ALC}(\mathcal{D})$ are insufficient for the formalization of space and propose an extension.

The fundamental idea of the spatial reasoner is the treatment of spatial regions as subsets of $\mathbb{R}^2$ and to define subsumption between polygons with respect to the relation g_contains

as defined in Section 6.2 (see also Figure 6.2). The relation g_contains has the properties of an order relation (reflexive, antisymmetric, transitive), i.e. it has the same properties as the subsumption relation for concepts. With this definition of spatial subsumption we can reduce the satisfiability problem to the decision whether a set of polygons is connected (i.e. there exists a non-empty intersection) or disjoint.

We introduce spatial predicates applicable to descriptions of polygons. With the polygon restriction we gain applicability of efficient algorithms (e.g. the simplex procedure) for solving the satisfiability problem. We use one-place predicates for expressing equality ($\mathsf{equal_p}$) or containment ($\mathsf{g\_inside_p}$) of polygons with respect to a reference polygon p which is used as the second argument of the relation (e.g. $\mathsf{g\_inside_p}(\lambda_x) \equiv \mathsf{g\_inside}(\lambda_x, \mathsf{p})$). Note, that such a polygon name p is treated as a constant by the spatial reasoner. We have the choice to either represent this polygon as a polyline with appropriate coordinate values (and use the simplex procedure for deciding connectedness) or explicitly assert for a polygon name p its qualitative relationships with other polygon names (e.g. $\mathsf{g\_inside}(\mathsf{p}, \mathsf{q})$). The second alternative could be realized by integrating into the spatial reasoner "background knowledge" about particular polygons (e.g. the relationships as described in Figure 6.5).

We assume an attribute has_area whose filler is from the spatial domain (i.e. a polygon description). For instance, we can now define a concept northern_german_region by using the 'for-all' constructor $\forall\, \mathsf{r}\,.\,\mathsf{P}$ and a unique polygon name $\mathsf{p_5}$.

$$\textbf{northern\_german\_region} \doteq \forall\, \mathsf{has\_area}\,.\,\mathsf{g\_inside_{p_5}}$$

For northern_german_region the possible filler of has_area is restricted to a polygon inside of $\mathsf{p_5}$. The polygon $\mathsf{p_5}$ defines the area of Northern Germany. The construct $\mathsf{g\_inside_{p_5}}$ subsumes every region of Northern Germany whose associated polygon is g_inside of $\mathsf{p_5}$. Additionally, we need a predicate for expressing equality of polygons since subsumption of arbitrary subregions is not always desired. For instance, the concept federal_state_hh (HH is part of the car license number for Hamburg) contains the equality condition in order to prevent subsumption with subregions of the city of Hamburg area (see also Figure 6.5):

$$\textbf{federal\_state\_hh} \doteq \mathsf{german\_federal\_state} \sqcap \forall\, \mathsf{has\_area}\,.\,\mathsf{equal_{p_2}}$$

The polygon $\mathsf{p_2}$ defines the area of the federal state Hamburg. $\mathsf{equal_{p_2}}$ does not subsume any subregion of $\mathsf{p_2}$. Note that due to the definition of g_inside, $\mathsf{g\_inside_{p_2}}$ subsumes $\mathsf{equal_{p_2}}$. However, we need more expressivity in order to adequately characterize spatial relationships between certain individuals. For instance, the concept definition for hh_border_district specifies that an individual is an instance of hh_border_district iff it is an instance of district_of_hh and is associated with another individual which is an instance of hh_border_district and both individuals have fillers for has_area such that the predicate t_inside holds between these fillers. This situation is illustrated in Figure 6.4.

$$\textbf{hh\_border\_district} \doteq \mathsf{district\_of\_hh} \sqcap (\bigcirc\ \mathsf{t\_inside}\,.\,\mathsf{federal\_state\_hh})$$

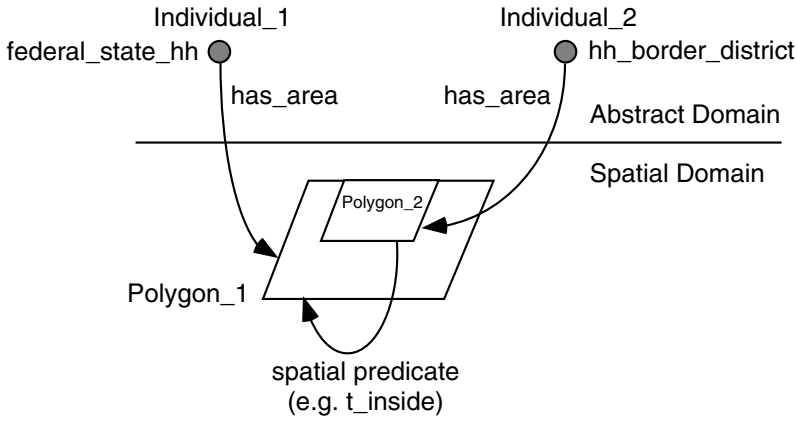


Figure 6.4: Relationship between abstract and spatial domain.

In a first approach, we propose the concept-forming operator $\bigcirc$ that captures the intuition as indicated above. The assignment function is extended for $\bigcirc$ concept terms (be sr a name for an elementary spatial relation as defined in Section 6.2 and C be a concept term):

$$(\bigcirc\ \mathsf{sr}\,.\,\mathsf{C})^{\mathcal{I}} = \{x|\ \exists\, y_1, y_2, z : (x, y_1) \in \mathsf{has\_area}^{\mathcal{I}},\ (z, y_2) \in \mathsf{has\_area}^{\mathcal{I}},\\ (y_1, y_2) \in \mathsf{sr}^{\mathcal{I}},\ x \neq z,\ z \in \mathsf{C}^{\mathcal{I}}\}$$

Note that this concept-forming operator is restricted since only elementary spatial relations and not abstract roles are allowed in place of sr.

6.4 Spatioterminological Inferences: an Extended Example

The use of the constructs presented in the previous section is demonstrated with a city map example. Figure 6.5 illustrates the surrounding area of Öjendorf (shown in Figure 6.1). The city Hamburg (represented by the polygon $\mathsf{p_2}$) is located in Germany (polygon $\mathsf{p_1}$), especially in Northern Germany (polygon $\mathsf{p_5}$) and next to the federal state Schleswig-Holstein (polygon $\mathsf{p_4}$). The district Öjendorf (polygon $\mathsf{p_3}$) is located inside of the area of Hamburg. Actually, it is a border district to Schleswig-Holstein.

The formal model is presented with a description logic TBox shown in Figure 6.6. The concept $\mathsf{german\_federal\_state}$ is a primitive concept, i.e. it is defined only with necessary conditions. For instance, the filler of $\mathsf{has\_area}$ for an instance of $\mathsf{german\_federal\_state}$ must be a polygon name which is restricted to $\mathsf{g\_inside_{p_1}}$ which, in turn, describes a polygon inside $\mathsf{p_1}$.

Because the concept $\mathsf{german\_federal\_state}$ is primitive, it does not subsume the concept $\mathsf{northern\_german\_region}$. However, due to spatial subsumption, $\mathsf{northern\_german\_region}$ subsumes $\mathsf{federal\_state\_hh}$ and also $\mathsf{federal\_state\_sh}$ (the concept for Schleswig-Holstein). The

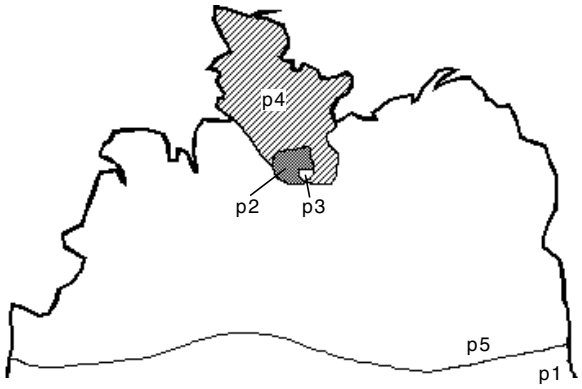


Figure 6.5: A sketch of the northern part of Germany with polygons for Germany (p_1), Northern Germany (p_5), the federal states Schleswig-Holstein (p_4) and Hamburg (p_2) as well as a small district of Hamburg (p_3). Polygon p_3 is assumed to be inside p_2 but p_2 is not inside p_4.

predicates min_n which means $[\mathsf{n} \ldots \infty]$ and max_n which means $[-\infty \ldots \mathsf{n}]$ are defined over natural numbers and are also provided by some DL systems.

Another northern_german_region is district_of_hh. Note again that the area of district_of_hh is not a german_federal_state because this concept is primitive (see above). The same holds for district_of_sh.

While the implicit subsumption relationships discussed above are quite obvious, the last two concept definitions provide more difficult examples. Based on the definitions given above, it can be proven that hh_border_district_to_sh is subsumed by hh_border_district. A hh_border_district is a district_of_hh which touches the area of federal_state_hh from the inside (relation t_inside). The polygon of federal_state_hh is explicitly given by the predicate $\mathsf{equal}_{\mathsf{p}_2}$ (see the concept definition of federal_state_hh). If a district_of_hh touches the polygon of federal_state_sh, its corresponding area must be tangentially inside the polygon of federal_state_hh.

A TBox classifier that deals with the semantics of spatial relations must find these implicit subsumption relationships in order to correctly and completely classify the terminological knowledge base.

In some DL systems, a set of rules can be defined to assert additional constraints for ABox instances when certain conditions (represented by a concept term) are met. For instance, in Hamburg and Schleswig-Holstein, the mountain height is less than 1000 (meters). This relationship is asserted by a rule (operator $\rightarrow$) that fires for every individual that is classified as a member of the concept northern_german_region. The second rule adds additional constraints to instances of district_of_hh.

northern_german_region $\rightarrow \forall\, \mathsf{mountain\_height} \,.\, \mathsf{max}_{1000}$

district_of_hh $\rightarrow \forall\, \mathsf{zip\_code} \,.\, \mathsf{min}_{20000} \sqcap \forall\, \mathsf{area\_descriptor} \,.\, \mathsf{min}_{1000}$

$\textbf{german\_federal\_state} \sqsubseteq \forall\, \mathsf{has\_area} \,.\, \mathsf{g\_inside}_{\mathsf{p_1}}$

$\textbf{northern\_german\_region} \doteq \forall\, \mathsf{has\_area} \,.\, \mathsf{g\_inside}_{\mathsf{p_5}}$

$\textbf{federal\_state\_hh} \doteq \mathsf{german\_federal\_state} \sqcap \forall\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p_2}}$

$\textbf{federal\_state\_sh} \doteq \mathsf{german\_federal\_state} \sqcap \forall\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p_4}}$

$\textbf{district\_of\_hh} \doteq \forall\, \mathsf{has\_area} \,.\, \mathsf{g\_inside}_{\mathsf{p_2}}$

$\textbf{district\_of\_sh} \doteq \forall\, \mathsf{has\_area} \,.\, \mathsf{g\_inside}_{\mathsf{p_4}}$

$\textbf{sh\_border\_district} \doteq \forall\, \mathsf{has\_area} \,.\, \mathsf{g\_inside}_{\mathsf{p_4}} \sqcap (\bigcirc\ \mathsf{t\_inside} \,.\, \mathsf{federal\_state\_sh})$

$\textbf{hh\_border\_district} \doteq \mathsf{district\_of\_hh} \sqcap (\bigcirc\ \mathsf{t\_inside} \,.\, \mathsf{federal\_state\_hh})$

$\textbf{hh\_border\_district\_to\_sh} \doteq \mathsf{district\_of\_hh} \sqcap$
$(\bigcirc\ \mathsf{spatially\_related} \,.\, \mathsf{federal\_state\_hh}) \sqcap (\bigcirc\ \mathsf{touching} \,.\, \mathsf{federal\_state\_sh})$

Figure 6.6: TBox for the Northern Germany example.

Automatic classification is also important for assertional knowledge defined in the ABox. The following statements define partial information about individuals in our city map domain.

$$\begin{aligned}
\mathsf{hamburg} &: \mathsf{federal_state_hh} \\
\mathsf{\ddot{o}jendorf} &: \mathsf{district_of_hh} \\
\langle \mathsf{\ddot{o}jendorf}, \mathsf{p_3} \rangle &: \mathsf{has_area} \\
\mathsf{vierlande} &: \forall\, \mathsf{has_area} \,.\, \mathsf{g_inside}_{\mathsf{p_4}} \wedge \\
&\qquad (\bigcirc\ \mathsf{touching} \,.\, \forall\, \mathsf{has_area} \,.\, \mathsf{equal}_{\mathsf{p_3}}) \wedge (\bigcirc\ \mathsf{spatially_related} \,.\, \mathsf{federal_state_sh})
\end{aligned}$$

The individual hamburg is declared to be an instance of federal_state_hh. The individual öjendorf is a district_of_hh. The filler of the has_area role for öjendorf is $\mathsf{p_3}$. The ABox reasoner computes that the federal state hamburg and the district öjendorf are instances of the concept northern_german_region. This is basically due to the fact that the polygon $\mathsf{p_2}$ and $\mathsf{p_3}$ are inside the polygon $\mathsf{p_5}$. Thus, the mountain heights in the associated areas are asserted to be less than 1000 meters (see the rules defined above). This kind of derived information can be used to guide the map interpretation process by applying conceptual background knowledge. If the number 7434 in Figure 6.1 were asserted as a filler for the mountain height of öjendorf, the ABox would derive an inconsistency which indicates that another hypothesis has to be tried.

In the last assertion, another individual (named vierlande) which touches the polygon of öjendorf is defined. Since vierlande is by definition subsumed by $\forall\, \mathsf{has\_area} \,.\, \mathsf{g\_inside}_{\mathsf{p_4}}$, it cannot be a district_of_hh but must be inside of district_of_sh. However, it touches the Öjendorf polygon ($\mathsf{p_3}$) and therefore, it must be automatically classified as a sh_border_district.

The examples illustrate the importance of complete inference algorithms for TBox and ABox classification. For instance, if the implicit subsumption relationship between the concepts hh_border_district and hh_border_district_to_sh were not detected, we could declare an instance of hh_border_district_to_sh and claim that a valid zip code in this area might be 7434 which is certainly inconsistent (cf. the rule definition for district_of_hh). Another hypothesis is that 7434 might be an area descriptor. This hypothesis is consistent with the terminological background knowledge defined in our TBox example and might be used as an intermediate result for further interpretation steps.

6.5 Related Work

The idea of incorporating conceptual knowledge (especially knowledge that can be modeled with a decidable description logic) into spatial reasoning and image interpretation problems has been proposed in [Haarslev et al., 1994]. Rather than Reiter and Mackworth (see the description of MAPSEE in [Reiter and Mackworth, 1989]), who use first order predicate logic, we use a description logic as a basis for our image interpretation problems. In order to be able to validate the image interpretation knowledge itself (i.e. the TBox), we cannot include a domain closure axiom, i.e. we cannot enumerate all objects in every image to be interpreted. In other words: Neither can the problem be reduced to model checking nor to satisfiability checking in propositional logic. Lange and Schröder [Lange and Schröder, 1994] also discuss the problem of image interpretation in a formal, logical framework. The incorporation of concrete domain predicates for image interpretation problems is presented by Schröder and Neumann [Schröder and Neumann, 1996; Schröder, 1998].

Many other approaches for modeling spatial objects and their relations have been published. The ontological assumptions for the approach presented in this chapter are based on a Newtonian conception of space (see [Borgo et al., 1996]). In contrast to the Leibnizian conception (assuming space to be strictly dependent on the relations holding between physical objects), the cartesian structure of our spatial reasoning approach allows spatial relations to be defined by topological relations between areas defined by polygons (with an external or absolute reference system). The Leibnizian conception has been adopted in many approaches inspired by natural language interpretation problems. For sake of brevity, we do not discuss the large amount of work on logical models of space in this area.

Grigni et al. [Grigni et al., 1995] study the computational problems in developing an inference system for checking the satisfiability of (conjunctive) combinations of spatial relations. This work is important for checking the consistency of combinations of concept terms containing predicates based on spatial relations. Grigni et al. point out that in topological inferences the aspects of relational consistency and planarity interact in rather complex ways. They showed that besides the relational consistency problem a planarity problem has to be solved when areas are assumed to be disjoint. With this additional restriction, the complexity of the satisfiability problem becomes NP-hard in many cases. Lemon [Lemon, 1996] showed that in some "spatial" logics based on convex regions one can construct consistent sentences that have no models in the intended geometrical interpretation, i.e. the

logics are incomplete with respect to the intended geometrical interpretation.

6.6 Summary

In this chapter, we have demonstrated that topological relations directly influence the kind of conceptual or terminological knowledge that can (and must) be derived by a formal inference engine. On the other hand, assertions about concepts restrict the set of possible spatial relations between different individuals.

We have seen that the use of incomplete reasoning services in practical applications is problematic. For instance, in our application domain the reasoning service might be used to test whether the hypothesis "7434 is the zip code of Öjendorf" is consistent. In the example above we have seen that the correct answer depends on complete TBox classification algorithms. In this case, an incomplete reasoner that does not detect the implicit subsumption relationships in the TBox (see the discussion above) must answer "may be". However, if we pose the negated query "Is 7434 definitely not the zip code of Öjendorf" the answer must also be "may be" because an inconsistency cannot be derived due to incompleteness. The question is whether "may be" answers can be used for solving problems in a geographical information system, especially when "may be" happens to be interpreted as "no." Similar problems are likely to occur in incomplete approaches (see e.g. [Russ et al., 1996] for an image interpretation approach that uses an undecidable description logic).

One idea of the approach presented in this chapter is to reduce the complexity of the reasoning algorithms by also considering quantitative spatial data which are available in many practical applications. If concrete polygons are given, no relational consistency checking (see above) is required but standard algorithms from computational geometry can be used. In our map interpretation scenario, the incorporation of a spatial reasoner with a Newtonian view (i.e. with quantitative data) helps to avoid problems of so-called "spatial" logics. We have discussed some arguments that dealing only with qualitative relations neglects some aspect of space (cf. [Grigni et al., 1995; Lemon, 1996]) when, for instance, the qualitative calculus implies additional properties of geometric objects such as convexity or disjointness of regions.

The spatial reasoning extension proposed in this chapter is no general geometrical theorem prover. The advantage of our approach which is based also on quantitative information about spatial regions is that the satisfiability test for finite conjunctions of predicates can be reduced to well-known algorithms in computational geometry (basically polygon intersection). Qualitative relations that are grounded in quantitative data provide a bridge to conceptual knowledge and allow more extensive reasoning services to be exploited for solving practical problems.

The treatment of predicate concept terms such as $\forall\, \mathsf{has\_area} \,.\, \mathsf{g\_inside}_{\mathsf{p_i}}$ and $\forall\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p_i}}$ is also possible using a concrete domain approach (see below). A prototype implementation [Prien, 1998] using the CLASSIC description logic (and its extension interface) [Brachman et al., 1991; Borgida and Patel-Schneider, 1994; Borgida et al., 1996] demonstrates that

the concept constructor $\bigcirc$ can be integrated into CLASSIC. However, it reveals also the disadvantages of this constructor because it cannot be freely combined with other language constructs. Therefore, we investigate in the next chapter the logic $\mathcal{ALC}(\mathcal{D})$ which is extended by a role-forming predicate operator that combines abstract attributes and concrete predicates.

Chapter 7

Description Logics and Concrete Domains

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

We have developed a new description logic called $\mathcal{ALCRP}(\mathcal{D})$ in order to provide a foundation for integrating reasoning about qualitative relations with terminological reasoning using description logics. Our research is strongly motivated by spatial domains such as geographic information systems (GIS). In this context, inferences about qualitative spatial relations should not be considered in isolation but should be integrated with inferences about conceptual knowledge of domain objects (e.g. automatic consistency checking and classification). The main idea of our approach is to deal with knowledge about abstract domain objects using description logic theory and to deal with spatial objects and their qualitative *topological* relations using predicates defined over these objects. In Chapter 3 we used topological relations only as primitives in the sense of logic. Although some inferences about spatial objects are integrated into ABox reasoning, not all implicit information has been exploited for terminological reasoning.

With the help of $\mathcal{ALCRP}(\mathcal{D})$ we can extend the treatment of qualitative relations –and topological relations in particular– especially with respect to TBox reasoning. Thus, the theory presented in this chapter allows one to detect both inconsistencies and implicit information in formal conceptual models for spatial domain objects. Only the combination of terminological and topological reasoning ensures that this can be achieved according to the intended semantics of the spatial domain objects. The combination of formal conceptual and qualitative reasoning can serve as a theoretical basis for knowledge representation in the domains mentioned above and can be used to solve important application problems (see Chapter 6).

$\mathcal{ALCRP}(\mathcal{D})$ is based on the description logic $\mathcal{ALC}(\mathcal{D})$ which is defined in [Baader and Hanschke, 1991]. $\mathcal{ALC}(\mathcal{D})$ divides the set of logical objects into two disjoint sets, the *abstract* and the *concrete* objects, e.g. real numbers. Abstract objects can be related to concrete objects via features (functional roles). Relationships between concrete objects are

described with a set of domain-specific predicates. Referring to these predicates, properties of abstract objects can also be expressed using a *concept-forming* predicate operator. In $\mathcal{ALC}(\mathcal{D})$, the pair consisting of a set of concrete objects and a set of predicates defined over these objects is called a *concrete domain*. $\mathcal{ALCRP}(\mathcal{D})$ extends $\mathcal{ALC}(\mathcal{D})$ by introducing *defined roles* that are based on a *role-forming* predicate operator.

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

We present the syntax and semantics of the language for specifying concept and role inclusions. In accordance with [Baader and Hanschke, 1991] we also define the notion of a concrete domain.

Definition 7.1 (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 $\mathsf{P}_\mathcal{D} \subseteq \Delta_\mathcal{D}^n$. 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 $\mathsf{P}_1^{\mathsf{n}_1}(\mathsf{x}_{11}, \ldots, \mathsf{x}_{1\mathsf{n}_1}) \wedge \ldots \wedge \mathsf{P}_\mathsf{m}^{\mathsf{n}_\mathsf{m}}(\mathsf{x}_{\mathsf{m}1}, \ldots, \mathsf{x}_{\mathsf{m}\mathsf{n}_\mathsf{m}})$ is decidable (m is finite, $\mathsf{P}_\mathsf{i}^{\mathsf{n}_\mathsf{i}} \in \Phi_\mathcal{D}$, and x_{jk} is a name for an object from $\Delta_\mathcal{D}$).

Definition 7.2 (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 $\mathsf{f}_1\mathsf{f}_2 \ldots \mathsf{f}_\mathsf{n}$) 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 $\mathsf{u}_1, \mathsf{u}_2, \ldots, \mathsf{u}_\mathsf{n}$ as well as $\mathsf{v}_1, \mathsf{v}_2, \ldots, \mathsf{v}_\mathsf{m}$ are feature chains, then the expression $\exists(\mathsf{u}_1, \ldots, \mathsf{u}_\mathsf{n})(\mathsf{v}_1, \ldots, \mathsf{v}_\mathsf{m}).\mathsf{P}$ (*role-forming predicate operator*) is a *complex* role term. Let S be a role name and let T be a role term. Then $\mathsf{S} \doteq \mathsf{T}$ is a *terminological axiom*. This type of terminological axiom is also called *role introduction*.

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

Definition 7.3 (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, u_i is a feature chain, 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 $\mathsf{C} \sqcup \neg\mathsf{C}$ ($\mathsf{C} \sqcap \neg\mathsf{C}$).

Definition 7.4 (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 side 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 7.5 (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 $\Delta^\mathcal{I} \cup \Delta^\mathcal{D}$, 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, T be a role term, $\mathsf{u}_1, \ldots, \mathsf{u}_n$ 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:

$$
\begin{aligned}
(\mathsf{C} \sqcap \mathsf{D})^\mathcal{I} &:= \mathsf{C}^\mathcal{I} \cap \mathsf{D}^\mathcal{I} \\
(\mathsf{C} \sqcup \mathsf{D})^\mathcal{I} &:= \mathsf{C}^\mathcal{I} \cup \mathsf{D}^\mathcal{I} \\
(\neg\mathsf{C})^\mathcal{I} &:= \Delta^\mathcal{I} \setminus \mathsf{C}^\mathcal{I} \\
(\exists\, \mathsf{R}\,.\,\mathsf{C})^\mathcal{I} &:= \{x \in \Delta^\mathcal{I} \mid \exists\, y \in \Delta^\mathcal{I} : (x, y) \in \mathsf{R}^\mathcal{I}, y \in \mathsf{C}^\mathcal{I}\} \\
(\forall\, \mathsf{R}\,.\,\mathsf{C})^\mathcal{I} &:= \{x \in \Delta^\mathcal{I} \mid \forall\, y : (x, y) \in \mathsf{R}^\mathcal{I} \Rightarrow y \in \mathsf{C}^\mathcal{I}\} \\
(\exists\, \mathsf{u}_1, \ldots, \mathsf{u}_n\,.\,\mathsf{P})^\mathcal{I} &:= \{x \in \Delta^\mathcal{I} \mid \exists x_1, \ldots, x_n \in \Delta^\mathcal{D} : \\
&\qquad (x, x_1) \in {\mathsf{u}_1}^\mathcal{I}, \ldots, (x, x_n) \in {\mathsf{u}_n}^\mathcal{I}, \\
&\qquad (x_1, \ldots, x_n) \in \mathsf{P}^\mathcal{I}\}
\end{aligned}
$$

$$(\exists\,(\mathsf{u}_1,\ldots,\mathsf{u}_n)(\mathsf{v}_1,\ldots,\mathsf{v}_m)\,.\,\mathsf{P})^{\mathcal{I}} := \{(x,y)\in\Delta^{\mathcal{I}}\times\Delta^{\mathcal{I}}\mid \\ \exists x_1,\ldots,x_n,y_1,\ldots,y_m\in\Delta^{\mathcal{D}}: \\ (x,x_1)\in \mathsf{u}_1{}^{\mathcal{I}},\ldots,(x,x_n)\in \mathsf{u}_n{}^{\mathcal{I}}, \\ (y,y_1)\in \mathsf{v}_1{}^{\mathcal{I}},\ldots,(y,y_m)\in \mathsf{v}_m{}^{\mathcal{I}}, \\ (x_1,\ldots,x_n,y_1,\ldots,y_m)\in \mathsf{P}^{\mathcal{I}}\}$$

An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ iff it 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 $\mathsf{S}^{\mathcal{I}}=\mathsf{T}^{\mathcal{I}}$ for all role introductions $\mathsf{S}\doteq\mathsf{T}$ in $\mathcal{T}$.

7.1.2 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 follows:

Definition 7.6 (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 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 it is a model of $\mathcal{T}$ and furthermore satisfies all assertional axioms in $\mathcal{A}$.

7.2 Decidability and Undecidability Results

In [Lutz and Möller, 1997] as well as [Haarslev et al., 1998a] it is shown that, unfortunately, the inference problem of checking the consistency of 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.

In order to introduce the notion of restrictedness we need the notion of a negation normal form. An unfolded term is in negation normal form if negation is used only for concept names (for details see [Haarslev et al., 1999b]).

Definition 7.7 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:[1]

(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 $\mathsf{R_1}$ 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 $\mathsf{R_2}$ 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 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 $\mathcal{T}$.

These restrictions guarantee the so-called finite model property for restricted $\mathcal{ALCRP}(\mathcal{D})$. For instance, concept terms containing nested some- and all-term with defined roles (e.g. $\exists\,\mathsf{R_c}\,.\,(\forall\,\mathsf{R_c}\,.\,\mathsf{C})$ with $\mathsf{R_c}$ a defined role) are not restricted (see [Haarslev et al., 1999b] for more details).

Theorem 7.1 The ABox consistency problem for restricted $\mathcal{ALCRP}(\mathcal{D})$ concept terms is decidable if $\mathcal{D}$ is an admissible concrete domain.

Proof. The proof is given in [Haarslev et al., 1999b]. □

Proposition 7.1 The set of restricted $\mathcal{ALCRP}(\mathcal{D})$ concept terms is closed under negation.

Proof. See [Haarslev et al., 1999b]. □

[1] For technical reasons, we assume that a concept term is a subconcept term of itself.

7.3 Spatioterminological Reasoning

This section introduces a concrete domain which can represent spatial relations between domain objects. We focus on topological relations known from the RCC theory [Randell et al., 1992]. This concrete domain is used later on for two examples demonstrating spatioterminological inferences.

Before presenting the examples we briefly introduce the concrete domain $\mathcal{S}_2$. We consider specific spatial objects whose spatial representations are given as polygons. We show that $\mathcal{S}_2$ provides predicates which can be used to describe qualitative spatial RCC-8 relations as roles between spatial objects.

Definition 7.8 The concrete domain $\mathcal{S}_2 = (\Delta_{\mathcal{S}_2}, \Phi_{\mathcal{S}_2})$ is defined w.r.t. the topological space $\langle \mathbb{R}^2, 2^{\mathbb{R}^2} \rangle$. The domain $\mathcal{S}_2$ 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 is-region with $\mathsf{is\text{-}region}^{\mathcal{S}_2} = \Delta_{\mathcal{S}_2}$ and its negation is-no-region with $\mathsf{is\text{-}no\text{-}region}^{\mathcal{S}_2} = \emptyset$.

- The 8 basic predicates dc, ec, po, tpp, ntpp, tppi, ntppi and eq correspond to the RCC-8 relations. Due to brevity we 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 $\mathsf{p_1\text{-}\cdots\text{-}p_n}$ and we have $(r_1, r_2) \in \mathsf{p_1\text{-}\cdots\text{-}p_n}^{\mathcal{S}_2}$ iff $(r_1, r_2) \in \mathsf{p_1}^{\mathcal{S}_2}$ or ... or $(r_1, r_2) \in \mathsf{p_n}^{\mathcal{S}_2}$. The predicate dc-ec-po-tpp-ntpp-tppi-ntppi-eq is also called spatially-related.

- A binary predicate inconsistent-relation with $\mathsf{inconsistent\text{-}relation}^{\mathcal{S}_2} = \emptyset$ is the negation of spatially-related.

Proposition 7.2 The concrete domain $\mathcal{S}_2$ is admissible.

Proof. This is proven in [Haarslev et al., 1999b]. □

The $\mathcal{ALCRP}(\mathcal{D})$ approach together with this concrete domain is more general than the approach presented in Chapter 6 where the concept-forming operator ◯ was restricted to a set of special predicates. Moreover, $\mathcal{ALCRP}(\mathcal{D})$ supports defined roles based on predicates. This is also not possible in the approach reported in Chapter 6.

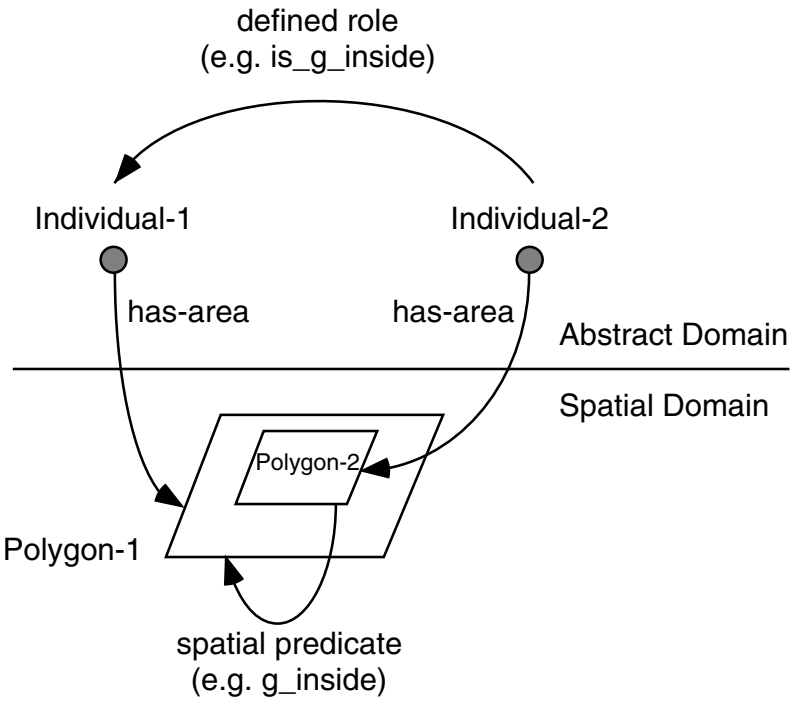


Figure 7.1: Relationship between abstract and concrete domain

7.4 Examples for Spatioterminological Reasoning

How can predicates over the concrete domain of polygons be used to support spatioterminological inferences with the description logic $\mathcal{ALCRP}(\mathcal{D})$? First of all, as an ontological commitment, we assume that each abstract domain object is associated with its spatial representation via the feature (or attribute) has_area (see Figure 7.1). For better readability, we use the topological relations from Figure 6.2 as predicate names instead of the original RCC-8 names. Now, we can use the concept-forming predicate operator in combination with one-place predicates for restricting role fillers of has_area to be specific spatial regions. For instance, subsumption between concept terms such as $\exists\,\mathsf{has\_area}\,.\,\mathsf{g\_inside}_{\mathsf{p_i}}$ (with $\mathsf{g\_inside}_{\mathsf{p_i}} \in \Phi_{\mathcal{S}_2}$) resembles inclusion of regions because every concept term $\exists\,\mathsf{has\_area}\,.\,\mathsf{g\_inside}_{\mathsf{p_i}}$ subsumes the term $\exists\,\mathsf{has\_area}\,.\,\mathsf{g\_inside}_{\mathsf{p_j}}$ iff $\mathsf{p_i}$ g_contains (i.e. spatially subsumes) $\mathsf{p_j}$.

7.4.1 Reconsidering the Hamburg Example

We reconsider the Hamburg example, already introduced in Section 6.4, and apply the above mentioned technique to modeling the regions of the German federal states, Northern Germany, a district of the city of Hamburg, etc. The advancement compared to Section 6.4 is the use of defined roles. The restrictedness criterion (cf. Definition 7.7) for the following set of TBox axioms is trivially fulfilled because they contain no nested exists or universal quantifiers.

$\textbf{northern\_german\_region} \doteq \exists\, \mathsf{has\_area} \,.\, \mathsf{g\_inside}_{\mathsf{p_5}}$

$\textbf{district\_of\_hh} \doteq \exists\, \mathsf{has\_area} \,.\, \mathsf{g\_inside}_{\mathsf{p_2}} \sqcap \exists\, \mathsf{has\_area} \,.\, \neg \mathsf{equal}_{\mathsf{p_2}}$

The concept northern_german_region is defined by an existential restriction for the attribute has_area whose filler is constrained to be any polygon that is g_inside of $\mathsf{p_5}$ which defines the area of Northern Germany (see Figure 6.5). In other words: The concept denoted by $\exists\, \mathsf{has\_area} \,.\, \mathsf{g\_inside}_{\mathsf{p_5}}$ subsumes every region of Northern Germany whose associated polygon is g_inside of $\mathsf{p_5}$. Therefore, district_of_hh is automatically classified as a subconcept of northern_german_region.

$\textbf{german\_federal\_state} \doteq \mathsf{federal\_state} \sqcap (\exists\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p_2}} \sqcup \exists\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p_4}} \sqcup \ldots)$

$\textbf{federal\_state\_hh} \doteq \mathsf{german\_federal\_state} \sqcap \exists\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p_2}}$

$\textbf{federal\_state\_sh} \doteq \mathsf{german\_federal\_state} \sqcap \exists\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p_4}}$

The concept definition of german_federal_state contains a disjunction of concept terms that characterize the locations of all possible German federal states. Due to the definition of equal, the predicate $\mathsf{equal}_{\mathsf{p}}$ does not subsume arbitrary regions in Germany. As a consequence, the area of, for instance, district_of_hh is not subsumed by the area of german_federal_state.

We also define the concepts for the federal states Hamburg and Schleswig-Holstein. We would like to emphasize that both concepts are subsumed by northern_german_region. This is due to the definition of the spatial relations in the previous section. For instance, the predicate $\mathsf{equal}_{\mathsf{p_2}}$ is subsumed by $\mathsf{g\_inside}_{\mathsf{p_2}}$ and, in turn, this predicate is subsumed by $\mathsf{g\_inside}_{\mathsf{p_5}}$ because the region $\mathsf{p_5}$ g_contains $\mathsf{p_2}$.

In many cases, restrictions about spatial relations have to be combined with additional restrictions. For example, how can we define a concept that describes a district of Hamburg that touches the federal state Hamburg from the inside? Note that it is not sufficient that the corresponding district polygon (e.g. $\mathsf{p_3}$ in Figure 6.5) is inside any polygon that is equal to the state polygon (e.g. $\mathsf{p_2}$). The domain object that refers to this polygon with the role has_area must also be subsumed by the concept federal_state_hh (see the example presented in Figure 7.1). For modeling spatial relations we declare corresponding roles as part of the TBox. The following TBox axioms fulfill the restrictedness criterion because the nested concept terms employ only the $\exists\, \mathsf{f} \,.\, \mathsf{P}$ constructor.

$\textbf{is\_t\_inside} \doteq \exists\, (\mathsf{has\_area})(\mathsf{has\_area}) \,.\, \mathsf{t\_inside}$

$\textbf{hh\_border\_district} \doteq \mathsf{district\_of\_hh} \sqcap \exists\, \mathsf{is\_t\_inside} \,.\, \mathsf{federal\_state\_hh}$

The concept hh_border_district is discussed as an example for the use of the role-forming predicate restriction introduced by is_t_inside. The associated polygon of any individual

that is subsumed by this concept has to be in the t_inside relationship with another polygon that, in turn, is referred to by an instance that is subsumed by the concept federal_state_hh.

While the subsumption relationships discussed above are quite obvious, the advantages of TBox reasoning with spatial relations become apparent if we consider more complex cases, e.g. the following axiom is computed by other (non-DL) components and added to our TBox (e.g. imagine a scenario employing machine learning techniques). The restrictedness criterion is fulfilled.

$$\textbf{unknown} \doteq \mathsf{district\_of\_hh} \sqcap \exists\,(\exists\,(\mathsf{has\_area})(\mathsf{has\_area})\,.\,\mathsf{spatially\_related})\,.\,\mathsf{federal\_state\_hh} \sqcap \exists\,(\exists\,(\mathsf{has\_area})(\mathsf{has\_area})\,.\,\mathsf{touching})\,.\,\mathsf{federal\_state\_sh}$$

If the polygon of district_of_hh touches the polygon of federal_state_sh, then the polygon of district_of_hh is also t_inside the polygon of federal_state_hh. Therefore, it can be proven that unknown is subsumed by hh_border_district (see Appendix A). The spatial constellation defined by the concept unknown could also be characterized as a "Hamburg border district to Schleswig-Holstein." Note however, if district_of_hh had only been defined by the term $\exists\,\mathsf{has\_area}\,.\,\mathsf{g\_inside}_{\mathsf{p}_2}$ (see above), unknown would *not* have been subsumed by hh_border_district because an abstract individual whose associated polygon had been *equal* to p_2 would have been a member of unknown but not a member of hh_border_district.

7.4.2 ABox Reasoning for GIS Applications Concerning Environmental Planning

Environmental information systems can benefit from spatioterminological reasoning in many ways. First, queries can be posed as concepts composed with respect to an ontology underlying a certain TBox. The description logic reasoner will answer the query by finding all instances that are subsumed by the query concept. Second, the ability to test ABox instances for consistency can be used to implement a planning system which is based on hypothesize and test strategies. For instance, let us assume that the following TBox fragment is used to model domain objects shown in the map of Figure 7.2. The restrictedness criterion for TBoxes is fulfilled.

$$\textbf{is\_touching} \doteq \exists\,(\mathsf{has\_area})(\mathsf{has\_area})\,.\,\mathsf{touching}$$

$$\textbf{is\_connected} \doteq \exists\,(\mathsf{has\_area})(\mathsf{has\_area})\,.\,\mathsf{connected}$$

$$\textbf{dangerous\_object} \doteq \mathsf{freeway} \sqcup \mathsf{chemical\_plant} \sqcup \ldots$$

$$\textbf{insecure\_object} \doteq \mathsf{dangerous\_object} \sqcup (\mathsf{unfenced\_object} \sqcap \exists\,\mathsf{is\_connected}\,.\,\mathsf{dangerous\_object})$$

$$\textbf{secure\_playground} \doteq \mathsf{playground} \sqcap \forall\,\mathsf{is\_touching}\,.\,\neg\mathsf{insecure\_object}$$

We suppose that the objects depicted in Figure 7.2 are represented in an ABox as instances of general concepts such as building, region, road etc. These concepts directly model the

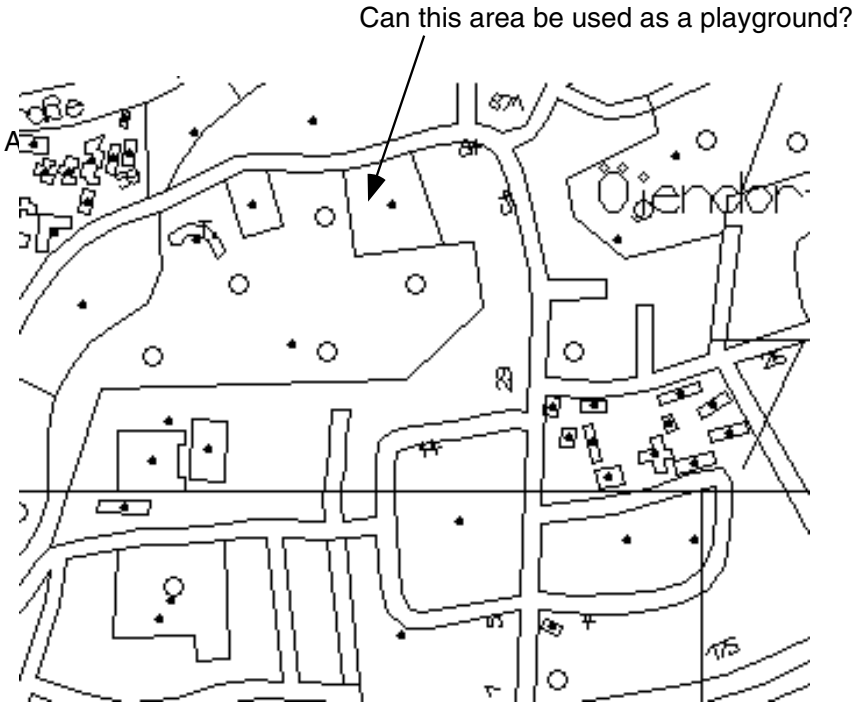


Figure 7.2: A subsection from the Öjendorf map (see text).

information given in the underlying database. The geometry is assumed to be represented by corresponding polygons as fillers for the attribute has_area as required by the ontology underlying our TBox domain model. In order to check whether, for instance, the region area_1 which is indicated by an arrow in Figure 7.2 can be used as a secure playground, we simply add the ABox axiom area_1 : secure_playground.[2] If the description logic reasoner computes that the ABox is consistent, all constraints imposed by secure_playground are satisfied. Hence, according to our (simple) domain model, area_1 might be suitable for a playground. Note that these inferences are correct only if the semantics of spatial relations is adequately considered by the TBox reasoner as described before.

7.5 Summary

Based on the description logic $\mathcal{ALCRP}(\mathcal{D})$, we have shown how spatial and terminological reasoning can be combined in the TBox and ABox. Thus, the fruitful research on description logics has been extended to cope with qualitative spatial relations between spatial objects. A first prototype implementation of $\mathcal{ALCRP}(\mathcal{D})$ is described in [Turhan, 1998]. The $\mathcal{ALCRP}(\mathcal{D})$ approach demonstrates how constraint reasoning and description logics complement each other. The examples presented in this article indicate that conceptual knowledge formalized with a description logic and spatial knowledge concerning topological

[2]ABox statements can be automatically generated by a graphical interface (see Chapter 3 on GenEd).

relations provides solutions for important problems, for instance in GIS applications. An important application of description logic theory in this context is, for instance, schema reasoning for data integration in GIS systems. Defined relations provide a bridge from spatial to conceptual knowledge and support more extensive reasoning services to be exploited for solving practical problems. The next chapter applies the research result on $\mathcal{ALCRP}(\mathcal{D})$ to visual language theory. Using $\mathcal{ALCRP}(\mathcal{D})$ a first semantics for visual spatial queries is sketched.

Part IV

Visual Language Theory Revisited

Chapter 8

Visual Spatial Query Languages: A Semantics Using Description Logic

The previous chapter motivated the integration of spatial reasoning into descriptions logics and presented with $\mathcal{ALCRP}(\mathcal{D})$ a description logic that supports spatial reasoning. This is the reason why $\mathcal{ALCRP}(\mathcal{D})$ is a good starting point for developing a semantics for visual spatial query languages. The use of $\mathcal{ALCRP}(\mathcal{D})$ extends the work described in Chapter 3 where the logic $\mathcal{ALCQ}$ was used for specifying diagrams. The description logic $\mathcal{ALCQ}$ is in some sense more expressive than $\mathcal{ALCRP}(\mathcal{D})$ since it allows *qualified number restrictions* but also less expressive than $\mathcal{ALCRP}(\mathcal{D})$ since it offers no *concrete domains* and no *defined roles*. Furthermore, the approach from Chapter 3 considered spatial relations (e.g. touching, containing, etc.) as uninterpreted (primitive) binary relations with respect to description logic theory (see Section 8.2.2 for a discussion of the problems). In order to correctly deal with spatial objects we needed an external geometric reasoner about the factual world asserting proper spatial relationships and object properties in an ABox. This was a feasible approach but its terminological reasoning about space was incomplete in the sense that the semantics of qualitative spatial relations were not fully captured. With the use of $\mathcal{ALCRP}(\mathcal{D})$ it is now possible to better capture the semantics of spatial relationships using appropriate concrete domains and defined roles.

The work presented in the following is a first treatment that uses $\mathcal{ALCRP}(\mathcal{D})$ as a suitable description logic in order to deal with the semantics of visual *spatial* query languages for geographic information systems. The work is based on [Haarslev, 1998b; Haarslev, 1999; Haarslev et al., 1999d; Haarslev et al., 2000c].

8.1 Introduction

In the context of visual language (VL) theory $\mathcal{ALCRP}(\mathcal{D})$ can be considered as a decidable formalism for spatial and/or temporal knowledge representation if appropriate concrete domains are provided. Suitable candidates for concrete domains can be based on theories

about qualitative spatial reasoning (e.g. Egenhofer's work [Egenhofer, 1991] or RCC8 [Randell et al., 1992]) or about time (e.g. Allen's interval logic [Allen, 1983]). These concrete domains could provide a basis for specifying visual procedural semantics (time) or for specifying qualitative and quantitative (e.g. geometric) relationships about space. However, the following investigation focusses on spatial reasoning.

$\mathcal{ALCRP}(\mathcal{D})$ instantiated with a domain for qualitative spatial reasoning is well suited to deal with truly visual representations. In the scope of this chapter we consider representations as 'truly visual' that exploit geometric and/or depictional constraints permitted by the plane. A good example is the domain of visual spatial query languages for GIS, where query languages usually deal with geographic entities such as lakes, rivers, forests, etc. that are represented as two-dimensional elements in the query language. We motivate the application of this spatial formalism to visual spatial query languages with the spatial query system VISCO presented in the chapters 4 and 5. The formalism can be used to define the semantics of visual spatial queries, to reason about query subsumption, and to deal with multiple worlds or query completion with the help of default reasoning. Examples for these kinds of reasoning are discussed later on in this chapter.

8.2 Modeling in VL Theory with Concrete Domains

Why is the use of description logics with concrete domains so important for VL theory? As already mentioned above the description logic $\mathcal{ALCQ}$ can neither deal with concepts defined with the help of arithmetic nor with roles defined by predicates. For instance, it is not possible to specify in $\mathcal{ALCQ}$ a *defined* concept normal_cottage that represents every conceivable cottage whose space is between 30 m^2 and 70 m^2. It is only possible to rely on an external reasoner that has to assert the concept membership for normal_cottage in the ABox. This is illustrated with the following example where $\mathcal{ALC}$ as a sublanguage of $\mathcal{ALCQ}$ is used to define some 'cottage concepts' as primitives.

small_cottage $\sqsubseteq$ normal_cottage

normal_cottage $\sqsubseteq$ spacious_cottage

spacious_cottage $\sqsubseteq$ cottage $\sqcap \exists$ has_space . $\top$

With these definitions we can declare an ABox containing an individual c_1 which represents a cottage with a floor space of 65 m^2. Suppose an external reasoner asserts for c_1 the concept membership for small_cottage[1]. However, an $\mathcal{ALC}$ ABox reasoner cannot catch the intended "domain contradiction" that the same cottage cannot have both a floor space which is equal to 65 m^2 and less than 30 m^2.

c_1 : cottage, $(c_1, 65)$: has_space, c_1 : small_cottage

[1] A small cottage has a floor space of less than 30 m^2.

An $\mathcal{ALC}$ reasoner correctly classifies this ABox as consistent although one would like to catch this as an invalid entry. This is the motivation behind our notion of *unintended models*. We define a model as *unintended* in a description logic, if the logic cannot fully capture the intended semantics of a particular domain and thus its reasoner correctly classifies a TBox or ABox as coherent that should turn out as incoherent in a more expressive logic.

The deficiency of the logic $\mathcal{ALC}$ to deal with concepts defined with the help of algebra was a primary motivation for the 'concrete domain' approach realized by the description logic $\mathcal{ALC}(\mathcal{D})$. The interesting question for VL theory is "what can be already modeled with $\mathcal{ALC}(\mathcal{D})$ and when and why do we need $\mathcal{ALCRP}(\mathcal{D})$?" In order to answer this question we investigate in the following sections the spatial modeling capabilities of $\mathcal{ALC}(\mathcal{D})$ and $\mathcal{ALCRP}(\mathcal{D})$ in detail.

8.2.1 Reasoning with $\mathcal{ALC}(\mathcal{D})$

$\mathcal{ALC}(\mathcal{D})$ extends the logic $\mathcal{ALC}$ by concrete domains. The six concept-forming operators of $\mathcal{ALCRP}(\mathcal{D})$ (see Definition 7.3) but not the role-forming predicate operator are exactly the language elements of $\mathcal{ALC}(\mathcal{D})$. The basic logic $\mathcal{ALC}$, is a true subset of $\mathcal{ALC}(\mathcal{D})$ that, in turn, is a true subset of $\mathcal{ALCRP}(\mathcal{D})$.

The notion of a *concrete domain* was formally introduced in Definition 7.1. Informally speaking, a concrete domain can be understood as a device providing a bridge between conceptual reasoning with abstract entities and (qualitative) constraint reasoning with concrete or qualitative data. Examples for admissible concrete domains are $\mathcal{R}$ (over the set $\mathbb{R}$ of all real numbers with predicates built by first order means from (in)equalities between integer polynomials in several indeterminates, see [Tarski, 1951]) or $\mathcal{S}_2$ (over the set of all two-dimensional polygons with topological relations from Figure 6.2 as predicates, see also Definition 7.8). The name 'concrete domain' is in some sense misleading since it suggests that a concrete domain realizes reasoning about 'concrete' (e.g. numeric) data. This kind of reasoning is sometimes supported (e.g. in the domain $\mathcal{R}$) but in our application we mainly use concrete domains for reasoning about the satisfiability of finite conjunctions of qualitative predicates. For instance, the domain $\mathcal{S}_2$ qualitatively decides the satisfiability of conjunctions such as $\mathsf{touching}(I_1, I_2) \wedge \mathsf{contains}(I_2, I_3) \wedge \mathsf{touching}(I_1, I_3)$ without any notion for 'concrete' polygons. This is a well-known example for a constraint satisfaction problem.

Without loss of generality we introduce a λ-like notation for anonymous predicates of the domain $\mathcal{R}$. Formally, each anonymous predicate and its negation could be replaced by unique names for the λ-term and its negated counterpart and, moreover, the negation sign in front of a λ-term can be safely moved inside of this term.

The fact that $\mathcal{ALC}(\mathcal{D})$ (and $\mathcal{ALCRP}(\mathcal{D})$) can be parameterized by only one concrete domain is not a limitation because in [Haarslev et al., 1999b] it is shown that the union of two admissible concrete domains again results in an admissible concrete domain. For instance, the union of the concrete domains $\mathcal{R}$ and $\mathcal{S}_2$ is a useful domain providing tools for

expressing both spatial and arithmetic constraints. This domain is used in the remainder of this chapter.

Let us reconsider the previous cottage example using $\mathcal{ALC}(\mathcal{D})$. We now define the 'cottage concepts' with concrete domain predicates expressing the appropriate constraints for the floor space.

$\textbf{small\_cottage} \doteq \mathsf{cottage} \sqcap \exists\, \mathsf{has\_space} \,.\, \lambda_{\mathcal{R}} x \,.\, (x < 30)$

$\textbf{normal\_cottage} \doteq \mathsf{cottage} \sqcap \exists\, \mathsf{has\_space} \,.\, \lambda_{\mathcal{R}} x \,.\, (x < 70)$

$\textbf{spacious\_cottage} \doteq \mathsf{cottage} \sqcap \exists\, \mathsf{has\_space} \,.\, \lambda_{\mathcal{R}} x \,.\, (x < 200)$

A reasoner for $\mathcal{ALC}(\mathcal{D})$ immediately recognizes the subsumption relationship between these concepts, i.e. spacious_cottage subsumes normal_cottage that, in turn, subsumes small_cottage. Using the following ABox assertions, an $\mathcal{ALC}(\mathcal{D})$ reasoner recognizes the individual c_1 as a member of spacious_cottage and c_2 as a member of both spacious_cottage and normal_cottage.

$\mathsf{c}_1 : \mathsf{cottage},\ (\mathsf{c}_1, 80) : \mathsf{has\_space}$

$\mathsf{c}_2 : \mathsf{cottage},\ (\mathsf{c}_2, 60) : \mathsf{has\_space}$

Let us consider an alternative definition for the concepts small_cottage, normal_cottage, and spacious_cottage, which are part of our GIS scenario.

$\textbf{cottage} \sqsubseteq \mathsf{building}$

$\textbf{small\_cottage} \doteq \mathsf{cottage} \sqcap \exists\, \mathsf{has\_space} \,.\, \lambda_{\mathcal{R}} x \,.\, (x < 30)$

$\textbf{normal\_cottage} \doteq \mathsf{cottage} \sqcap \exists\, \mathsf{has\_space} \,.\, \lambda_{\mathcal{R}} x \,.\, (x \geq 30 \wedge x < 70)$

$\textbf{spacious\_cottage} \doteq \mathsf{cottage} \sqcap \exists\, \mathsf{has\_space} \,.\, \lambda_{\mathcal{R}} x \,.\, (x \geq 70)$

The first axiom defines a cottage as a specialization of a building. The next three concepts define cottages of various sizes using the predicate exists restriction. These three definitions are mutually exclusive due to their size-restricting predicates.

8.2.2 Unintended Models in $\mathcal{ALC}(\mathcal{D})$

Characteristics of visual languages very often depend on spatial relationships between language elements. For instance, the topological relations used by GenEd (see Chapter 3) cannot be adequately defined in $\mathcal{ALC}(\mathcal{D})$. With $\mathcal{ALC}(\mathcal{D})$ one can only define concepts describing individuals which are dependent on properties reachable via feature chains originating from a single individual and expressed with concrete predicates. It is not possible to define roles specified by predicates that express relationships between individuals. Therefore, TBox and ABox reasoning in $\mathcal{ALC}(\mathcal{D})$ is *spatially incomplete*[2] for the intended semantics of spatial relations.

[2]We use the informal term *spatially incomplete* for referring to *unintended models* that are allowed in $\mathcal{ALC}(\mathcal{D})$ but not intended by the spatial domain. Of course, TBox and ABox reasoning in $\mathcal{ALC}(\mathcal{D})$ is sound and complete with respect to the semantics of $\mathcal{ALC}(\mathcal{D})$.

Using the concrete domain $\mathcal{R} \cup \mathcal{S}_2$ we illustrate the spatial incompleteness of $\mathcal{ALC}(\mathcal{D})$ with an example demonstrating unintended models. As an ontological commitment we use a feature has_area that may have a spatial region as concrete filler describing the occupied space of an individual. The unintended models are possible because $\mathcal{ALC}(\mathcal{D})$ cannot appropriately capture the semantics of spatial relations. Note that in contrast to the following sections we have to consider the roles is_touching, is_connected, and is_g_inside as primitive in this subsection since $\mathcal{ALC}(\mathcal{D})$ has no means to express defined roles, e.g. $\mathsf{has\_area}(\mathsf{i}_1, \mathsf{r}_1) \wedge \mathsf{has\_area}(\mathsf{i}_2, \mathsf{r}_2) \wedge \mathsf{touching}(\mathsf{r}_1, \mathsf{r}_2) \not\Rightarrow \mathsf{is\_touching}(\mathsf{i}_1, \mathsf{i}_2)$.

fishing_cottage $\doteq \mathsf{cottage} \sqcap \exists\, \mathsf{is\_touching} \,.\, \mathsf{river}$

mosquito_free_forest $\doteq \mathsf{forest} \sqcap \forall\, \mathsf{is\_connected} \,.\, \neg\mathsf{river}$

paradise_cottage $\doteq \mathsf{fishing\_cottage} \sqcap \exists\, \mathsf{is\_g\_inside} \,.\, \mathsf{forest} \sqcap \forall\, \mathsf{is\_g\_inside} \,.\, \mathsf{mosquito\_free\_forest}$

We define a paradise cottage as a fishing cottage located in a mosquito-free forest, i.e. the forest is not spatially connected with a river. However, a fishing cottage is defined as a cottage that touches a river. It follows that the forest containing a fishing cottage must also be spatially connected with this river. Obviously, the paradise cottage is only a dream that cannot exist in the real world. This is due to the intended semantics of the underlying spatial relations:

> A situation where a region r_1 (cottage) is *g_inside* another region r_2 (forest) and this region r_1 is also *touching* a third region r_3 (river) implies that r_2 is *connected* to r_3, i.e. $\mathsf{g\_inside}(\mathsf{r}_1, \mathsf{r}_2) \wedge \mathsf{touching}(\mathsf{r}_1, \mathsf{r}_3) \Rightarrow \mathsf{connected}(\mathsf{r}_2, \mathsf{r}_3)$.

Thus, the concept paradise_cottage should be recognized as incoherent. This is not possible in $\mathcal{ALC}(\mathcal{D})$ due to the absence of defined roles capturing the semantics of the spatial predicates. With $\mathcal{ALC}(\mathcal{D})$ these roles can only be defined as primitive, i.e. they cannot interact with one another.

This deficiency also holds for ABox reasoning. The following assertions describe a spatial constellation in correspondence with the TBox defined above.

$\mathsf{c} : \mathsf{normal\_cottage},\ (\mathsf{c}, 60) : \mathsf{has\_space},\ (\mathsf{c}, \mathsf{S}_\mathsf{c}) : \mathsf{has\_area}$

$\mathsf{r} : \mathsf{river},\ (\mathsf{r}, \mathsf{S}_\mathsf{r}) : \mathsf{has\_area},\ (\mathsf{c}, \mathsf{r}) : \mathsf{is\_touching}$

$\mathsf{f} : \mathsf{forest},\ (\mathsf{f}, \mathsf{S}_\mathsf{f}) : \mathsf{has\_area},\ (\mathsf{c}, \mathsf{f}) : \mathsf{is\_g\_inside}$

If we pose the query "Is the forest f spatially connected with the river r?" using the TBox and ABox declarations as defined above, an $\mathcal{ALC}(\mathcal{D})$ reasoner correctly answers *no*. Even adding the assertion

$\mathsf{f} : \exists\, \mathsf{is\_touching} \,.\, \mathsf{river}$

to the previous ABox would *not* cause a contradiction for the individual f because the reasoner has no knowledge about the interaction between the roles is_touching and is_connected.

8.2.3 Reasoning with $\mathcal{ALCRP}(\mathcal{D})$

A solution for this 'unintended model' problem is possible in $\mathcal{ALCRP}(\mathcal{D})$ with *defined* roles using the *role-forming predicate restriction*. With this operator one can define roles that properly reflect spatial relationships. The role-forming predicate operator of $\mathcal{ALCRP}(\mathcal{D})$ significantly extends the expressivity of the language. Moreover, the possible union of concrete domains enhances $\mathcal{ALCRP}(\mathcal{D})$'s expressiveness due to the role-forming predicate restriction that can be used to relate pairs of individuals in a way that is not possible in $\mathcal{ALC}(\mathcal{D})$ (see also [Haarslev et al., 1999b]). This is in contrast to $\mathcal{ALC}(\mathcal{D})$ where the union of two concrete domains can be reduced to syntactic transformations.

In the following we demonstrate with the same scenario as above that these unintended models cannot exist with $\mathcal{ALCRP}(\mathcal{D})$. Note that the roles is_g_inside et cetera are now defined via spatial predicates.

cottage $\sqsubseteq$ building

small_cottage $\doteq$ cottage $\sqcap \exists$ has_space . $\lambda_{\mathcal{R}} x$. $(x < 30)$

normal_cottage $\doteq$ cottage $\sqcap \exists$ has_space . $\lambda_{\mathcal{R}} x$. $(x \geq 30 \wedge x < 70)$

spacious_cottage $\doteq$ cottage $\sqcap \exists$ has_space . $\lambda_{\mathcal{R}} x$. $(x \geq 70)$

is_g_inside $\doteq \exists$ (has_area)(has_area) . g_inside

cottage_in_forest $\doteq$ cottage $\sqcap \exists$ is_g_inside . forest

is_touching $\doteq \exists$ (has_area)(has_area) . touching

fishing_cottage $\doteq$ cottage $\sqcap \exists$ is_touching . river

is_connected $\doteq \exists$ (has_area)(has_area) . connected

mosquito_free_forest $\doteq$ forest $\sqcap \forall$ is_connected . $\neg$river

paradise_cottage $\doteq$ fishing_cottage $\sqcap \exists$ is_g_inside . forest $\sqcap \forall$ is_g_inside . mosquito_free_forest

A reasoner for $\mathcal{ALCRP}(\mathcal{D})$ will recognize that paradise_cottage is incoherent, i.e. no individual can ever be a member of this concept. This is due to the spatial inference that a mosquito-free forest has to be connected with a river since it contains a cottage that is touching a river (see also the previous section). However, the definition of a mosquito-free forest requires that anything that is spatially connected to the forest must not be a river. This is an obvious contradiction in the TBox.

c : cottage, $(\mathsf{c}, 60)$: has_space, $(\mathsf{c}, \mathsf{S_c})$: has_area

r : river, $(\mathsf{r}, \mathsf{S_r})$: has_area, $(\mathsf{S_c}, \mathsf{S_r})$: touching

f : forest, $(\mathsf{f}, \mathsf{S_f})$: has_area, $(\mathsf{S_c}, \mathsf{S_f})$: g_inside

Using an ABox as shown above, the reasoner will also correctly answer the query "Is the forest f spatially connected with the river r?" with *yes*, i.e. adding the assertion

$\mathsf{f} : \neg\exists\, \mathsf{is\_connected} \,.\, \mathsf{river}$

to the following ABox would cause a contradiction for f and thus validate the query. The contradiction with the negated query can be explained with the same spatial inference as the one used in the TBox. Adding for the individual f the assertion $\mathsf{f} : \neg\exists\, \mathsf{is\_connected} \,.\, \mathsf{river}$ (that is equivalent to $\mathsf{f} : \forall\, \mathsf{is\_connected} \,.\, \neg\mathsf{river}$) will cause a clash with the entailed relationship $(\mathsf{f}, \mathsf{r}) : \mathsf{is\_connected}$ since r is asserted as a member of river.

We would like to emphasize that no assertions about role memberships for is_g_inside, is_connected, and is_touching have to be supplied. These relationships are automatically established by an $\mathcal{ALCRP}(\mathcal{D})$ reasoner with respect to the fillers of has_area.

8.3 Semantics of Spatial Queries

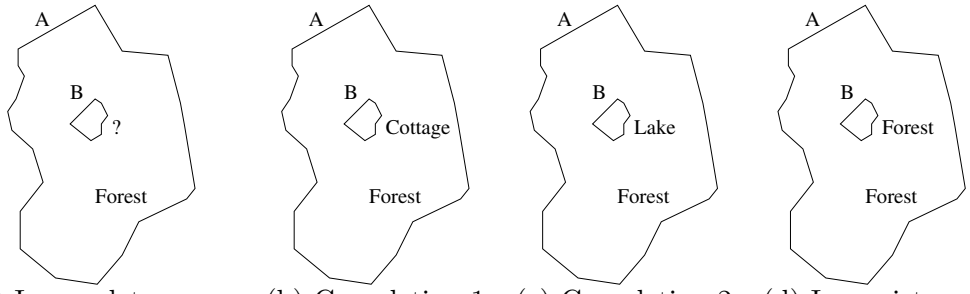
The previous sections motivated the development of the description logic $\mathcal{ALCRP}(\mathcal{D})$ and demonstrated its usefulness for spatial reasoning with visual representations. We introduced semantic entities such as buildings, cottages, forests, rivers, etc. These entities are suitable candidates for elements of visual spatial query languages. This is motivated by the development of the VISCO system. In VISCO we assume that basic map objects are predefined in a GIS. Furthermore, spatial areas are defined by polygons. Map elements (e.g. polylines, polygons) are annotated with labels such as "forest", "building", "river" etc. that directly correspond to the semantic entities characterized above.

We imagine a VISCO application scenario for querying a GIS as follows. Instead of textually writing a complicated SQL query, a user simply draws a constellation of spatial entities which resemble the intended constellation of interest. Using the basic vocabulary provided by the GIS, the user has to annotate drawing elements by concept names (e.g. this polygon represents a forest). The parser of VISCO would analyze the drawing and create a corresponding ABox as semantic representation. Thus, the *semantics* of a query is defined by an ABox derived from a spatial constellation.

Sometimes it might be hard for users to fully specify a query. For instance, in VISCO it is possible to select the meaning of a query element from a list containing hundreds predefined concept descriptions. However, it might be possible to dramatically reduce the number of choices, if additional knowledge about the query is known. Therefore, a completion facility is needed to resolve semantic ambiguities or to complete underspecified information by using default rules for further specialization. The next subsections describe the usefulness of spatial default reasoning and the query processing and reasoning process.

8.3.1 Completion of Queries

Default knowledge is used to make queries complete in the absence of precise information, if it can be applied in a consistent way. For the sake of brevity we do not discuss the



(a) Incomplete query (b) Completion 1 (c) Completion 2 (d) Inconsistency

Figure 8.1: Automatic completion of visual queries by application of default rules.

formal representation of default knowledge and its rules of inference. The integration of default and terminological reasoning is discussed in [Baader and Hollunder, 1995a; Baader and Hollunder, 1995b]. Its application to spatial reasoning with $\mathcal{ALCRP}(\mathcal{D})$ is formally analyzed in [Möller and Wessel, 1999].

In order to analyze the modeling problems in this context, we begin with a more detailed discussion of a visual query example. Let us assume, a person is interested in buying a cottage located in a forest. In Figure 8.1(a) the user just started to formulate the query. After (s)he has specified that the type of the surrounding polygon A should be a forest, the type of the small polygon B must be specified. A smart interface should use formal derivation processes for computing plausible candidates for object "type" specifications. For narrowing the set of possibilities we assume that two default rules are applicable: one says the interior small polygon B could be a cottage (Figure 8.1(b)) and another one states that B could be a lake (Figure 8.1(c)) if this does not lead to inconsistencies. Since an object can be either a lake or a cottage, there is no way to believe in both possibilities at the same time. This kind of default rule interaction is a simple example demonstrating the necessity of considering different *possible worlds* which must be maintained by the reasoning system. Depending on the default rule being used to conclude new knowledge, different subsequent conclusions might be possible.

Other potentially active default rules might produce only inconsistencies with the set of current assertions. For instance, if a default rule is applied which says the small polygon B is also a forest (Figure 8.1(d)), we will get a contradiction if an axiom (as part of our conceptual background knowledge) states that a forest can never contain another forest. Thus, in our query context, the latter default cannot be applied and, as a consequence of computing and appropriately interpreting the set of possible worlds, we can compose a situation-adapted menu for the graphical user interface and the user can select between meaningful concepts for object B. In our specific example, the menu will contain items for cottage and lake but not for forest.

If more than one possible world is computed, an intuitive criterion would be to select the world originating from a default with the more specific precondition or conclusion. For instance, in the query shown in Figure 8.2(a) we would prefer a default concluding that

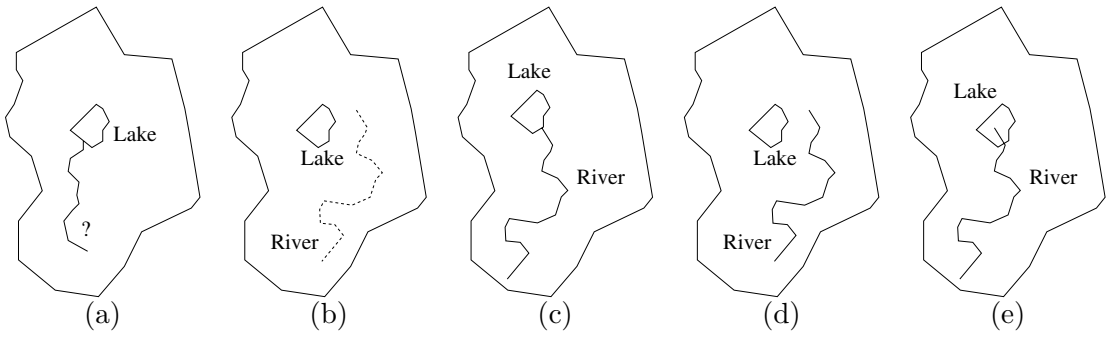


Figure 8.2: Scenarios for situation-adapted completion of queries (see text).

the thin graphical object might be a 'river flowing into a lake' (which might be a useful concept in our scenario) instead of a more general default concluding only that the object is an ordinary river.

The automatic augmentation of visual queries by conclusions of applied default rules can be seen as a specialization process. Therefore, this process might not only be useful during the construction of a visual query, but also useful as a tool for query refinement after a query has been executed which returned too many results. In our GIS context we also have to consider how to extend knowledge about spatial relationships between domain objects.

In the context of sketch-based visual querying, on the one hand it is sometimes useful to leave some spatial relations between graphical objects unspecified because they are unknown or simply because the user is not willing to specify them. On the other hand, in order to actually draw a picture, the user *must* specify each spatial relation, even if it is just one of several possible (base) relations. The problem of how to specify "don't care relations" or "example relations" is well known and inherent in diagrammatic representations. It is similar to the problem of displaying visual disjunctions.

For example, in the query shown in Figure 8.2b, we have a visible disjoint relation between the river and the lake. If we intended the river to be disjoint from the lake, the query answering system would not find any rivers flowing into this lake. The problem is to specify that the river should be strictly inside the forest, but to leave the relation to the lake unspecified. As a possible solution to this problem, we could simply *ignore* each visible disjoint relation. But, with this interpretation, we can now no longer state a query searching for rivers *not* flowing into this specific lake, which might be a very useful concept. We propose the following solution. For objects like the river that are drawn with a specific drawing attribute such as dashing, the universal spatial relation to other objects (disjunction of all base relations) is asserted. Dashed objects introduce no spatial query constraints. However, in some cases this would usually not match the users intention because the answer set of the query will be too large. With the help of default knowledge we can automatically refine the query in a way that is appropriate according to the semantics of the objects involved in a query. So, we can guide the interpretation of spatial aspects by

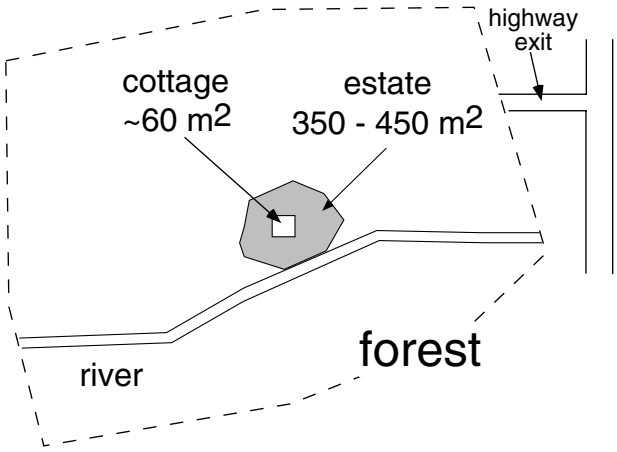


Figure 8.3: Spatial sketch for first query.

the help of conceptual background knowledge and by the application of defaults, returning different hypotheses as possible worlds. A river flows into a lake or not, i.e. graphically both objects are either touching (see also Figure 8.2c) or they are disjoint (see Figure 8.2d). With respect to a lake, there are no other possibilities. In our world model a river never overlaps with a lake (see also Figure 8.2e). This is assumed to be stated as an axiom as part of our general conceptual background knowledge. Besides defaults involving concept constraints we also have to take care of default rules with conclusions yielding new *relation* constraints. For instance, one rule could conclude the relationship touching and the other one the relationship disjoint (see [Möller and Wessel, 1999] for a discussion).

8.3.2 Reasoning about Visual Spatial Queries

The applicability of $\mathcal{ALCRP}(\mathcal{D})$ is demonstrated by extending the GIS example. We imagine a scenario for a GIS query where somebody is planning to acquire a nice cottage. We assume the existence of a GIS offering information about suitable areas in the countryside. The buyer intends to use the cottage for weekends and short holidays. However, the potential buyer is living and working in a major city and is only interested in real estate that can be conveniently reached via a highway. A short travel distance by car from a highway exit to the cottage is the first precondition. Furthermore, the cottage should be located in a forest with a river in the immediate vicinity. The buyer and its family also want a cottage that provides at least 75 m^2 floor space. The estate itself should have about 400 m^2. Having these requirements in mind a query (see Figure 8.3) is sketched reflecting the topological and geometric constraints.[3] A parser can translate the sketch into a semantically equivalent description using a GIS taxonomy containing concept descriptions for the spatial vocabulary of this domain. The following ABox $\mathcal{A}_0$ is derived from Figure 8.3.

[3]We are aware of the scaling problems with drawings and offer first solutions with VISCO's query language. However, in this chapter we deliberately ignore these problems.

$\mathsf{c} : \mathsf{cottage} \sqcap \exists\, \mathsf{has\_space} \,.\, \lambda_{\mathcal{R}} x \,.\, (x > 75),\ (\mathsf{c}, \mathsf{e}) : \mathsf{is\_g\_inside}$

$\mathsf{e} : \mathsf{estate} \sqcap \exists\, \mathsf{has\_space} \,.\, \lambda_{\mathcal{R}} x \,.\, (x > 350 \wedge x < 450)$

$\mathsf{r} : \mathsf{river},\ (\mathsf{r}, \mathsf{e}) : \mathsf{is\_touching}$

$\mathsf{f} : \mathsf{forest},\ (\mathsf{e}, \mathsf{f}) : \mathsf{is\_g\_inside}$

$\mathsf{h} : \mathsf{highway\_exit},\ (\mathsf{h}, \mathsf{f}) : \mathsf{is\_touching}$

We use concept and role expressions as defined in the previous TBoxes. The cottage is described by the individual c with a predicate-exists restriction asserting a floor space of more than 75 m^2. The cottage c has to be inside of an estate with a size between 350 and 450 m^2. As a simplification we assume that the river r has to touch the estate e that is inside of a forest f. The short driving distance from the highway exit h to the cottage c is represented by the condition that h has to touch the borderline of the forest f.[4] For sake of simplicity we deliberately abstracted away the distance constraints. Of course, it is also possible to express these constraints with the domain $\mathcal{R}$.

Additionally, we assume the following new or revised concept definitions ($\top_{\mathcal{R}}$ names the predicate required for testing the membership in the domain $\mathcal{R}$ for a concrete individual x, see also Definition 7.3).

$\mathbf{estate} \sqsubseteq \mathsf{spatial\_area} \sqcap \exists\, \mathsf{has\_space} \,.\, \lambda_{\mathcal{R}} x \,.\, (\top_{\mathcal{R}}(x))$

$\mathbf{estate\_in\_forest} \doteq \mathsf{estate} \sqcap \exists\, \mathsf{is\_g\_inside} \,.\, \mathsf{forest}$

$\mathbf{cottage\_in\_forest} \doteq \mathsf{cottage} \sqcap \exists\, \mathsf{is\_g\_inside} \,.\, \mathsf{estate\_in\_forest}$

$\mathbf{fishing\_cottage} \doteq \mathsf{cottage} \sqcap \exists\, \mathsf{is\_g\_inside} \,.\, (\mathsf{estate} \sqcap \exists\, \mathsf{is\_touching} \,.\, \mathsf{river})$

The realizing component of the $\mathcal{ALCRP}(\mathcal{D})$ reasoner will compute the following most specific subsuming concepts (also referred to as *parents*) of the cottage c: expensive_cottage and fishing_cottage. The parents of the estate e will be estate_in_forest. The other individuals r, f, h keep their asserted concepts as parents.

With the help of an abstraction process (e.g. see [Hollunder, 1994]) we can replace Abox $\mathcal{A}_0$ by an Abox $\mathcal{A}_1$ containing a single assertion for c with the synthesized[5] concept description $\mathsf{cottage}_{\mathsf{c}_1}$. The other two concept definitions are only used to enhance the readability of $\mathsf{cottage}_{\mathsf{c}_1}$.

[4] This is a simplification again since the extent of a forest can easily cause a long driving distance.

[5] The name of a synthesized concept description contains as index the individual name from which the description was derived. In order to distinguish between several abstractions, the individual itself is also indexed.

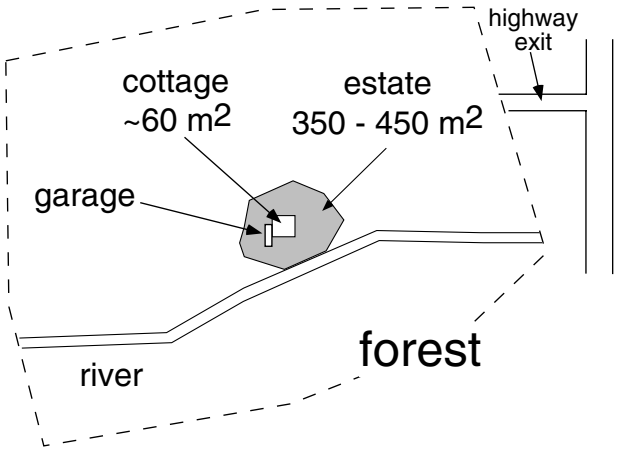


Figure 8.4: Spatial sketch for second query.

$\mathbf{forest_{f_1}} \doteq \mathsf{forest} \sqcap \exists\, \mathsf{is\_touching} \,.\, \mathsf{highway\_exit}$

$\mathbf{estate_{e_1}} \doteq \mathsf{estate} \sqcap \exists\, \mathsf{has\_space} \,.\, \lambda_{\mathcal{R}} x \,.\, (x > 350 \wedge x < 450) \sqcap$
$\exists\, \mathsf{is\_g\_inside} \,.\, \mathsf{forest_{f_1}} \sqcap \exists\, \mathsf{is\_touching} \,.\, \mathsf{river}$

$\mathbf{cottage_{c_1}} \doteq \mathsf{cottage} \sqcap \exists\, \mathsf{has\_space} \,.\, \lambda_{\mathcal{R}} x \,.\, (x > 75) \sqcap \exists\, \mathsf{is\_g\_inside} \,.\, \mathsf{estate_{e_1}}$

The revised ABox $\mathcal{A}_1$ now consists only of the assertion $\mathsf{c} : \mathsf{cottage_{c_1}}$. The newly created concept $\mathsf{cottage_{c_1}}$ is classified by the reasoner and integrated into the concept taxonomy. The semantic validity of this query is automatically verified during classification, i.e. to check whether the concept is coherent. For instance, if the forest f were required to be 'mosquito-free' (see above), the $\mathcal{ALCRP}(\mathcal{D})$ reasoner would immediately recognize the incoherence of $\mathsf{cottage_{c_1}}$. This information could be used by the spatial parser for generating an explanation to the user and for identifying the source of the contradiction.

Let us assume that the executed query $\mathsf{c} : \mathsf{cottage_{c_1}}$ returns more than 100 matches. The next step for the user might be to refine the query by adding more constraints.[6] One could add more requirements to the estate, e.g. we ask for a garage connected to the cottage. The extended sketch (see Figure 8.4) corresponds to the ABox $\mathcal{A}_2$ that results from adding the following new assertions to ABox $\mathcal{A}_0$.

$\mathsf{g} : \mathsf{garage},\ (\mathsf{c}, \mathsf{g}) : \mathsf{is\_touching}$

The abstraction process reduces ABox $\mathcal{A}_2$ to ABox $\mathcal{A}_3$ consisting only of the assertion $\mathsf{c} : \mathsf{cottage_{c_2}}$ using the following synthesized concept description.

$\mathbf{cottage_{c_2}} \doteq \mathsf{cottage} \sqcap \exists\, \mathsf{has\_space} \,.\, \lambda_{\mathcal{R}} x \,.\, (x > 75) \sqcap \exists\, \mathsf{is\_g\_inside} \,.\, \mathsf{estate_{e_1}} \sqcap$
$\exists\, \mathsf{is\_touching} \,.\, \mathsf{garage}$

[6] Of course, one of the most important criteria is the price of the estate. This is neglected due to the non-spatial nature of this part of the query.

The $\mathcal{ALCRP}(\mathcal{D})$ reasoner recognizes the relationship in the taxonomy that $\mathsf{cottage}_{c_1}$ subsumes $\mathsf{cottage}_{c_2}$. It can be rewritten as $\mathsf{cottage}_{c_3}$ that even textually demonstrates the subsumption relationship.

$$\mathbf{cottage}_{c_3} \doteq \mathsf{cottage}_{c_1} \sqcap \exists\, \mathsf{is\_touching} \,.\, \mathsf{garage}$$

For executing the refined query the optimizer can benefit from the detected query subsumption and reduce the search space to the set of query matches already computed for ABox $\mathcal{A}_1$. Note that these query matches are members of the concept $\mathsf{cottage}_{c_1}$. This type of query optimization is an important aspect in applying description logics to database theory (see [Borgida, 1995] for an introduction to these topics).

The benefits of computing a concept subsumption taxonomy can be even more subtle. Imagine a query from another user looking for a cottage located in a forest that is connected to a river. The ABox $\mathcal{A}_4$ derived from the sketch might be structured as follows.

$$\mathsf{c} : \mathsf{cottage},\ \mathsf{e} : \mathsf{estate\_area},\ (\mathsf{c}, \mathsf{e}) : \mathsf{is\_g\_inside}$$
$$\mathsf{r} : \mathsf{river},\ \mathsf{f} : \mathsf{forest},\ (\mathsf{f}, \mathsf{r}) : \mathsf{is\_connected},\ (\mathsf{e}, \mathsf{f}) : \mathsf{is\_g\_inside}$$

The abstraction process creates the following concept definitions.

$$\mathbf{forest}_{f_2} \doteq \mathsf{forest} \sqcap \exists\, \mathsf{is\_connected} \,.\, \mathsf{river}$$
$$\mathbf{estate}_{e_2} \doteq \mathsf{estate} \sqcap \exists\, \mathsf{is\_g\_inside} \,.\, \mathsf{forest}_{f_2}$$
$$\mathbf{cottage}_{c_4} \doteq \mathsf{cottage} \sqcap \exists\, \mathsf{is\_g\_inside} \,.\, \mathsf{estate}_{e_2}$$

The resulting ABox $\mathcal{A}_4$ consists only of the assertion $\mathsf{c} : \mathsf{cottage}_{c_4}$. It turns out that the concept $\mathsf{cottage}_{c_4}$ subsumes the other concepts $\mathsf{cottage}_{c_i}$ although the concept descriptions are textually different. This is a rather complex proof also based on the spatial inference already explained above: $\mathsf{g\_inside}(\mathsf{e}, \mathsf{f}) \wedge \mathsf{touching}(\mathsf{e}, \mathsf{r}) \Rightarrow \mathsf{connected}(\mathsf{f}, \mathsf{r})$.

The abstraction process is applicable to ABoxes containing no joins or cycles. If joins or cycles are present in an ABox, i.e. the same individual is a filler of several roles or even related to itself through a cycle of role assertions, it depends on the expressiveness of the description logic whether an ABox can be reduced to a single concept membership assertion. For instance, joins can be expressed by restricting the number of possible role fillers or by equality restrictions for feature fillers. As mentioned above, other DLs also support the definition of cyclic concepts that might be required to fully reduce some ABoxes. Due to unknown decidability results $\mathcal{ALCRP}(\mathcal{D})$ currently does not allow cyclic concepts or number restrictions. Therefore, in case of ABoxes with joins or cycles, we can only partially reduce these ABoxes. This is illustrated in Figure 8.5 by adding a lake. The river has to flow into the lake and the same lake is touching the forest. This is an example for a join in a corresponding ABox. However, the reasoning with $\mathcal{ALCRP}(\mathcal{D})$ as described above is still valid and usable for query processing. Only the subsumption between ABox queries requires a more sophisticated approach.

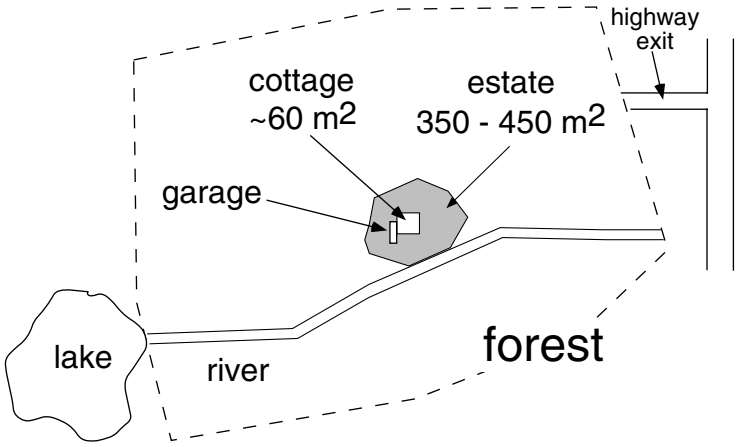


Figure 8.5: Spatial sketch for second query refined by an additional lake.

8.4 Using ABox Patterns for n-ary Queries

We have demonstrated that the abstraction process of rewriting a query ABox to a concept term provides a means to specify the semantics of a visual query. Unfortunately, there are also some drawbacks with this approach. First of all, since the semantics of a concept description is only the set of individuals that satisfy the concept, no n-ary query results can be returned. This is not surprising, since concept descriptions correspond to first-order formulae with only one free variable. Thus, the abstraction process is only successfully applicable to ABoxes where it can select exactly one primary "target individual" from the query. The target individual remains as the single individual and the other individuals are represented by the query concept derived by the abstraction process. In the case of the cottage example, the target object of the query is the cottage the user is looking for. However, considering VISCO, the semantics of a VISCO query is a set of n-tuples, so one would need more than one free variable to fully specify the semantics of VISCO's query language which can handle aggregates.

As a solution to this problem we have developed so-called *ABox patterns*, which are ordinary $\mathcal{ALCRP}(\mathcal{S}_2)$ ABoxes that may –in addition to ordinary individuals– contain *variables*, e.g. x?, y?. Intuitively speaking, given a "database ABox" $\mathcal{A}$ and a query ABox $\mathcal{Q}$ that contains variables, the *ABox pattern retrieval service* returns a set σ of *substitutions* which are mappings from variables in $\mathcal{Q}$ to ABox individuals in the database ABox $\mathcal{A}$ such that $\mathcal{A} \models_{\mathcal{T}} \sigma(\mathcal{Q})$ (w.r.t. the TBox $\mathcal{T}$).

$abox\_pattern\_retrieval(\mathcal{A}, \mathcal{Q}) := \{\sigma \mid \mathcal{A} \models_{\mathcal{T}} \sigma(\mathcal{Q})\}$.

$\sigma(Q)$ applies the given substitution σ to the query ABox $\mathcal{Q}$, replacing its variables with individuals from $\mathcal{A}$. As a subproblem, we need to decide the *ABox entailment problem*, e.g. the question whether $\mathcal{A} \models_{\mathcal{T}} \sigma(\mathcal{Q})$. This problem is decidable for $\mathcal{ALCRP}(\mathcal{S}_2)$ (see [Möller and Wessel, 1999]). The substitutions can simply be enumerated as mappings from $Vars(\mathcal{Q}) \rightarrow Individuals(\mathcal{Q})^{\| Vars(\mathcal{Q}) \|}$ and applied to $\mathcal{Q}$ yielding $\sigma(\mathcal{Q})$. If $\sigma(\mathcal{Q})$ is entailed by $\mathcal{A}$, then the range (image) of σ is the n-ary query result. Note that computing the

set of individuals which are members of a concept C is a special case of an ABox pattern retrieval.

$concept\_members(\mathcal{A}, \mathsf{C}) := abox\_pattern\_retrieval(\mathcal{A}, \{\mathsf{x?:C}\})$.

As an example, we reconsider the query where a user is looking for a cottage in a forest. Obviously, the price of the estate as well as the cottage and the forest itself are of interest, so instead of returning just the cottage the query should return triplets such as $\langle \mathsf{cottage}, \mathsf{estate}, \mathsf{forest} \rangle$ which can be further inspected. The corresponding query ABox $\mathcal{Q}$ might be defined as

$\mathcal{Q} = \{\mathsf{x?:cottage}, \mathsf{y?:estate}, \mathsf{z?:forest}, (\mathsf{x?}, \mathsf{y?}):\mathsf{g\_inside}, (\mathsf{y?}, \mathsf{z?}):\mathsf{g\_inside}\}$.

The user can also refer to specific individuals, e.g.

$\mathcal{Q} = \{\mathsf{x?:cottage}, \mathsf{y?:estate}, (\mathsf{x?}, \mathsf{y?}):\mathsf{g\_inside}, (\mathsf{y?}, \mathsf{black\_forest}):\mathsf{g\_inside}\}$, where $\mathsf{black\_forest}$ is a specific database ABox individual. Obviously, joins can easily be specified. Suppose we are looking for two cottages within the same estate that are located at the same river:

$\mathcal{Q} = \{\mathsf{x?:cottage}, \mathsf{y?:cottage}, \mathsf{x?} \neq \mathsf{y?}, \mathsf{z?:estate}, (\mathsf{x?}, \mathsf{z?}):\mathsf{g\_inside},$
$(\mathsf{y?}, \mathsf{z?}):\mathsf{g\_inside}, \mathsf{r?:river}, (\mathsf{x?}, \mathsf{r?}):\mathsf{touching}, (\mathsf{y?}, \mathsf{r?}):\mathsf{touching}\}$.

Since the unique name assumption also does not hold for variables, we introduce an additional assertion $x? \neq y?$ in order to ensure that $\sigma(x?) \neq \sigma(y?)$. This simply constrains the substitution σ and has no impact on $\mathcal{ALCRP}(\mathcal{S}_2)$.

8.5 Related Work

The experience with VISCO (see Chapter 4 and 5) has motivated the research presented in this chapter. VISCO can be classified as a visual query system for spatial information systems that uses 'sketched' queries combined with deductive reasoning. An operational semantics for VISCO can be found in [Wessel, 1998]. However, this chapter presented a proposal defining a descriptive semantics using description logics. A survey on visual query systems for database systems handling conventional data can be found in [Catarci et al., 1997]. Other relevant work [Meyer, 1994; Egenhofer, 1997] reviews especially visual query system for spatial information systems. A related approach that also uses spatial relations [Del Bimbo et al., 1994] deals with symbolic descriptions and retrieval in image databases. Another approach deals with pictorial query specifications for spatially referenced image databases [Soffer and Samet, 1998]. We refer to Section 4.3 for a review of the four approaches (see [Meyer, 1994; Calcinelli and Mainguenaud, 1994; Lee and Chin, 1995; Egenhofer, 1997]) which come closest to the ideas and concepts behind VISCO and to Section 3.6 for a review of other approaches related to VL theory.

[Meyer, 1994] also gives formal semantics for visual spatial queries using a mapping to the Datalog language, but there exists no formalization of topological relations or conceptual knowledge. To the best of our knowledge there exists no other approach or (visual) spatial query language addressing the semantics of spatial queries and their subsumption using a spatial logic such as $\mathcal{ALCRP}(\mathcal{D})$.

Meyer's recent work [Meyer, 1997] presents *picture logic*, a visual language for the specification of diagrams and diagram transformations. Picture logic is based on constraint logic programming handling constraints over real intervals. $\mathcal{ALCRP}(\mathcal{D})$ can also be instantiated with a similar concrete domain and it might be possible to also specify transformations over time if we utilize Allen's interval logic [Allen, 1983]. However, this is currently an open issue.

[Marriott and Meyer, 1997; Marriott and Meyer, 1998a] present a classification of visual languages by grammar hierarchies on the basis of *copy-restricted* constraint multiset grammars. We believe that constraint multiset grammars and the $\mathcal{ALCRP}(\mathcal{D})$ approach are getting quite close to each other since constraint specification and solving is now available in $\mathcal{ALCRP}(\mathcal{D})$ via concrete domains. However, this has to be more thoroughly analyzed.

For formalizing reasoning about spatial structures themselves many specific approaches have been published (see e.g. [Stock, 1997] for an overview). Ignoring decidability, Borgo et al. [Borgo et al., 1996] have developed a first order theory of space which formalizes different aspects such as mereology etc. An algebraic (but still undecidable) theory about space has been proposed by [Pratt and Lemon, 1997; Pratt and Schoop, 1997]. Research on the RCC theory is also summarized in [Cohn et al., 1997]. While first axiomatizations used first-order logic, recently, the spatial relations used in RCC have been defined in terms of intuitionistic logic and propositional modal logic [Bennett, 1995]. Although qualitative reasoning with RCC can be used in many applications, in GIS also *conceptual* knowledge combined with qualitative relations has to be considered. This problem is not addressed in the above-mentioned related work but a first solution was presented in this chapter.

We refer to Section 3.6 for a discussion of related work concerned with visual language theory and to [Marriott et al., 1998] for an extensive review of visual language theory.

8.6 Summary

The formalism presented in this chapter can be used to define the semantics of visual spatial queries and to reason about query validity and subsumption. We would like to emphasize that this approach has no restrictions about the ordering of input and the type of allowed relations, provided the corresponding concrete domain is admissible. It does not rely on special parsing techniques because this approach is purely declarative. It can even deal with ambiguous specifications since a DL reasoner can compute every model satisfying the specifications. This is addressed with the help of default reasoning. A problem with the approach presented in this chapter could be the worst-case time complexity of the underlying classification algorithms. However, almost every logical or constraint-oriented approach with an interesting expressiveness has to deal with tractability and decidability. It is also important to note that complexity issues of DLs are very well understood and analyzed.

Based on recent findings [Horrocks, 1997; Horrocks, 1998; Horrocks and Patel-Schneider, 1999; Haarslev and Möller, 1999b; Horrocks et al., 2000a] about optimizing DL reasoners

for the average case we investigated optimization techniques for $\mathcal{ALCRP}(\mathcal{D})$ reasoners (see [Turhan, 2000; Turhan and Haarslev, 2000]) as a first step towards an optimized implementation of $\mathcal{ALCRP}(\mathcal{D})$. The next chapters present our research on developing optimization techniques for the description logic $\mathcal{ALCNH}_{R^+}$.

Part V

Practical Reasoning with Description Logics

The fifth part of this monograph presents research about practical reasoning with expressive description logics. Chapter 9 introduces a new tableaux calculus deciding the ABox consistency problem for the expressive description logic $\mathcal{ALCNH}_{R^+}$. Prominent language features of $\mathcal{ALCNH}_{R^+}$ are number restrictions, role hierarchies, transitively closed roles, and generalized concept inclusions. This chapter is based on [Haarslev and Möller, 1999c; Haarslev and Möller, 2000b].

Chapter 10 investigates novel optimization techniques for practical reasoning with expressive ABox description logic (DL) systems. As an extension to state-of-the-art optimization techniques new algorithms and data structures for implementing a DL system supporting TBoxes and ABoxes are discussed. The new techniques can be divided into two major approaches: (i) design of optimizations for the tableaux calculus and (ii) exploitation of new transformation techniques for TBoxes and ABoxes in order to achieve improvements in average case performance. The advances are demonstrated by an empirical analysis of the DL system RACE. This chapter is based on [Haarslev et al., 1998c; Haarslev et al., 1998b; Haarslev and Möller, 1999a; Haarslev and Möller, 1999d; Haarslev and Möller, 1999b; Haarslev and Möller, 2000a; Haarslev and Möller, 2000d; Haarslev et al., 2000a].

Chapter 11 presents techniques addressing the problem of TBoxes which contain a large number of concept introduction axioms. This is demonstrated with an empirical analysis of the performance of RACE applied to knowledge bases containing more than 100000 axioms. 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. This chapter is based on [Haarslev and Möller, 2000c].

Chapter 9

Expressive ABox Reasoning with Number Restrictions, Role Hierarchies, and Transitively Closed Roles

Experiences with concept languages indicate that description logics (DLs) with at least negation and disjunction are required to solve practical modeling problems without resorting to ad hoc extensions. The requirements derived from practical applications of DLs ask for even more expressive languages. For instance, in [Sattler, 1996] the need for transitive roles is demonstrated for representing part-whole relations, family relations or partial orderings in general. It is argued that the trade-off between expressivity and complexity favors the integration of transitively closed roles instead of a transitive closure operator for roles. Other examples are given in [Horrocks, 1998], where the area of medical terminology is discussed. Design studies for the Galen project identified the need for modeling transitive part-whole, causal and compositional relations, and for organizing these relations into a hierarchy. Moreover, generalized concept inclusions were also required as a modeling tool, e.g. for expressing sufficient conditions for named concepts.

Motivated by the above-mentioned requirements we introduce in this chapter an ABox tableaux calculus for the description logic $\mathcal{ALCNH}_{R^+}$. It augments the basic logic $\mathcal{ALC}$ [Schmidt-Schauss and Smolka, 1991] with number restrictions, role hierarchies, and transitively closed roles. Note that these language features imply the presence of generalized concept inclusions and cyclic concepts. The use of number restrictions in combination with transitive roles and role hierarchies is syntactically restricted: no number restrictions are possible for (i) transitive roles and (ii) for any role which has a transitive subrole. Furthermore, we assume that the unique name assumption holds for ABox individuals.

$\mathcal{ALCNH}_{R^+}$ is an extension of $\mathcal{ALCNH}$ that itself can be polynomially reduced to $\mathcal{ALCNR}$ [Buchheit et al., 1993] and vice versa. It is possible to rephrase every hierarchy of role

names with a set of role conjunctions and vice versa [Buchheit et al., 1993]. Thus, our work on $\mathcal{ALCNH}_{R^+}$ extends the work on $\mathcal{ALCNR}$ 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]. Recently, the work on these logics has been extended for the language $\mathcal{ALCQHI}_{R^+}$ [Horrocks et al., 1999; Horrocks et al., 2000b]. Another approach is presented in [De Giacomo and 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 $\mathcal{CIQ}$. 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. the transitive closure) but does not support role hierarchies and allows number restrictions only for primitive roles.

9.1 Defining the Language

The next subsections introduce the syntax and semantics of the concept language and the assertional language of $\mathcal{ALCNH}_{R^+}$.

9.1.1 The Concept Language

Definition 9.1 (Role Hierarchy) Let us assume a set of role names R. The disjoint subsets P, T, and F of R denote non-transitive, transitive role names, and feature names, respectively ($R = P \cup T \cup F$). If $\mathsf{R}, \mathsf{S} \in R$ are role names, then $\mathsf{R} \sqsubseteq \mathsf{S}$ is called a *role inclusion* axiom. A *role hierarchy* $\mathcal{R}$ is a finite set of role inclusion axioms. We define $\sqsubseteq^*_{\mathcal{R}}$ as the reflexive transitive closure of $\sqsubseteq$ over the role hierarchy $\mathcal{R}$.

Additionally we define the set of ancestors and descendants of a role.

Definition 9.2 (Role Descendants/Ancestors) Given $\sqsubseteq^*_{\mathcal{R}}$, the following definitions are introduced.

- $\mathsf{R}^{\downarrow} = \{\mathsf{S} \in R \mid \mathsf{S} \sqsubseteq^*_{\mathcal{R}} \mathsf{R}\}$ (*descendants* of a role R)
- $\mathsf{R}^{\uparrow} = \{\mathsf{S} \in R \mid \mathsf{R} \sqsubseteq^*_{\mathcal{R}} \mathsf{S}\}$ (*ancestors* of a role R)
- $RS^{\uparrow} = \bigcup_{\mathsf{R} \in RS} \mathsf{R}^{\uparrow}$ (*ancestors* of a role set RS)
- $RS^{\downarrow} = \bigcup_{\mathsf{R} \in RS} \mathsf{R}^{\downarrow}$ (*descendants* of a role set RS)
- $S := \{\mathsf{R} \in P \mid \mathsf{R}^{\downarrow} \cap T = \emptyset\}$ (*simple* roles that are neither transitive nor have a transitive role as descendant)

Definition 9.3 (Concept Terms) Let C be a set of concept names that 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, $n > 1$, and $m > 0$ ($n, m \in \mathbb{N}$), then the following expressions are also concept terms:

- $\top$ (*top concept*)

- $\bot$ (*bottom concept*)
- $\mathsf{C} \sqcap \mathsf{D}$ (*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*)
- $\exists_{\leq m}\,\mathsf{S}$ (*at most number restriction*)
- $\exists_{\geq n}\,\mathsf{S}$ (*at least number restriction*).

Note that $\top$ ($\bot$) can also be expressed as $\mathsf{C} \sqcup \neg\mathsf{C}$ ($\mathsf{C} \sqcap \neg\mathsf{C}$). 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.

The concept language is syntactically restricting the combination of number restrictions and transitive roles. Number restrictions are only allowed for *simple* roles. This restriction is motivated by an undecidability result [Horrocks et al., 1999] in case of an unrestricted combinability.

Definition 9.4 (Generalized Concept Inclusions) If C and D are concept terms, then $\mathsf{C} \sqsubseteq \mathsf{D}$ (*generalized concept inclusion* or *GCI*) is a terminological axiom. A finite set of terminological axioms $\mathcal{T}_{\mathcal{R}}$ is called a *terminology* or *TBox* w.r.t. to a role hierarchy $\mathcal{R}$.[1]

GCIs can be used to represent terminological cycles. There exist at least two ways to deal with GCIs in a tableaux calculus. The "internalization" approach (e.g. see in [Horrocks and Sattler, 1999]) makes use of the fact that the expressiveness of GCIs is already implied by the combination of role hierarchies and transitive roles. For instance, this allows one to introduce an internal transitive role U as a superrole of all other roles. Then, a GCI $\mathsf{C} \sqsubseteq \mathsf{D}$ can be internalized as $\forall\,\mathsf{U}\,.\,(\neg\mathsf{C} \sqcup \mathsf{D})$ and there is no need to adapt a tableaux calculus w.r.t. GCIs. However, with the presence of arbitrary ABoxes one has also to consider unrelated individuals. For instance, in this case the internalization approach could introduce a new internal root individual that is related with every other individual in the ABox via the superrole U. Then, the all-concepts corresponding to the internalized GCIs are added to the root individual. Alternatively, one could directly add the corresponding assertions to all ABox individuals (e.g. $\mathsf{i}{:}\forall\,\mathsf{U}\,.\,(\neg\mathsf{C} \sqcup \mathsf{D})$) instead of creating a root individual. We decided to pursue a different and more direct approach. An ABox tableaux calculus is presented which offers new constructs and rules for dealing with GCIs (see Definition 9.7).

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

Definition 9.5 (Semantics) An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ (the domain) and an interpretation function $\cdot^{\mathcal{I}}$. The interpretation function maps each concept name C to a subset $\mathsf{C}^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name R to a subset $\mathsf{R}^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Let the symbols C, D be concept expressions, R be a role name, S be simple role name, $n > 1$, and $m > 0$

[1] The reference to $\mathcal{R}$ is omitted in the following if we use $\mathcal{T}$.

$(n, m \in \mathbb{N})$. Then the interpretation function can be extended to arbitrary concept and role terms as follows ($\|\cdot\|$ denotes the cardinality of a set):

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

An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ (w.r.t. to a role hierarchy $\mathcal{R}$) iff it satisfies

- $\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$ for all terminological axioms $\mathsf{C} \sqsubseteq \mathsf{D}$ in $\mathcal{T}$,
- $\mathsf{R}^{\mathcal{I}} \subseteq \mathsf{S}^{\mathcal{I}}$ for all role inclusion axioms $\mathsf{R} \sqsubseteq \mathsf{S}$ in $\mathcal{R}$,
- $\mathsf{R}^{\mathcal{I}} = (\mathsf{R}^{\mathcal{I}})^{+}$ for every $\mathsf{R} \in T$, and
- $\Delta^{\mathcal{I}} \subseteq (\exists_{\leq 1}\, \mathsf{F})^{\mathcal{I}}$ for every $\mathsf{F} \in F$

A concept term C *subsumes* a concept term D w.r.t. a TBox $\mathcal{T}$ (written $\mathsf{D} \preceq_{\mathcal{T}} \mathsf{C}$), iff $\mathsf{D}^{\mathcal{I}} \subseteq \mathsf{C}^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. A concept term C is *satisfiable* w.r.t. a TBox $\mathcal{T}$ iff there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $\mathsf{C}^{\mathcal{I}} \neq \emptyset$.

One of the basic reasoning services for a description logic formalism is computing the subsumption relationship between named concepts (i.e. elements from C). This inference is needed in the TBox to build a hierarchy of concept names w.r.t. specificity. Satisfiability and subsumption can be mutually reduced to each other since $\mathsf{C} \preceq_{\mathcal{T}} \mathsf{D}$ iff $\mathsf{C} \sqcap \neg \mathsf{D}$ is not satisfiable w.r.t. $\mathcal{T}$ and C is unsatisfiable w.r.t. $\mathcal{T}$ iff $\mathsf{C} \preceq_{\mathcal{T}} \bot$.

9.1.2 The Assertional Language

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 follows.

Definition 9.6 (ABox Assertions) Let $O = O_O \cup O_N$ be a set of individual names, where the set O_O is disjoint to the set O_N (see also below). If C is a concept term, R a role name, and $\mathsf{a}, \mathsf{b} \in O$ are individual names, then the following expressions are *assertional axioms*:

- $\mathsf{a}:\mathsf{C}$ (*concept assertion*),
- $(\mathsf{a},\mathsf{b}):\mathsf{R}$ (*role 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 of $\Delta^{\mathcal{I}}$ in a way such 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 $\mathsf{a} \neq \mathsf{b}$, or if we assume without loss of generality that $\mathsf{a} \in O_N, \mathsf{b} \in O_O$.[2] An interpretation $\mathcal{I}$ w.r.t. to a TBox $\mathcal{T}$ 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 $\mathcal{A}$ (or *satisfies* $\mathcal{A}$) w.r.t. a TBox $\mathcal{T}$ and a role hierarchy $\mathcal{R}$ iff it is a model of $\mathcal{T},\mathcal{R}$ and furthermore satisfies all assertional axioms in $\mathcal{A}$.[3] An ABox is *consistent* w.r.t. a TBox $\mathcal{T}$ iff it has a model w.r.t. $\mathcal{T}$. 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. An individual a is called an *instance* of a concept term C in an interpretation $\mathcal{I}$ iff $\mathsf{a}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$. The *direct types* of an individual are the most specific atomic concepts which the individual is an instance of.

The ABox consistency problem is to decide whether a given ABox $\mathcal{A}$ is consistent w.r.t. a TBox $\mathcal{T}$. Satisfiability of concept terms can be reduced to ABox consistency as follows: A concept term C is satisfiable iff the ABox $\{\mathsf{a}:\mathsf{C}\}$ is consistent. *Instance checking* tests whether an individual a is an instance of a concept term C w.r.t. an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$, i.e. whether $\mathcal{A}$ entails $\mathsf{a}:\mathsf{C}$ w.r.t. $\mathcal{T}$. This problem is reduced to the problem of deciding if the ABox $\mathcal{A} \cup \{\mathsf{a}:\neg\mathsf{C}\}$ is inconsistent.

9.2 ABox Reasoning versus Concept Consistency

ABox reasoning truly extends the usefulness of description logics in practical applications. The increase of expressiveness is also reflected in an increase of the complexity of the tableaux rules (see Section 9.4.1 for more details). An alternative might be the so-called "precompletion approach" originally developed for the language $\mathcal{ALCQ}$ [Hollunder, 1994] and extended to the logic $\mathcal{ALCH}_{R^+}$ [Tessaris and Gough, 1999]. The idea behind the precompletion approach is to transform given ABoxes such that ABox consistency is reduced to concept satisfiability. This is achieved by generating a precompletion of an ABox where

[2] The set of "old" individuals names characterizes all individuals for which the unique name assumption holds while the set of "new" names denotes individuals which are constructed during a proof.

[3] For sake of brevity we may omit the reference to $\mathcal{T}$ if we use $\mathcal{I}$.

all role filler relationships between ABox individuals (e.g. $\mathsf{i}_1, \ldots, \mathsf{i}_n$) have been absorbed into corresponding concept terms (e.g. $\mathsf{C}_1, \ldots, \mathsf{C}_n$). Then, ABox consistency can be reduced to testing the satisfiability of a concept conjunction (e.g. $\mathsf{C}_1 \sqcap \ldots \sqcap \mathsf{C}_n$). The advantage of this approach is that it allows one to reuse existing tableaux provers for concept consistency. However, there currently exist no calculi for computing the precompletion of ABoxes for languages such as $\mathcal{ALCNH}_{R^+}$ or even $\mathcal{ALCQHI}_{R^+}$. Moreover, for practical applications one can argue that a translational approach via precompletion techniques might raise problems for relating results from the concept consistency tester (e.g. concept incoherence) to corresponding ABox individuals and their assertions (e.g. ABox incoherence).

Another difficulty results from applying the optimization technique called dependency-directed backtracking (see Section 10.2.2 for a discussion of this technique). The translational approach will need similar information from a concept consistency tester in order to avoid unnecessary backtracking. An ABox tableaux calculus can easily avoid this problem as illustrated with the following ABox $\mathcal{A}$.

$$\mathcal{A} = \{\mathsf{i} : \forall\,\mathsf{R}\,.\,\mathsf{A} \sqcup \forall\,\mathsf{R}\,.\,\mathsf{B},\ (\mathsf{i},\mathsf{k}) : \mathsf{R},\ (\mathsf{j},\mathsf{k}) : \mathsf{R},\ \mathsf{j} : \mathsf{C} \sqcup \mathsf{D},\ \mathsf{k} : \neg\mathsf{A} \sqcap \neg\mathsf{B}\}$$

To deal with disjunctions the $\mathcal{ALCNH}_{R^+}$ ABox tableaux calculus (see below for details) non-deterministically generates new ABoxes. An effective search procedure has to use backtracking to exhaustively explore all possible alternatives. For instance, the case $\mathsf{i}:\forall\,\mathsf{R}\,.\,\mathsf{A}$ must be explored. Another choice point is the disjunction $\mathsf{j}:\mathsf{C} \sqcup \mathsf{D}$. Without loss of generality, the system tries $\mathsf{j}:\mathsf{C}$ before considering constraints for k (note that C might contain concept value restrictions). Afterwards, the concept constraint $\mathsf{i}:\forall\,\mathsf{R}\,.\,\mathsf{A}$ is treated in combination with the role constraint $(\mathsf{i},\mathsf{k}):\mathsf{R}$. After some additional expansion steps, this part of the search tree will lead to a clash because $\mathsf{k}:\mathsf{A}$ and $\mathsf{k}:\neg\mathsf{A}$ will be elements of the ABox. Now, if the system backtracks to the choice point $\mathsf{j}:\mathsf{C} \sqcup \mathsf{D}$ and tries $\mathsf{j}:\mathsf{D}$ it is bound to detect the same clash for k again. Since D can be a very complex concept term, many expansion steps are definitely wasted. Using dependency-directed backtracking, one detects that $\mathsf{j}:\mathsf{C}$ is not involved in the clash and backtracking is set up again for trying $\mathsf{i}:\forall\,\mathsf{R}\,.\,\mathsf{B}$.

9.3 An ABox Example

Before we continue with the calculus for $\mathcal{ALCNH}_{R^+}$, we illustrate in the following the expressiveness of $\mathcal{ALCNH}_{R^+}$ with a TBox and ABox example about family relationships. This example uses prominent features of $\mathcal{ALCNH}_{R^+}$ such as transitive roles, role hierarchies, number restrictions and generalized concept inclusions.

In the TBox *family* we assume a role has_descendant which is declared to be transitive, has_gender which is declared as a feature, and a role has_sibling. The TBox *family* contains the following role axioms.

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

$\textbf{has\_sister} \sqsubseteq \mathsf{has\_sibling}$

$\textbf{has\_brother} \sqsubseteq \mathsf{has\_sibling}$

The TBox *family* contains concept axioms specifying the domain and/or range of the roles introduced above (the domain A of a role R can be expressed by the axiom $\exists_{\geq 1}\,\mathsf{R} \sqsubseteq \mathsf{A}$ and the range B by $\top \sqsubseteq \forall\,\mathsf{R}\,.\,\mathsf{B}$).

$\exists_{\geq 1}\,\mathsf{has\_descendant} \sqsubseteq \mathsf{human}$

$\top \sqsubseteq \forall\,\mathsf{has\_descendant}\,.\,\mathsf{human}$

$\exists_{\geq 1}\,\mathsf{has\_child} \sqsubseteq \mathsf{parent}$

$\exists_{\geq 1}\,\mathsf{has\_sibling} \sqsubseteq \mathsf{sibling}$

$\top \sqsubseteq \forall\,\mathsf{has\_sibling}\,.\,\mathsf{sibling}$

$\top \sqsubseteq \forall\,\mathsf{has\_sister}\,.\,\mathsf{sister}$

$\top \sqsubseteq \forall\,\mathsf{has\_brother}\,.\,\mathsf{brother}$

$\top \sqsubseteq \forall\,\mathsf{has\_gender}\,.\,(\mathsf{female} \sqcup \mathsf{male})$

The next axioms guarantee the disjointness between the concepts female, male, and human.

$\textbf{female} \sqsubseteq \neg(\mathsf{human} \sqcup \mathsf{male})$

$\textbf{male} \sqsubseteq \neg(\mathsf{human} \sqcup \mathsf{female})$

$\textbf{human} \sqsubseteq \neg(\mathsf{female} \sqcup \mathsf{male})$

After these preliminaries we start with axioms expressing basic knowledge about family members. We use $\mathsf{C} \doteq \mathsf{D}$ as an abbreviation for $\mathsf{C} \sqsubseteq \mathsf{D}$ and $\mathsf{D} \sqsubseteq \mathsf{C}$.

$\textbf{human} \sqsubseteq \exists_{\geq 1}\,\mathsf{has\_gender}$

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

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

$\textbf{parent} \doteq \exists_{\geq 1}\,\mathsf{has\_child}$

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

$\textbf{father} \doteq \mathsf{man} \sqcap \mathsf{parent}$

The next axioms describe some aspects of relatives of a family. Note the inferred equivalences between the concept pairs "mother_with_. . . /mother_of_. . ." and "mother_having_. . ." as shown in Figure 9.1.

mother_having_only_female_kids $\doteq$ mother $\sqcap \forall$ has_child . $\forall$ has_gender . female
mother_having_only_daughters $\doteq$ mother $\sqcap \exists_{\geq 1}$ has_child $\sqcap \forall$ has_child . woman
mother_with_kids $\doteq$ mother $\sqcap \exists_{\geq 2}$ has_child
grandpa $\doteq$ man $\sqcap \exists$ has_child . parent
great_grandpa $\doteq$ man $\sqcap \exists$ has_child . ($\exists$ has_child . parent)
grandma $\doteq$ woman $\sqcap \exists$ has_child . parent
great_grandma $\doteq$ woman $\sqcap \exists$ has_child . ($\exists$ has_child . parent)
aunt $\doteq$ woman $\sqcap \exists$ has_sibling . parent
uncle $\doteq$ man $\sqcap \exists$ has_sibling . parent
sibling $\doteq$ sister $\sqcup$ brother
sister $\doteq$ woman $\sqcap \exists_{\geq 1}$ has_sibling
brother $\doteq$ man $\sqcap \exists_{\geq 1}$ has_sibling
mother_of_siblings $\doteq$ mother $\sqcap \forall$ has_child . sibling

There still exists no formal relationship between the notions "mother having kids" and "mother of siblings." This is expressed by the next two axioms. The last axiom defines a concept mother_of_sisters which has the other specific "mother_..." concepts as parents (see Figure 9.1).

$\exists_{\geq 2}$ has_child $\sqsubseteq \ \forall$ has_child . sibling
$\exists$ has_child . sibling $\sqsubseteq \ \exists_{\geq 2}$ has_child
mother_of_sisters $\doteq$ mother $\sqcap \forall$ has_child . (sister $\sqcap \forall$ has_sibling . sister)

Using the TBox *family*, the ABox *smith_family* is specified. It consists of several assertions about the individuals alice, betty, charles, doris, and eve. The individual alice is the mother of her two children betty and charles.

alice : woman $\sqcap \exists_{\leq 2}$ has_child
(alice, betty) : has_child
(alice, charles) : has_child

The individual betty is the sibling of charles and the mother of doris and eve, who are the only siblings of each other. The individual charles is the only brother of betty.

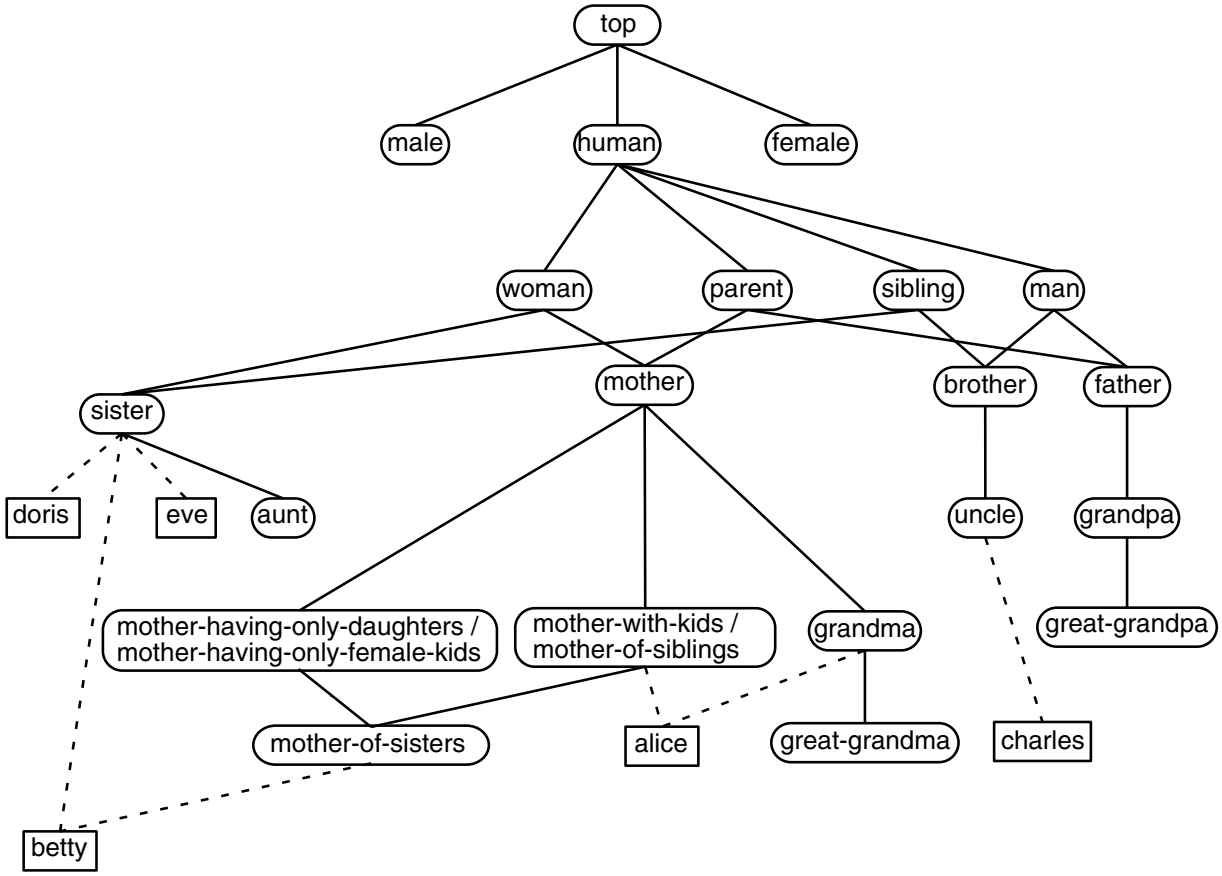


Figure 9.1: Concept hierarchy of the TBox *family* augmented with the individuals from the ABox *smith_family*. Ovals represent atomic concepts, rectangles denote ABox individuals, solid lines show the direct subsumption relationship, and dashed lines the instance membership of the individuals for their direct types.

$$
\begin{aligned}
\mathsf{betty} &: \mathsf{woman} \sqcap \exists_{\leq 2}\, \mathsf{has\_child} \sqcap \exists_{\leq 1}\, \mathsf{has\_sibling} \\
(\mathsf{betty}, \mathsf{doris}) &: \mathsf{has\_child} \\
(\mathsf{betty}, \mathsf{eve}) &: \mathsf{has\_child} \\
(\mathsf{betty}, \mathsf{charles}) &: \mathsf{has\_sibling} \\
\mathsf{charles} &: \mathsf{brother} \sqcap \exists_{\leq 1}\, \mathsf{has\_sibling} \\
(\mathsf{charles}, \mathsf{betty}) &: \mathsf{has\_sibling} \\
\mathsf{doris} &: \exists_{\leq 1}\, \mathsf{has\_sibling} \\
\mathsf{eve} &: \exists_{\leq 1}\, \mathsf{has\_sibling} \\
(\mathsf{doris}, \mathsf{eve}) &: \mathsf{has\_sister} \\
(\mathsf{eve}, \mathsf{doris}) &: \mathsf{has\_sister}
\end{aligned}
$$

Figure 9.1 also shows the inferred *direct types* of the individuals in the ABox *smith_family*. The individual alice has as direct types {mother_of_siblings, grandma}, the individual betty has {mother_of_sisters, sister}, charles has {uncle}, and doris and eve have {sister}. These inferences clearly demonstrate the expressiveness of $\mathcal{ALCNH}_{R^+}$.

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

In the following we devise a *tableaux* algorithm to decide the consistency of $\mathcal{ALCNH}_{R^+}$ ABoxes. The algorithm is characterized by a set of tableaux or *completion* rules and by a particular *completion strategy* ensuring a specific order for applying the completion rules to assertional axioms of an ABox. The strategy is essential to guarantee the completeness of the ABox consistency algorithm. The purpose of the calculus is to generate a so-called completion for an initial ABox $\mathcal{A}$ in order to prove the consistency of $\mathcal{A}$ or its inconsistency if no completion can be found.

The treatment of features is omitted in the following because any ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ can be transformed such that features are declared as roles and for every feature name F an axiom $\top \sqsubseteq \exists_{\leq 1} \mathsf{F}$ is added to $\mathcal{T}$. Obviously, the new ABox w.r.t. the new TBox is consistent iff $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$.

First, we have to introduce new assertional axioms needed to define the augmentation of an ABox.

Definition 9.7 (Additional ABox Assertions) Let C be a concept term, the individual names $\mathsf{a}, \mathsf{b} \in O$, and $x \notin O$, then the following expressions are also assertional axioms:

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

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 $\mathsf{a}^{\mathcal{I}} \neq \mathsf{b}^{\mathcal{I}}$.

Given the new ABox assertions we define for any concept term its negation normal form.

Definition 9.8 (Negation Normal Form) We assume the same naming conventions as in Definition 9.3. The negation normal form is defined by applying the following transformations in such a way that a negation sign may occur only in front of concept names. This transformation is possible in linear time.

- $\neg\top \rightarrow \bot$
- $\neg\bot \rightarrow \top$
- $\neg\neg\mathsf{C} \rightarrow \mathsf{C}$
- $\neg(\mathsf{C} \sqcap \mathsf{D}) \rightarrow \neg\mathsf{C} \sqcup \neg\mathsf{D}$
- $\neg(\mathsf{C} \sqcup \mathsf{D}) \rightarrow \neg\mathsf{C} \sqcap \neg\mathsf{D}$

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

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

Definition 9.9 (Augmented ABox) For an initial ABox $\mathcal{A}$ w.r.t a TBox $\mathcal{T}$ and a role hierarchy $\mathcal{R}$ we define its *augmented* ABox or its *augmentation* $\mathcal{A}'$ by applying the following rules to $\mathcal{A}$. 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 $O_O = \{\mathsf{a}_1, \ldots, \mathsf{a}_n\}$ be the set of individuals mentioned in $\mathcal{A}$, then the following set of inequality assertions is added to $\mathcal{A}'$: $\{\mathsf{a}_i \neq \mathsf{a}_j \mid \mathsf{a}_i, \mathsf{a}_j \in O_O, \mathsf{i}, \mathsf{j} \in 1..n, \mathsf{i} \neq \mathsf{j}\}$. From this point on, if we refer to an initial ABox $\mathcal{A}$ we always mean its augmented ABox. Obviously, if $\mathcal{A}'$ is an augmentation of $\mathcal{A}$ then $\mathcal{A}'$ is consistent iff $\mathcal{A}$ is consistent.

The tableaux calculus also requires the notion of *blocking* the applicability of tableaux rules. This is based on so-called concept sets, an ordering for individuals, and on the notion of blocking individuals.

Definition 9.10 (Concept Sets) Given an ABox $\mathcal{A}$ and an individual a occurring in $\mathcal{A}$, we define the *concept set* of a as $\sigma(\mathcal{A}, \mathsf{a}) := \{\top\} \cup \{\mathsf{C} \mid \mathsf{a}\!:\!\mathsf{C} \in \mathcal{A}\}$.

Definition 9.11 (Individual 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 9.12 (Blocking Individual, blocked) Let $\mathcal{A}$ be an ABox and $\mathsf{a}, \mathsf{b} \in O$ be individuals in $\mathcal{A}$. We call a the blocking individual of b if the following conditions hold:

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

If there exists a blocking individual a for b, then b is said the be *blocked* (by a).

9.4.1 Completion Rules

We are now ready to define the *completion rules* that are intended to generate a so-called completion (see also below) of an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$.

Definition 9.13 (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. a is not blocked, 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 $\mathcal{A}$

R∃$_{\geq n}$ The number restriction exists rule (generating).
if 1. $\mathsf{a}:\exists_{\geq n}\,\mathsf{R} \in \mathcal{A}$, and
2. a is not blocked, and
3. $\neg\exists\,\mathsf{b}_1,\ldots,\mathsf{b}_n \in O,\ \mathsf{S}_1,\ldots,\mathsf{S}_n \in \mathsf{R}^{\downarrow}$:
$\{(\mathsf{a},\mathsf{b}_k):\mathsf{S}_k \mid \mathsf{k} \in 1..\mathsf{n}\} \cup \{\mathsf{b}_i \neq \mathsf{b}_j \mid \mathsf{i},\mathsf{j} \in 1..\mathsf{n}, \mathsf{i} \neq \mathsf{j}\} \subseteq \mathcal{A}$
then $\mathcal{A}' = \mathcal{A} \cup \{(\mathsf{a},\mathsf{b}_k):\mathsf{R} \mid \mathsf{k} \in 1..\mathsf{n}\} \cup \{\mathsf{b}_i \neq \mathsf{b}_j \mid \mathsf{i},\mathsf{j} \in 1..\mathsf{n}, \mathsf{i} \neq \mathsf{j}\}$
where $\mathsf{b}_1,\ldots,\mathsf{b}_n \in O_N$ are not used in $\mathcal{A}$

R∃$_{\leq n}$ The number restriction merge rule (nondeterministic).
if 1. $\mathsf{a}:\exists_{\leq n}\,\mathsf{R} \in \mathcal{A}$, and
2. $\exists\,\mathsf{b}_1,\ldots,\mathsf{b}_m \in O,\ \mathsf{S}_1,\ldots,\mathsf{S}_m \in \mathsf{R}^{\downarrow}$: $\{(\mathsf{a},\mathsf{b}_1):\mathsf{S}_1,\ldots,(\mathsf{a},\mathsf{b}_m):\mathsf{S}_m\} \subseteq \mathcal{A}$
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 b_i in $\mathcal{A}$ by b_j

We call the rules R$\sqcup$ and R$\exists_{\leq n}$ *nondeterministic* rules since they can be applied to the same set of assertions in different ways. 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 the only rules that introduce new individuals in an ABox.

The increase of expressiveness in $\mathcal{ALCNH}_{R^+}$ gained by supporting ABox reasoning is reflected in tableaux rules that are more complex than in comparable approaches for concept consistency. The universal concept restriction rule takes care of GCIs and usually causes additional complexity by adding disjunctions to an ABox. The generating rules have a more complex premise since they may test only for a blocking situation if they are applied to new individuals, i.e. a blocking situation can never occur for old individuals. The necessity of this additional precondition is illustrated by the following example. We define a concept D where R is a transitive superrole of S.

$$\mathbf{D} \doteq \mathsf{C} \sqcap \exists \mathsf{S}\,.\,\mathsf{C} \sqcap \exists_{\leq 1} \mathsf{S} \sqcap \forall \mathsf{R}\,.\,\exists \mathsf{S}\,.\,\mathsf{C}$$
$$\mathcal{A} := \{(\mathsf{i},\mathsf{j}) : \mathsf{S},\ (\mathsf{j},\mathsf{k}) : \mathsf{S},\ \mathsf{i} : \mathsf{D},\ \mathsf{j} : \mathsf{D},\ \mathsf{k} : \neg\mathsf{C}\}$$

Then, we define an ABox $\mathcal{A}$ which is obviously unsatisfiable due to a clash for the individual k with $\mathsf{C} \sqcap \neg\mathsf{C}$. However, if blocking were allowed for old individuals, the R$\exists$C-rule would not create a S-successor with qualification C for the individual j. As a consequence, the number restriction merge rule would never merge this successor with the individual k which causes the inconsistency of $\mathcal{A}$.

The next proposition proves that the completion rules given above preserve the satisfiability of ABoxes.

Proposition 9.1 (Invariance) Let $\mathcal{A}$ be an augmented ABox and $\mathcal{A}'$ be an ABox derived from $\mathcal{A}$. Then:

1. If $\mathcal{A}'$ is derived from $\mathcal{A}$ by applying a deterministic rule, then $\mathcal{A}$ is satisfiable iff $\mathcal{A}'$ is satisfiable.

2. If $\mathcal{A}'$ is derived from $\mathcal{A}$ by applying a nondeterministic rule, then $\mathcal{A}$ is satisfiable if $\mathcal{A}'$ is satisfiable. Conversely, if $\mathcal{A}$ is satisfiable and a nondeterministic rule is applicable to $\mathcal{A}$, then it can be applied in such a way that it yields a satisfiable ABox $\mathcal{A}'$.

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 satisfiable if $\mathcal{A}'$ is satisfiable.

"$\Rightarrow$" In order to show that $\mathcal{A}'$ is satisfiable after applying a deterministic rule to the satisfiable ABox $\mathcal{A}$, we examine each applicable rule separately. We assume that $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ satisfies $\mathcal{A}$. It is an obvious consequence that $\mathsf{R}^{\mathcal{I}} \subseteq \mathsf{S}^{\mathcal{I}}$ iff $\mathsf{R} \sqsubseteq^*_{\mathcal{R}} \mathsf{S}$.

If the conjunction rule is applied to $\mathsf{a}\!:\!\mathsf{C} \sqcap \mathsf{D} \in \mathcal{A}$, then we get $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{a}\!:\!\mathsf{C},\ \mathsf{a}\!:\!\mathsf{D}\}$. Since $\mathcal{I}$ satisfies $\mathsf{a}\!:\!\mathsf{C} \sqcap \mathsf{D}$, $\mathcal{I}$ 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 $\mathcal{I}$ satisfies $\mathcal{A}$, it holds that $(\mathsf{a}^{\mathcal{I}},\mathsf{b}^{\mathcal{I}})\in\mathsf{S}^{\mathcal{I}}\subseteq\mathsf{R}^{\mathcal{I}}$. Since $\mathcal{I}$ satisfies $\mathsf{a}:\forall\,\mathsf{R}\,.\,\mathsf{C}$, it holds that $\mathsf{b}^{\mathcal{I}}\in\mathsf{C}^{\mathcal{I}}$. Thus, $\mathcal{I}$ 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 $\mathsf{S}\in\mathsf{T}^{\downarrow}\subseteq\mathsf{R}^{\downarrow}$, $\mathsf{T}\in T$ such that we get $\mathcal{A}'=\mathcal{A}\cup\{\mathsf{b}:\forall\,\mathsf{T}\,.\,\mathsf{C}\}$. Since $\mathcal{I}$ satisfies $\mathcal{A}$, we have $\mathsf{a}^{\mathcal{I}}\in(\forall\,\mathsf{R}\,.\,\mathsf{C})^{\mathcal{I}}$ and $(\mathsf{a}^{\mathcal{I}},\mathsf{b}^{\mathcal{I}})\in\mathsf{S}^{\mathcal{I}}\subseteq\mathsf{T}^{\mathcal{I}}\subseteq\mathsf{R}^{\mathcal{I}}$. Since $\mathcal{I}$ 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 $(\mathsf{a}^{\mathcal{I}},\mathsf{c}^{\mathcal{I}})\in\mathsf{T}^{\mathcal{I}}\subseteq\mathsf{R}^{\mathcal{I}}$ and $\mathsf{a}^{\mathcal{I}}\notin(\forall\,\mathsf{R}\,.\,\mathsf{C})^{\mathcal{I}}$ in contradiction to the assumption. Thus, $\mathcal{I}$ 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 $\mathcal{I}$ satisfies $\mathcal{A}$, it holds that $\mathsf{C}^{\mathcal{I}}=\Delta^{\mathcal{I}}$. Thus, it holds that $\mathsf{a}^{\mathcal{I}}\in\mathsf{C}^{\mathcal{I}}$ and $\mathcal{I}$ 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 new ABox $\mathcal{A}'=\mathcal{A}\cup\{(\mathsf{a},\mathsf{b}):\mathsf{R},\ \mathsf{b}:\mathsf{C}\}$. Since $\mathcal{I}$ satisfies $\mathcal{A}$, 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 $\mathcal{I}'=(\Delta^{\mathcal{I}},\cdot^{\mathcal{I}'})$ 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 new ABox $\mathcal{A}'=\mathcal{A}\cup\{(\mathsf{a},\mathsf{b}_\mathsf{k}):\mathsf{R}\,|\,\mathsf{k}\in 1..\mathsf{n}\}\cup\{\mathsf{b}_\mathsf{i}\neq\mathsf{b}_\mathsf{j}\,|\,\mathsf{i},\mathsf{j}\in 1..\mathsf{n},\mathsf{i}\neq\mathsf{j}\}$. Since $\mathcal{I}$ satisfies $\mathcal{A}$, 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 $\mathcal{I}'=(\Delta^{\mathcal{I}},\cdot^{\mathcal{I}'})$ satisfies $\mathcal{A}'$.

2. "$\Leftarrow$" Assume that $\mathcal{A}'$ is satisfied by $\mathcal{I}'=(\Delta^{\mathcal{I}},\cdot^{\mathcal{I}'})$. We show that $\mathcal{A}$ is also satisfiable by examining the nondeterministic rules.

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 $\mathcal{I}'$.

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 $x^{\mathcal{I}}:=x^{\mathcal{I}'}$ for every $x\neq\mathsf{b}_\mathsf{i}$. Obviously $\mathcal{I}=(\Delta^{\mathcal{I}},\cdot^{\mathcal{I}})$ satisfies $\mathcal{A}$.

"$\Rightarrow$" We suppose that $\mathcal{I}=(\Delta^{\mathcal{I}},\cdot^{\mathcal{I}})$ satisfies $\mathcal{A}$ 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 satisfiable, 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 $\mathcal{I}$ 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 satisfiable, it holds $\mathsf{a}^{\mathcal{I}}\in(\exists_{\leq n}\,\mathsf{R})^{\mathcal{I}}$ and $\|\{x\,|\,(x,y)\in\mathsf{R}^{\mathcal{I}}\}\|\leq n$. However, it is also $\|\{\mathsf{b}\,|\,(\mathsf{a}^{\mathcal{I}},\mathsf{b}^{\mathcal{I}})\in\mathsf{R}^{\mathcal{I}}\}\|>m$ with $m\geq n$.[4] Thus, we can conclude by the Pigeonhole Principle (e.g. see [Lewis and Papadimitriou, 1981, page 26]) that there exist at least two R-successors $\mathsf{b}_\mathsf{i},\mathsf{b}_\mathsf{j}$ of a such that

[4]Without loss of generality we only need to consider the case that $m=n+1$.

$\mathsf{b_i}^{\mathcal{I}} = \mathsf{b_j}^{\mathcal{I}}$. Since $\mathcal{I}$ satisfies $\mathcal{A}$, we have $\mathsf{b_i} \not\approx \mathsf{b_j} \notin \mathcal{A}$ and at least one of the two individuals must be a new individual. Let us assume that $\mathsf{b_i} \in O_N$ and $\mathsf{b_i} = \mathsf{b_j}$, then $\mathcal{I}$ obviously satisfies $\mathcal{A}[\mathsf{b_i}/\mathsf{b_j}]$. □

Given an initial ABox $\mathcal{A}$, more than one rule might be applicable to $\mathcal{A}$. This is controlled by a completion strategy in accordance with the ordering for new individuals (see Definition 9.11).

Definition 9.14 (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⊓, R⊔, R∀C, R$\forall_+$C, R$\forall_x$, R$\exists_{\leq n}$) as long as possible.
  2. Apply a generating rule (R∃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 with this strategy. It ensures that the rules are applied to new individuals w.r.t. the ordering '$\prec$'.

Definition 9.15 (Clash Triggers) We assume the same naming conventions as used above. An ABox $\mathcal{A}$ is called *contradictory* 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*:
  $\mathsf{a}:\bot \in \mathcal{A}$ or $\{\mathsf{a}:\mathsf{C}, \mathsf{a}:\neg\mathsf{C}\} \subseteq \mathcal{A}$, where C is a concept name.
- *Number restriction merging clash*:
  $\exists\, \mathsf{S_1}, \ldots, \mathsf{S_m} \in \mathsf{R}^{\downarrow} : \{\mathsf{a}:\exists_{\leq n}\, \mathsf{R}\} \cup \{(\mathsf{a}, \mathsf{b_i}):\mathsf{S_i} \mid \mathsf{i} \in 1..\mathsf{m}\} \cup \{\mathsf{b_i} \not\approx \mathsf{b_j} \mid \mathsf{i}, \mathsf{j} \in 1..\mathsf{m}, \mathsf{i} \neq \mathsf{j}\} \subseteq \mathcal{A}$ 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. In the following we have to show that a model can be constructed for any complete ABox.

9.4.2 Decidability of the ABox Consistency Problem

The following lemma proves that whenever a generating rule has been applied to an individual a, the concept set $\sigma(\cdot, \mathsf{a})$ of a does not change for succeeding ABoxes.

Lemma 9.1 (Stability) Let $\mathcal{A}$ be an ABox and $\mathsf{a} \in O_N$ be mentioned 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. **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. □

The next lemma guarantees the uniqueness of a blocking individual for a blocked individual. This is a precondition for defining a particular interpretation from $\mathcal{A}$.

Lemma 9.2 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 and
2. a has exactly one blocking individual.

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 9.1 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 9.12. □

Definition 9.16 Let $\mathcal{A}$ be a complete ABox generated by the calculus. We define the *canonical interpretation* $\mathcal{I}_\mathcal{A} = (\Delta^{\mathcal{I}_\mathcal{A}}, \cdot^{\mathcal{I}_\mathcal{A}})$ w.r.t. $\mathcal{A}$ as follows:

1. $\Delta^{\mathcal{I}_\mathcal{A}} := \{\mathsf{a} \mid \mathsf{a} \text{ is an individual in } \mathcal{A}\}$
2. $\mathsf{a}^{\mathcal{I}_\mathcal{A}} := \mathsf{a}$ iff a is mentioned in $\mathcal{A}$
3. $\mathsf{a} \in \mathsf{A}^{\mathcal{I}_\mathcal{A}}$ iff $\mathsf{a}:\mathsf{A} \in \mathcal{A}$
4. $(\mathsf{a}, \mathsf{b}) \in \mathsf{R}^{\mathcal{I}_\mathcal{A}}$ iff $\exists\, \mathsf{c}_0, \ldots, \mathsf{c}_\mathsf{n}, \mathsf{d}_0, \ldots, \mathsf{d}_{\mathsf{n}-1}$ mentioned in $\mathcal{A}$ such that[5]
   (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
   (c) $\forall \mathsf{i} \in 0..\mathsf{n}-1$:
      i. $\mathsf{d}_\mathsf{i} = \mathsf{c}_\mathsf{i}$, or
      ii. d_i is a blocking individual for c_i and $(\mathsf{d}_\mathsf{i}, \mathsf{c}_{\mathsf{i}+1}):\mathsf{S}_{\mathsf{i}+1} \in \mathcal{A}$, and
   (d) if $\mathsf{n} > 1$
      $\forall\, \mathsf{i} \in 1..\mathsf{n} : \exists \mathsf{R}' \in T,\ \mathsf{R}' \in \mathsf{R}^\downarrow,\ \mathsf{S}_\mathsf{i} \in \mathsf{R}'^\downarrow$
   else
      $\mathsf{S}_1 \in \mathsf{R}^\downarrow$.

The construction of the canonical interpretation for the case 4 is illustrated with two examples in Figure 9.2. The following two cases can be seen as special cases of case 4 introduced above ($\mathsf{n} = 1, \mathsf{c}_0 = \mathsf{a}, \mathsf{c}_1 = \mathsf{b}$):

- $\mathsf{c}_0 = \mathsf{d}_0$: $(\mathsf{a}, \mathsf{b}) \in \mathsf{R}^\mathcal{I}$ iff $(\mathsf{c}_0, \mathsf{c}_1):\mathsf{S}_1 \in \mathcal{A}$ for a role $\mathsf{S}_1 \in \mathsf{R}^\downarrow$,
- $\mathsf{c}_0 \neq \mathsf{d}_0$: $(\mathsf{a}, \mathsf{b}) \in \mathsf{R}^\mathcal{I}$ iff d_0 is a blocking individual for 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 9.2 the canonical interpretation is well-defined because there exists a unique blocking individual for each blocked individual.

Theorem 9.1 (Soundness) Let $\mathcal{A}$ be a complete ABox generated by the calculus, then $\mathcal{A}$ is satisfiable.

[5]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.

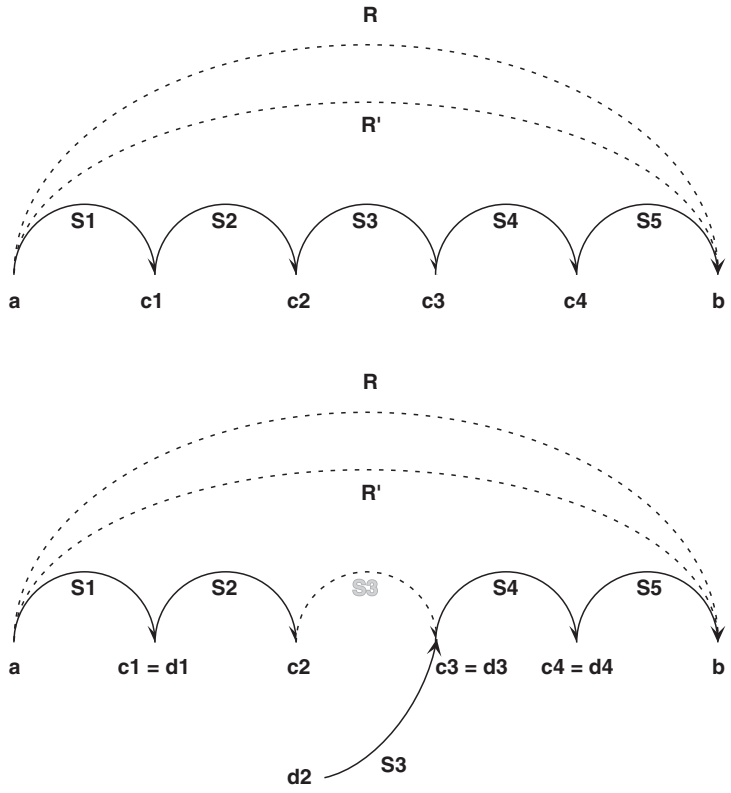


Figure 9.2: Construction of the canonical interpretation (two examples for case 4). In the lower example we assume that the individual $\mathsf{d2}$ is a blocking individual for $\mathsf{c2}$ (see text).

Proof. Let $\mathcal{I}_\mathcal{A} = (\Delta^{\mathcal{I}_\mathcal{A}}, \cdot^{\mathcal{I}_\mathcal{A}})$ be the canonical interpretation for the ABox $\mathcal{A}$.[6]

All role inclusion axioms in the role hierarchy $\mathcal{R}$ of $\mathcal{T}$ are satisfied: For every $\mathsf{S} \sqsubseteq \mathsf{R}$ in $\mathcal{R}$ it holds that $\mathsf{S}^{\mathcal{I}_\mathcal{A}} \subseteq \mathsf{R}^{\mathcal{I}_\mathcal{A}}$. This can be shown as follows. If $(\mathsf{a}, \mathsf{b}) \in \mathsf{S}^{\mathcal{I}_\mathcal{A}}$, case 4 of Definition 9.16 must be applicable. Hence, there exists a chain of subroles possibly with gaps and blocking individuals. Thus, the corresponding case for the construction of $\mathcal{I}_\mathcal{A}$, which adds (a, b) to $\mathsf{S}^{\mathcal{I}_\mathcal{A}}$, is also applicable to R since $\mathsf{S} \in \mathsf{R}^{\downarrow}$ (see 4d). Therefore, there is also the tuple $(\mathsf{a}, \mathsf{b}) \in \mathsf{R}^{\mathcal{I}_\mathcal{A}}$.

All transitivity axioms are satisfied, i.e. transitive roles are interpreted in the correct way: $\forall\, \mathsf{R} \in T : \mathsf{R}^{\mathcal{I}_\mathcal{A}} = (\mathsf{R}^{\mathcal{I}_\mathcal{A}})^+$. If there exist $(\mathsf{a}, \mathsf{b}) \in \mathsf{R}^{\mathcal{I}_\mathcal{A}}$ and $(\mathsf{b}, \mathsf{c}) \in \mathsf{R}^{\mathcal{I}_\mathcal{A}}$ then case 4 in Definition 9.16 must have been applied for each tuple. But then, there also exists a chain of roles from a to c (possibly with gaps and blocking individuals) such that (a, c) is added to $\mathsf{R}^{\mathcal{I}_\mathcal{A}}$ as well.

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

For any $(\mathsf{a}, \mathsf{b}) : \mathsf{R} \in \mathcal{A}$ or $\mathsf{a} \neq \mathsf{b} \in \mathcal{A}$, $\mathcal{I}_\mathcal{A}$ satisfies them by definition. Next we consider assertions of the form $\mathsf{a} : \mathsf{C}$. We show by induction on the structure of C that $\mathsf{a} \in \mathsf{C}^{\mathcal{I}_\mathcal{A}}$.

[6] For sake of brevity we denote the interpretation of individual names by their names (e.g. a instead of $\mathsf{a}^{\mathcal{I}}$ due to the fact that $\mathcal{I}_\mathcal{A}$ is the canonical interpretation and $\mathsf{a} = \mathsf{a}^{\mathcal{I}}$.

If C is a concept name, then $\mathsf{a} \in \mathsf{C}^{\mathcal{I}_\mathcal{A}}$ by definition of $\mathcal{I}_\mathcal{A}$. If $\mathsf{C} = \top$, then obviously $\mathsf{a} \in \top^{\mathcal{I}_\mathcal{A}}$. The case $\mathsf{C} = \bot$ cannot occur since $\mathcal{A}$ is clash-free.

If $\mathsf{C} = \neg\mathsf{D}$, then D is a concept name since all concepts are in negation normal form (see Definition 9.9). $\mathcal{A}$ is clash-free and cannot contain $\mathsf{a}:\mathsf{D}$. Thus, $\mathsf{a} \notin \mathsf{D}^{\mathcal{I}_\mathcal{A}}$, i.e. $\mathsf{a} \in \Delta^{\mathcal{I}_\mathcal{A}} \setminus \mathsf{D}^{\mathcal{I}_\mathcal{A}}$. Hence $\mathsf{a} \in (\neg\mathsf{D})^{\mathcal{I}_\mathcal{A}}$.

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, $\mathsf{a} \in \mathsf{C}_1{}^{\mathcal{I}_\mathcal{A}}$ and $\mathsf{a} \in \mathsf{C}_2{}^{\mathcal{I}_\mathcal{A}}$. Hence $\mathsf{a} \in (\mathsf{C}_1 \sqcap \mathsf{C}_2)^{\mathcal{I}_\mathcal{A}}$.

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, $\mathsf{a} \in \mathsf{C}_1{}^{\mathcal{I}_\mathcal{A}}$ or $\mathsf{a} \in \mathsf{C}_2{}^{\mathcal{I}_\mathcal{A}}$. Hence $\mathsf{a} \in (\mathsf{C}_1 \sqcup \mathsf{C}_2)^{\mathcal{I}_\mathcal{A}}$.

If $\mathsf{C} = \forall\,\mathsf{R}\,.\,\mathsf{D}$, then we have to show that for all b with $(\mathsf{a},\mathsf{b}) \in \mathsf{R}^{\mathcal{I}_\mathcal{A}}$ it holds that $\mathsf{b} \in \mathsf{D}^{\mathcal{I}_\mathcal{A}}$. If $(\mathsf{a},\mathsf{b}) \in \mathsf{R}^{\mathcal{I}_\mathcal{A}}$, then according to Definition 9.16, b is a successor of a via a chain of roles $\mathsf{S}_\mathsf{i} \in \mathsf{R}^\downarrow$ or there exist corresponding blocking individuals as domain elements of $\mathsf{S}_\mathsf{i} \in \mathsf{R}^\downarrow$, i.e. the chain might contain "gaps" with associated blocking individuals (see Figure 9.2). Since $(\mathsf{a},\mathsf{b}) \in \mathsf{R}^{\mathcal{I}_\mathcal{A}}$ and $\mathsf{S}_\mathsf{i}{}^{\mathcal{I}_\mathcal{A}} \subseteq \mathsf{R}^{\mathcal{I}_\mathcal{A}}$ there exists tuples $(\mathsf{c}_\mathsf{i},\mathsf{c}_{\mathsf{i}+1}) \in \mathsf{S}_\mathsf{i}{}^{\mathcal{I}_\mathcal{A}}$. Due to Definition 9.16 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 $\mathsf{b} \in \mathsf{D}^{\mathcal{I}_\mathcal{A}}$. As mentioned before, the chain of roles can have one or more "gaps" (see Figure 9.2). However, due to Definition 9.16 in case of a "gap" there exists a blocking individual such that a similar argument as in case 4 can be applied, i.e. in case of a gap between c_i and $\mathsf{c}_{\mathsf{i}+1}$ with d_i as blocking individual for c_i, the blocking condition ensures that the concept set of the blocking individual 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 $\mathsf{b} \in \mathsf{D}^{\mathcal{I}_\mathcal{A}}$ by induction hypothesis.

If $\mathsf{C} = \exists\,\mathsf{R}\,.\,\mathsf{D}$, then we have to show that there exists an individual $\mathsf{b} \in \Delta^{\mathcal{I}_\mathcal{A}}$ with $(\mathsf{a},\mathsf{b}) \in \mathsf{S}^{\mathcal{I}_\mathcal{A}}$, $\mathsf{S} \in \mathsf{R}^\downarrow$, and $\mathsf{b} \in \mathsf{D}^{\mathcal{I}_\mathcal{A}}$. Since ABox $\mathcal{A}$ is complete, we have either $(\mathsf{a},\mathsf{b}):\mathsf{S} \in \mathcal{A}$, $\mathsf{S} \in \mathsf{R}^\downarrow$, and $\mathsf{b}:\mathsf{D} \in \mathcal{A}$, or a is blocked by an individual c and $(\mathsf{c},\mathsf{b}):\mathsf{S} \in \mathcal{A}$ with $\mathsf{S} \in \mathsf{R}^\downarrow$. In the first case we have $(\mathsf{a},\mathsf{b}) \in \mathsf{R}^{\mathcal{I}_\mathcal{A}}$ by the definition of $\mathcal{I}_\mathcal{A}$ (case 4, $\mathsf{n} = 1$, $\mathsf{c}_\mathsf{i} = \mathsf{d}_\mathsf{i}$) and $\mathsf{b} \in \mathsf{D}^{\mathcal{I}_\mathcal{A}}$ by induction hypothesis. In the second case there exists the blocking individual c with $\mathsf{c}:\exists\,\mathsf{R}\,.\,\mathsf{D} \in \mathcal{A}$. 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 $\mathsf{b} \in \mathsf{D}^{\mathcal{I}_\mathcal{A}}$ and by the definition of $\mathcal{I}_\mathcal{A}$ (case 4, $\mathsf{n} = 1$, $\mathsf{c}_\mathsf{i} \neq \mathsf{d}_\mathsf{i}$) we have $(\mathsf{a},\mathsf{b}) \in \mathsf{R}^{\mathcal{I}_\mathcal{A}}$.

If $\mathsf{C} = \exists_{\geq n}\,\mathsf{R}$, we prove the hypothesis by contradiction. We assume that $\mathsf{a} \notin (\exists_{\geq n}\,\mathsf{R})^{\mathcal{I}_\mathcal{A}}$. 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 $\mathcal{I}_\mathcal{A}$. 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 $\mathsf{a} \notin (\exists_{\leq n}\,\mathsf{R})^{\mathcal{I}_\mathcal{A}}$. Then there exist at least $n+1$ distinct individuals $\mathsf{b}_1,\ldots,\mathsf{b}_{n+1}$ such that $(\mathsf{a},\mathsf{b}_i) \in \mathsf{R}^{\mathcal{I}_\mathcal{A}}$, $i \in 1..n+1$. According to Definition 9.16 the following two cases can occur. (1) The individual a is not

blocked. Then, 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. $\mathsf{b}_i \neq \mathsf{b}_j \in \mathcal{A}$, $i, j \in 1..n+1$, $i \neq j$. This contradicts the assumption that $\mathcal{A}$ is clash-free. (2) There exists a blocking individual c 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 blocking individual) and a non-transitive role R (R is required to be a simple role, see the syntactic restrictions for number restrictions and role hierarchies), there is no $(\mathsf{a}, \mathsf{b}_k) \in \mathsf{R}^{\mathcal{I}_\mathcal{A}}$ if there is no $(\mathsf{c}, \mathsf{b}_k) \in \mathsf{R}^{\mathcal{I}_\mathcal{A}}$ ($\mathsf{k} \in 1..\mathsf{n}+1$).

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, $\mathsf{a} \in \mathsf{D}^{\mathcal{I}_\mathcal{A}}$. Thus, $\mathcal{I}_\mathcal{A}$ satisfies $\forall\, x\,.\,(x : \mathsf{D})$. Finally, since $\mathcal{I}_\mathcal{A}$ satisfies all assertions in $\mathcal{A}$, $\mathcal{I}_\mathcal{A}$ satisfies $\mathcal{A}$. □

Theorem 9.2 (Completeness) Let $\mathcal{A}$ be an augmented ABox which is satisfiable, then there exists at least one completion of $\mathcal{A}$ computed by applying the completion rules to $\mathcal{A}$.

Proof. By contraposition: Obviously, an ABox containing a clash is unsatisfiable. If there does not exists a completion of $\mathcal{A}$, then it follows from Proposition 9.1 that ABox $\mathcal{A}$ is unsatisfiable. □

Definition 9.17 For any augmentation of an initial ABox $\mathcal{A}$, we define the *concept size* $n_\mathcal{A}$ as the number of concepts or subconcepts occurring in $\mathcal{A}$.[7] Note that $n_\mathcal{A}$ is bound by the length of the string expressing $\mathcal{A}$. The *size* of an ABox $\mathcal{A}$ is defined as $n_\mathcal{A} \times \|T\| + \|O_O\|$.

Lemma 9.3 Let $\mathcal{A}$ be an ABox and let $\mathcal{A}'$ be a completion of $\mathcal{A}$. In any set X consisting of individuals occurring in $\mathcal{A}'$ with a cardinality greater than $2^{n_\mathcal{A}}$ there exist at least two individuals $\mathsf{a}, \mathsf{b} \in X$ whose concept sets are equal.

Proof. Each assertion $\mathsf{a} : \mathsf{C_i} \in \mathcal{A}'$ may contain at most $n_\mathcal{A}$ different concepts $\mathsf{C_i}$. So there cannot exist more than $2^{n_\mathcal{A}}$ different concept sets for the individuals in $\mathcal{A}'$. □

Lemma 9.4 Let $\mathcal{A}$ be an ABox and let $\mathcal{A}'$ be a completion of $\mathcal{A}$. Then there occur at most $2^{n_\mathcal{A}}$ non-blocked new individuals in $\mathcal{A}'$.

Proof. Suppose we have $2^{n_\mathcal{A}} + 1$ non-blocked new individuals in $\mathcal{A}'$. From Lemma 9.3 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 9.11 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 fact $\sigma(\mathcal{A}', \mathsf{a}) = \sigma(\mathcal{A}', \mathsf{b})$ implies $\sigma(\mathcal{A}', \mathsf{a}) \supseteq \sigma(\mathcal{A}', \mathsf{b})$. Then we have either b is blocked by a or there exists an individual c with b blocked by c and $\mathsf{c} \prec \mathsf{a}$. Both cases contradict the hypothesis. □

Theorem 9.3 (Termination) Let $\mathcal{A}_\mathcal{T}$ be the augmented ABox w.r.t a TBox $\mathcal{T}$ and let n be the size of $\mathcal{A}_\mathcal{T}$. Every completion of $\mathcal{A}_\mathcal{T}$ is finite and its size is $O(2^{4n})$.

[7] We have to increase $n_\mathcal{A}$ by 1 if $\top$ does not occur in $\mathcal{A}$.

Proof. Let $\mathcal{A}'$ be a completion of $\mathcal{A}_\mathcal{T}$. From Lemma 9.4 we know that $\mathcal{A}'$ has at most 2^n non-blocked new individuals. Therefore, a total of at most $m \times 2^n$ new individuals may exists in $\mathcal{A}'$, where m is the maximum number of direct successors for any individual in $\mathcal{A}'$.

Note that m is bound 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 bound by 2^n. Hence, we have $m \leq 2^n + n$. Since the number of individuals in the initial ABox is also bound 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 bound by n and each assertion has a size linear in n. Hence, the total size of these assertions is bound $n \times n \times 2^{2n}$, 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 bound by $(2^{2n})^2$, i.e. $O(2^{4n})$. In conclusion, we have a size of $O(2^{4n})$ for $\mathcal{A}'$. □

Theorem 9.4 (Decidability) Let $\mathcal{A}_\mathcal{T}$ be an ABox w.r.t. a TBox $\mathcal{T}$. Checking whether $\mathcal{A}_\mathcal{T}$ is satisfiable is a decidable problem.

Proof. This follows immediately from the Theorems 9.1, 9.2, and 9.3. □

9.5 Summary

In this chapter we presented a tableaux calculus deciding the ABox consistency problem for the description logic $\mathcal{ALCNH}_{R^+}$. A highly optimized variant of this calculus is already implemented in the ABox description logic system RACE demonstrating the practical usefulness of $\mathcal{ALCNH}_{R^+}$. Although TBox reasoners for logics such as $\mathcal{ALCQHI}_{R^+}$ are available, the development of $\mathcal{ALCNH}_{R^+}$ and its optimized implementation in RACE is a novel approach. The next two chapters present the design and evaluation of major optimization techniques some of which are especially suited for $\mathcal{ALCNH}_{R^+}$.

Chapter 10

Optimization Techniques for Reasoning with Expressive ABox and Concept Expressions

This chapter investigates novel optimization techniques for practical reasoning with expressive ABox description logic (DL) systems. As an extension to state-of-the-art optimization techniques the chapter discusses new algorithms and data structures for implementing a DL system supporting TBoxes and ABoxes. The new techniques can be divided into two major approaches: (i) design of optimizations for the tableaux calculus and (ii) exploitation of new transformation techniques for TBoxes and ABoxes in order to achieve improvements in average case performance. The advances are demonstrated by an empirical analysis of the DL system RACE.

10.1 Introduction

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; Haarslev and Möller, 2000a]. RACE implements an $\mathcal{ALCNH}_{R^+}$ reasoner for answering queries concerning ABoxes and TBoxes. RACE is a successor of HAM-ALC [Haarslev and Möller, 1999a] and employs a large variety of well-known as well as novel optimization techniques. The techniques described in this chapter are fully implemented in RACE which is freely available for research purposes [Haarslev and Möller, 1999e].

The main contributions of this chapter can be summarized as follows:

- It is demonstrated that optimization techniques developed for testing concept consistency and concept subsumption [Horrocks and Patel-Schneider, 1999] scale up for testing ABox consistency and can be successfully integrated into an ABox reasoning architecture.

- We introduce and analyze several optimization techniques which are either novel or which significantly extend the techniques presented in [Horrocks and Patel-Schneider, 1999]. The new techniques being investigated are called *deep model merging*, *signature calculus*, *GCI transformation*, *role path contraction*, and *individual model merging* (see the detailed discussion below).

The new techniques are empirically evaluated using TBoxes derived from actual applications. In addition, the algorithms are analyzed with a set of synthetic TBox and ABox benchmark problems. The empirical results indicate a performance gain of up to several orders of magnitude.

10.2 Optimizing TBox and ABox Reasoning

The tableaux calculus introduced in the previous chapter is of theoretical interest for proving the decidability of the ABox consistency problem. For practical purposes such calculi are highly inefficient. Therefore, the development of optimization techniques is a very important research topic. This section briefly summarizes some of the already established optimization techniques 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. In addition to well-known optimized TBox reasoning algorithms described in [Baader et al., 1992; Baader et al., 1994b], the RACE architecture incorporates the following established optimization techniques (see also [Horrocks, 1997; Horrocks and Patel-Schneider, 1999; Horrocks et al., 2000a] for a detailed explanation): lexical normalization, lazy unfolding, dependency-directed backtracking, semantic branching, BCP constraint propagation, heuristics guided search, subtableaux caching, model caching and model merging tests. These techniques form the basis of other 'modern' DL system implementations (e.g. FaCT and DLP [Horrocks and Patel-Schneider, 1999]). The techniques were developed for reasoners deciding only the concept consistency problem. With our RACE architecture we demonstrate that these techniques scale up and can be integrated into an ABox reasoning architecture [Haarslev and Möller, 1999b]. In order to explain the effect of our new optimizations and for sake of completeness, some of the known techniques need to be reviewed in the following paragraphs.

10.2.1 Semantic Branching

In contrast to syntactic branching, 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]). Semantic branching is usually supported by various techniques intending to speed-up the search.

A *lookahead algorithm* or *constraint propagator* tries to reduce the order of magnitude of the open search space. Thus, after every expansion step the truth value of the newly

added assertions is propagated into all open disjuncts of all unexpanded or-assertions. As a result of this step, or-assertions might be become satisfied (i.e. one disjunct is satisfied), deterministic (i.e. exactly one disjunct remains open), or might even clash (all disjuncts are unsatisfied).

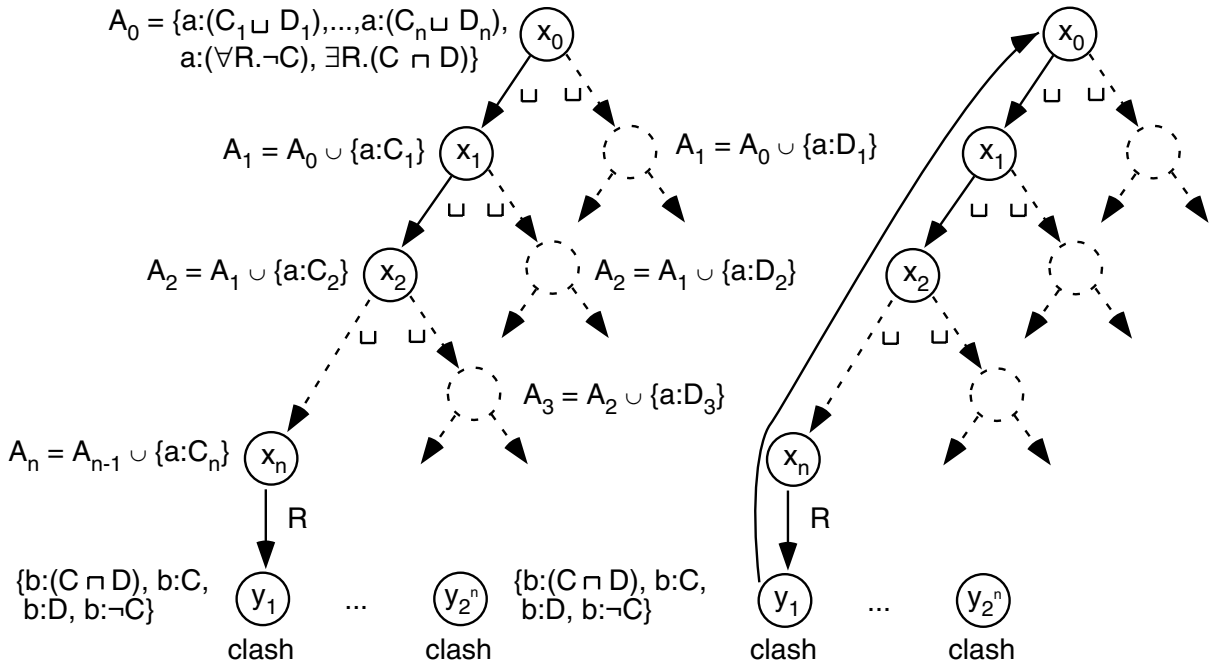
Various *heuristics* are used to select the next unexpanded or-assertion and disjunct. A dynamic selection scheme is often employed. The oldest-first strategy is used for selecting one or-assertion with at least two open disjuncts. In order to select a disjunct the heuristic counts the number of negated and unnegated occurrences for each open disjunct from the selected assertion in all other unexpanded or-assertions. These numbers are used as input for a priority function that selects the disjunct. The priority function achieves the following goals. It prefers disjuncts that occur frequently in unexpanded binary or-assertions and balanced or-assertions (i.e. containing a similar number of negated and unnegated occurrences of the same disjunct) but discriminates between unbalanced or-assertions. In order to perform the counting very quickly, RACE precomputes for every or-assertion two lists cross-referencing or-assertions that contain the assertions' concept in negated or unnegated form. Once a disjunct is selected, the priority function is also used to determine whether the disjunct is first tried in negated or unnegated form. In case of a failure, the other alternative is explored.

10.2.2 Dependency-directed Backtracking

Naive backtracking algorithms often explore regions of the search space rediscovering the same contradictions repeatedly. Dependency-directed backtracking records the dependencies of expanded assertions and in case of a clash backtracks to or-assertions that are responsible for at least one of the clash-causing assertions in the subtree (see [Freeman, 1995]). When an assertion is expanded, its dependencies are recorded, i.e. its precondition assertions are saved as a dependency set. In the RACE architecture, every or-assertion which is element of a dependency set of a precondition assertion is also an element of the dependency set of the resulting assertion (dependency propagation).

When a clash occurs, a so-called clash set which is the union of the dependency sets of the clash culprits (i.e. possibly containing the or-assertions that generated these assertions) is stored and backtracking is started. This idea is due to FaCT [Horrocks, 1997]. Whenever a semantic branching point is encountered during backtracking, it is checked whether this or-assertion is responsible for a clash culprit, i.e. whether it is a member of the clash set. If the or-assertion is not found in the clash set, one can safely bypass this branching point. In case the or-assertion is found, either the remaining semantic alternative is tried or this disjunct is considered as unsatisfiable in the current subtree. The backtracking continues but removes the current or-assertion from the clash set and adds the saved clash dependencies of the previously unsatisfiable alternative.

The dependency-directed backtracking technique is illustrated in Figure 10.1. Let us assume the satisfiability of the concept $(\mathsf{C}_1 \sqcup \mathsf{D}_1) \sqcap \ldots \sqcap (\mathsf{C}_n \sqcup \mathsf{D}_n) \sqcap \exists \mathsf{R}\,.\,(\mathsf{C} \sqcap \mathsf{D}) \sqcap \forall \mathsf{R}\,.\,\neg\mathsf{C}$



(a) Search tree for ABox A_0 (b) Backjumping to root

Figure 10.1: Illustration of the dependency-directed backtracking technique. The satisfiability of the concept term $(\mathsf{C}_1 \sqcup \mathsf{D}_1) \sqcap \ldots \sqcap (\mathsf{C}_n \sqcup \mathsf{D}_n) \sqcap \exists \mathsf{R}.(\mathsf{C} \sqcap \mathsf{D}) \sqcap \forall \mathsf{R}.\neg\mathsf{C}$ has to be checked. Using dependency-directed backtracking only n steps are required in contrast to the case without dependency-directed backtracking where 2^n steps are needed.

has to be checked.[1] The tableaux calculus (see previous chapter) will expand the initial ABox as shown in Figure 10.1a. After applying the R⊓-rule it creates the ABox A_0. Then, the R⊔-rule is applied until a disjunct has been selected for all disjunctions.[2] A new subtableaux for $(\mathsf{C} \sqcap \mathsf{D}) \sqcap \neg\mathsf{C}$ is created and the primitive clash for C is discovered. An uniformed search (i.e. without dependency-directed backtracking) would now backtrack to the last branching point and would always rediscover the same clash for the R-successor. Thus, the satisfiability test for the original concept term would require 2^n expansion steps. Of course, after the discovery of the first clash, the other 2^{n-1} expansion steps are definitely wasted because the clash culprits for the R-successor (i.e. $\mathsf{a}:\exists \mathsf{R}.(\mathsf{C} \sqcap \mathsf{D})$ and $\mathsf{a}:\forall \mathsf{R}.\neg\mathsf{C}$) *depend* on the initial ABox. Therefore, as illustrated in Figure 10.1b the remaining alternatives may be safely skipped due to the dependencies of the clashing assertions if the dependency-directed backtracking technique is used.

[1] In general, we can assume that the clash for the R-successor might be hard to detect.

[2] For sake of simplicity we assume a selection scheme in accordance with the syntactic ordering and no semantic branching, i.e. the first disjunct is selected for the left-most path.

10.2.3 Subtableaux Caching

In tableaux calculi for testing concept consistency (see e.g. [Horrocks, 1998] for a calculus for deciding concept consistency for $\mathcal{ALCH}f_{R^+}$), sets of concepts representing a conjunction are manipulated.[3] 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[4] and checks if a so-called "subtableaux" for $\{\mathsf{C}, \mathsf{D}, \mathsf{E}\}$ can be found that contains no clash. The concepts in the original concept set L are not relevant for this test. It can be shown that this so-called 'trace technique' [Schmidt-Schauss and Smolka, 1991] can be also applied to $\mathcal{ALCH}f_{R^+}$. 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. For brevity, in this context we also call a concept set for which consistency is to be tested a "subtableau."

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.[5] The cache value indicates whether the conjunction of the key is satisfiable. 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.

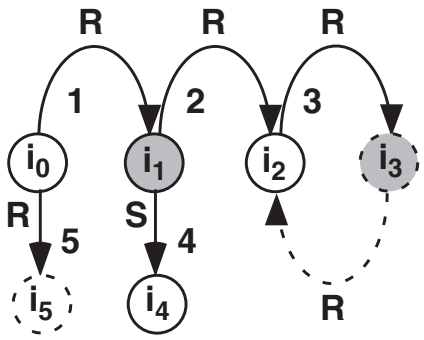
For *concept consistency*, the cache technique described above is sketched in [Horrocks and Patel-Schneider, 1999]. However, in an implementation of a tableaux calculus for deciding *ABox consistency*, the technique cannot be directly applied since it might be possible that the new individual which is generated by applying the rule for treating some-assertions must be identified with an old individual, i.e. additional assertions are 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 containing just the concept assertions 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.

It is always safe to cache the inconsistency of a subtableau. In case the consistency of a subtableau is also cached one has to take care that the caching technique remains sound. This is illustrated with the following example. Solving the consistency problem for the ABox

[3] These concept sets are also called "labels."

[4] For brevity, in the explanation of subtableaux caching we neglect the correct treatment of features and role hierarchies.

[5] 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 10.2: Caching example with blocking (see text).

$\{i_0 : E\}$ w.r.t. the following (cyclic) inclusion axioms demonstrates that caching depends on the context when a cache entry is generated.

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

$\mathbf{D} \sqsubseteq \exists R . C$

$\mathbf{E} \sqsubseteq (\exists R . C) \sqcup (\exists R . D)$

The proof steps are presented in Figure 10.2 where the sequence of "expansion" steps is indicated with numbers. In step 1, the initial problem $\{i_0 : E\}$ is presented. Since there is an axiom for E involving a disjunction we get two (alternative) ABoxes (see step 2). Let us assume the first alternative is tried first. This leads to a subtableau (step 3). The assertion from step 3 is expanded w.r.t. the axioms and we get the ABox in step 4. The assertion $i_1 : \exists R . D$ is expanded first. In step 5 the corresponding subtableau is considered. The right-hand side of the axiom for D is inserted (step 6). The assertion $i_2 : \exists R . C$ yields another subtableau (step 7). Due to the blocking strategy (see Definition 9.12), the constraint system in step 7 is not expanded (see the ABox in step 3 with the blocking individual i_1). In Figure 10.2 the canonical interpretation is indicated with a dashed arrow.

An often-employed strategy is to cache intermediate results, e.g. the satisfiability of D is stored as a cache entry at the end of step 7. In our example, there are some proof steps pending. In step 8 the remaining assertions from step 4 are considered. Obviously, the subtableau for the role S causes a clash. Therefore, the second alternative in step 2 has to be considered. The corresponding subtableau is presented in 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 has to be provided in order to guarantee the soundness of caching. This mechanism is implemented in RACE. Once the unsatisfiability of a concept (or the inconsistency of a constraint system) on which a cached pseudo model is dependent is detected, the corresponding cache entries have to be (recursively) removed.

Although memory consumption might lead to problems and cache 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 cache management techniques are needed in order to preserve an upper bound on memory consumption.

10.2.4 Lazy Unfolding

In the previous section it was explained that the tableaux calculus treats axioms by transforming them into universal concept restrictions. Obviously, this adds additional assertions to *every* subtableaux. Even worse, in most cases the additional assertions contain disjunctions, i.e. the search space is extended. Although techniques for efficiently managing the search space are integrated into modern DL architectures (e.g. semantic branching, BCP constraint propagation and heuristic search), in the average case it is always advantageous to reduce the number of assertions the prover has to deal with. Therefore, a technique called "lazy unfolding" is employed.[6] The idea is to exploit special forms of GCIs in the TBox. If there is a GCI $\mathsf{A} \sqsubseteq \mathsf{C}$ ($\neg\mathsf{A} \sqsubseteq \mathsf{C}$) in the TBox where A is a concept name and (i) there is no other GCI with A or $\neg\mathsf{A}$ on the left-hand side, (ii) there is no GCI $\mathsf{C} \sqsubseteq \mathsf{A}$ ($\mathsf{C} \sqsubseteq \neg\mathsf{A}$) in the TBox and (iii) the GCI $\mathsf{A} \sqsubseteq \mathsf{C}$ ($\neg\mathsf{A} \sqsubseteq \mathsf{C}$) is not cyclic, then the GCI $\mathsf{A} \sqsubseteq \mathsf{C}$ is called a primitive concept definition (i.e. a definition with only necessary conditions) or, in case of $\neg\mathsf{A} \sqsubseteq \mathsf{C}$, a negated primitive concept definition. If there exists an additional GCI $\mathsf{C} \sqsubseteq \mathsf{A}$ ($\mathsf{C} \sqsubseteq \neg\mathsf{A}$), the set $\{\mathsf{A} \sqsubseteq \mathsf{C}, \mathsf{C} \sqsubseteq \mathsf{A}\}$ ($\{\neg\mathsf{A} \sqsubseteq \mathsf{C}, \mathsf{C} \sqsubseteq \neg\mathsf{A}\}$) is called a concept definition (i.e. a definition with necessary and sufficient conditions). As usual we also write $\mathsf{A} \doteq \mathsf{C}$ ($\mathsf{A} \doteq \neg\mathsf{C}$). In the following, GCIs that represent (negated) primitive concept definitions are called *simple* GCIs.

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 (negated) primitive concept definition $\mathsf{A} \sqsubseteq \mathsf{C}$ ($\neg\mathsf{A} \sqsubseteq \mathsf{C}$), the definition of A ($\neg\mathsf{A}$) is inserted into the ABox as an assertion $\mathsf{i}\!:\!\mathsf{C}$. Alternatively, if there exists a concept definition for A, $\mathsf{i}\!:\!\mathsf{A}$ is "expanded" 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, if there exists a primitive concept definition for A but no negated primitive concept definition, then $\mathsf{i}\!:\!\neg\mathsf{A}$ need not be "expanded" using the above-mentioned technique.

Empirical tests indicate a significant speedup if the lazy unfolding technique is implemented in a prover architecture.

[6] The $\mathcal{KRIS}$ system also uses lazy unfolding in order to avoid unfolding [Baader et al., 1992; Baader et al., 1994b]. However, with GCIs, unfolding has to take care of terminological cycles.

10.2.5 GCI transformation

The general idea of this technique is a transformation or compilation process that tries to eliminate GCIs in a preprocessing phase [Horrocks, 1997; Horrocks and Tobies, 2000]. The GCI transformation is illustrated by the following example where the two GCIs $\{\mathsf{A} \sqsubseteq \mathsf{E} \sqcup \mathsf{F},\ \mathsf{A} \sqcap \mathsf{B} \sqsubseteq \mathsf{C} \sqcap \mathsf{D}\}$ can be transformed into the single concept inclusion axiom $\mathsf{A} \sqsubseteq (\mathsf{E} \sqcup \mathsf{F}) \sqcap (\neg\mathsf{B} \sqcup (\mathsf{C} \sqcap \mathsf{D}))$. Due to the transformation there remains only one concept inclusion with A on the left-hand side. Thus, we have a simple GCI.

Any non-simple GCI that cannot be absorbed has to be represented by universal concept restrictions (see the corresponding rule in Definition 9.13). 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.

10.2.6 Benchmark Problems used for Evaluation

The next sections discuss five techniques which are either novel or can be called extensions of optimization techniques discussed before. The effectiveness of these techniques is evaluated with TBoxes and ABoxes derived from actual applications and a set of synthetic TBox and ABox benchmark problems [Haarslev et al., 1999a].

- The 'Galen' TBoxes (named Galen2, Galen1, Galen) were already used in [Horrocks, 1997; Horrocks, 1998; Horrocks and Patel-Schneider, 1999]. The original TBox is named Galen and uses the logic $\mathcal{ALCH}f_{R^+}$. The other two TBoxes were derived from the original one, Galen2 uses $\mathcal{ALE}$ and Galen1 uses $\mathcal{ALC}$.

- The 'bike' TBoxes (using $\mathcal{ALCNH}$ with GCIs) contain configuration knowledge about various types of bicycles. Several ABoxes for the 'bike' TBoxes were also developed and used for benchmarking.

- The TBoxes 'BCS3' and 'BCS4' (using $\mathcal{ALC}$ with GCIs) are derived from a telecommunication application [Areces et al., 1999]. They are automatically generated with an exponentially increasing size and difficulty and make heavy use of terminological cycles and GCIs.

- The set of synthetic ABox benchmark problems was already used in [Haarslev and Möller, 1999b]. The new set of synthetic TBox problems evaluates the efficiency of the generating completion rules and the number merging rule. The runtimes for both problem sets (with problems between level $1-21$) are supposed to grow exponentially.

10.3 New Transformations on GCIs

Based on [Horrocks, 1997], we developed an extended scheme for transforming GCIs. These transformations are motivated by a heuristics trying 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). The transformation scheme starts with the sets D, G defined as follows. Let us assume that all concepts are in negation normal form and the terms 'concept definition' and 'primitive concept definition' are defined as above.

$D := \{\mathsf{A} \sqsubseteq \mathsf{C} \,|\, \mathsf{A} \sqsubseteq \mathsf{C} \text{ in } \mathcal{T},\, \mathsf{A} \in C\} \cup \{\mathsf{A} \doteq \mathsf{C} \,|\, \mathsf{A} \doteq \mathsf{C} \text{ in } \mathcal{T},\, \mathsf{A} \in C\}$
$G := \{\mathsf{C} \sqsubseteq \mathsf{D} \in \mathcal{T}\} \setminus D$

The procedure enhanced_gci_absorption specifies the absorption process implemented in RACE. It is called with a tuple consisting of the sets D, G and returns a new tuple with possibly modified sets. The transformation process employs six steps as described in the following. Note that the steps 4-5 repeat the steps 2-3 with a different value for the second parameter value.

Procedure 1 enhanced_gci_absorption($\langle D, G\rangle$)

1: $\langle D', G'\rangle \leftarrow$ absorb_defined_concepts($\langle D', G'\rangle$)
2: $\langle D', G'\rangle \leftarrow$ absorb_gcis($\langle D', G'\rangle$,*false*)
3: $\langle D', G'\rangle \leftarrow$ transform_absorb_gcis($\langle D', G'\rangle$,*false*)
4: $\langle D', G'\rangle \leftarrow$ absorb_gcis($\langle D', G'\rangle$,*true*)
5: $\langle D', G'\rangle \leftarrow$ transform_absorb_gcis($\langle D', G'\rangle$,*true*)
6: $\langle D', G'\rangle \leftarrow$ regroup_gcis($\langle D', G'\rangle$)
7: **return** $\langle D', G'\rangle$

The procedure absorb_gcis (whose definition is not shown here) is called with two parameters. The first parameter is the tuple containing the sets D, G, the second one is named *use_unfolding* and explained below. The procedure absorb_gcis iteratively transforms the GCIs in G and/or removes them from G and adds them to D. The process stops and returns G and D if no GCI transformation rule is applicable anymore. The process basically employs the transformation rules as given in [Horrocks, 1997]. We extended this scheme with the parameter *use_unfolding* controlling whether defined concepts may be unfolded by the transformation rules.

The procedure absorb_defined_concepts recognizes in G pairs of inclusions of the form $\{\mathsf{C} \sqsubseteq \mathsf{D}, \mathsf{D} \sqsubseteq \mathsf{C}\}$ where C is a concept name. These pairs are removed from G and appropriate concept definitions are added to D provided the concepts C, D are not cyclic.

The procedure transform_absorb_gcis also has a second parameter controlling whether unfolding may be applied. It iteratively transforms elements from D or G by splitting defined

Procedure 2 absorb_defined_concepts($\langle D, G\rangle$)

$D' \leftarrow D$; $G' \leftarrow G$
for all $\{\mathsf{C} \sqsubseteq \mathsf{D}, \mathsf{D} \sqsubseteq \mathsf{C}\} \in G$ **do**
  if $\mathsf{C} \in C \wedge \neg\text{cyclic}(\{\mathsf{C}, \mathsf{D}\})$ **then**
    $G' \leftarrow G' \setminus \{\mathsf{C} \sqsubseteq \mathsf{D}\}$
    $D' \leftarrow D' \cup \{\mathsf{C} \doteq \mathsf{D}\}$
  end if
end for
return $\langle D', G'\rangle$

Procedure 3 transform_absorb_gcis($\langle D, G\rangle$,use_unfolding)

$D' \leftarrow D$; $G' \leftarrow G$
while $G' \neq \emptyset$ **do**
  $G'' \leftarrow G'$
  $\langle D', G'\rangle \leftarrow$ split_concept_definitions($\langle D', G'\rangle$)
  $\langle D', G'\rangle \leftarrow$ absorb_gcis($\langle D', G'\rangle$,use_unfolding)
  $\langle D', G'\rangle \leftarrow$ split_gcis($\langle D', G'\rangle$)
  $\langle D', G'\rangle \leftarrow$ absorb_gcis($\langle D', G'\rangle$,use_unfolding)
  until $G' = G''$
end while
return $\langle D', G'\rangle$

concepts into two equivalent GCIs or by splitting a GCI (e.g. $\mathsf{C} \sqsubseteq \mathsf{D_1} \sqcap \ldots \sqcap \mathsf{D_n}$) from G into its equivalent parts (e.g. $\{\mathsf{C} \sqsubseteq \mathsf{D_1}, \ldots, \mathsf{C} \sqsubseteq \mathsf{D_n}\}$) using the distributive law.

The procedure split_concept_definitions splits a definition of the form $\mathsf{C} \doteq \mathsf{D_1} \sqcap \ldots \sqcap \mathsf{D_n}$ into two GCIs provided there exists a $\mathsf{D_i} \sqsubseteq \mathsf{E} \in G$.

Procedure 4 split_concept_definitions($\langle D, G \rangle$)

$D' \leftarrow D$; $G' \leftarrow G$
for all $\mathsf{C} \doteq \mathsf{D_1} \sqcap \ldots \sqcap \mathsf{D_n} \in D$ **do**
  if $\mathsf{D_i} \sqsubseteq \mathsf{E} \in G$ **then**
    $D' \leftarrow D' \setminus \{\mathsf{C} \doteq \mathsf{D_1} \sqcap \ldots \sqcap \mathsf{D_n}\}$
    $G' \leftarrow G' \cup \{\mathsf{C} \sqsubseteq \mathsf{D_1} \sqcap \ldots \sqcap \mathsf{D_n}, \mathsf{D_1} \sqcap \ldots \sqcap \mathsf{D_n} \sqsubseteq \mathsf{C}\}$
  end if
end for
return $\langle D', G' \rangle$

The procedure split_gcis splits a GCI of the form $\mathsf{C} \sqsubseteq \mathsf{D_1} \sqcap \ldots \sqcap \mathsf{D_n}$ into $\{\mathsf{C} \sqsubseteq \mathsf{D_1}, \ldots, \mathsf{C} \sqsubseteq \mathsf{D_n}\}$ if there exists a $\mathsf{D_i}$ $(1 \leq \mathsf{i} \leq \mathsf{n})$ of the form $\neg\mathsf{E}$ or $\mathsf{E_1} \sqcup \ldots \sqcup \mathsf{E_m}$. In the first case the GCI $\mathsf{C} \sqsubseteq \neg\mathsf{E}$ can be rewritten as $\mathsf{E} \sqsubseteq \neg\mathsf{C}$ and possibly be absorbed. In the second case the gci $\mathsf{C} \sqsubseteq \mathsf{E_1} \sqcup \ldots \sqcup \mathsf{E_m}$ might be subject to further transformations.

Procedure 5 split_gcis($\langle D, G \rangle$)

$G' \leftarrow G$
for all $\mathsf{C} \sqsubseteq \mathsf{D_1} \sqcap \ldots \sqcap \mathsf{D_n} \in G$ **do**
  if $\exists\, \mathsf{D_i} : \mathsf{i} \in 1..\mathsf{n}$, $((\mathsf{D_i} = \neg\mathsf{E}) \vee (\mathsf{D_i} = \mathsf{E_1} \sqcup \ldots \sqcup \mathsf{E_m}))$ **then**
    $G' \leftarrow (G' \setminus \{\mathsf{C} \sqsubseteq \mathsf{D_1} \sqcap \ldots \sqcap \mathsf{D_n}\}) \cup \{\mathsf{C} \sqsubseteq \mathsf{D_1}, \ldots, \mathsf{C} \sqsubseteq \mathsf{D_n}\}$
  end if
end for
return $\langle D, G' \rangle$

The procedure regroup_gcis applies the distributive law to elements of G in order to combine GCIs with identical right parts. This heuristics is advantageous for the Jeroslav-Wang technique (see [Freeman, 1995]) used to select a disjunct from a disjunction.

Procedure 6 regroup_gcis($\langle D, G \rangle$)

$G' \leftarrow G$
for all $\{\mathsf{C_1} \sqsubseteq \mathsf{D}, \ldots, \mathsf{C_n} \sqsubseteq \mathsf{D}\} \subseteq G$ **do**
  $G' \leftarrow G' \setminus \{\mathsf{C_1} \sqsubseteq \mathsf{D}, \ldots, \mathsf{C_n} \sqsubseteq \mathsf{D}\}$
  $G' \leftarrow G' \cup \{\mathsf{C_1} \sqcap \ldots \sqcap \mathsf{C_n} \sqsubseteq \mathsf{D}\}$
end for
return $\langle D, G' \rangle$

The procedure standard_gci_absorption uses none of the enhanced techniques.

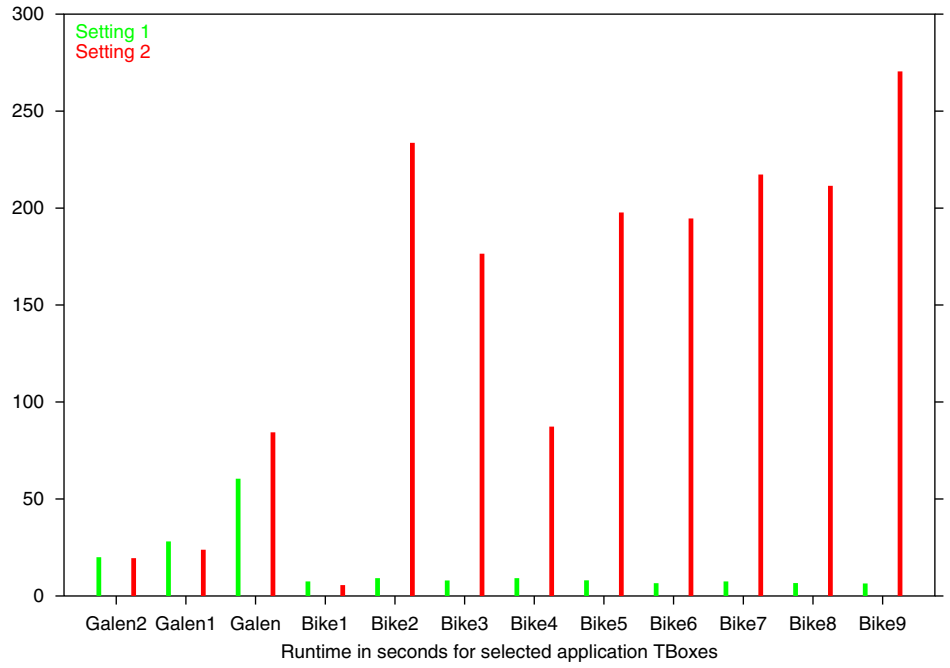


Figure 10.3: Evaluation of the enhanced GCI absorption technique. Setting 1 with enhanced_gci_absorption; Setting 2 with standard_gci_absorption.

Procedure 7 standard_gci_absorption($\langle D, G\rangle$)

return absorb_gcis($\langle D, G\rangle$,*true*)

The enhanced GCI absorption technique is empirically evaluated with selected application TBoxes. The results are shown in Figure 10.3. Two settings were used: 'Setting 1' has all optimizations enabled, i.e. the procedure enhanced_gci_absorption is used. 'Setting 2' is identical to 'Setting 1' except the procedure standard_gci_absorption is used instead of enhanced_gci_absorption. The application of the enhanced GCI absorption technique indicates a speed gain of at least one order of magnitude for the 'Bike' TBoxes. A slight speed gain for the Galen TBox is observed, while the other two 'Galen' TBoxes demonstrate a minor slow down due to an increased overhead since they do not contain GCIs. The other application TBoxes as described in Section 10.2.6 are either not affected by this optimization technique or show only a slight improvement.

10.4 Signature Calculus for $\mathcal{ALCNH}_{R^+}$

This section investigates a novel optimization technique, the so-called signature calculus, for the description logic $\mathcal{ALCNH}_{R^+}$. The signature calculus addresses a major source of inefficiency in the original tableaux calculus (see Chapter 9) that is caused by large numbers occurring in at-least or at-most concepts. Due to the expressivity of role hierarchies, it is not

possible to deal with number restrictions in a simple way by considering one representative individual instead of a number of individuals.

This source of inefficiency is illustrated with the following TBox (let $\mathsf{S_1}, \mathsf{S_2}, \mathsf{S_3}, \mathsf{R}$ be role names).

$\mathsf{S_1} \sqsubseteq \mathsf{R}$

$\mathsf{S_2} \sqsubseteq \mathsf{R}$

$\mathsf{S_3} \sqsubseteq \mathsf{R}$

$\mathsf{C} \sqsubseteq \exists_{\geq 10}\, \mathsf{S_1} \sqcap \exists_{\geq 10}\, \mathsf{S_2} \sqcap \exists_{\geq 10}\, \mathsf{S_3} \sqcap \forall\, \mathsf{S_1} \,.\, (\mathsf{A} \sqcap \mathsf{B}) \sqcap \forall\, \mathsf{S_2} \,.\, \neg\mathsf{A} \sqcap \forall\, \mathsf{S_3} \,.\, (\neg\mathsf{A} \sqcap \neg\mathsf{B}) \sqcap \exists_{\leq 2}\, \mathsf{R}$

In order to test the satisfiability of the concept C, the calculus from Section 9.4 creates for each role $\mathsf{S_i}$, $\mathsf{i} \in 1..3$ 10 $\mathsf{S_i}$-successors, i.e. we get 30 R-successors but only 2 R-successors are allowed. The all-restrictions ensure that only $\mathsf{S_2}$-successors and $\mathsf{S_3}$-successors can be successfully merged by the R$\exists_{\leq n}$-rule from Section 9.4.1. The concept C is still satisfiable if we replace in the example TBox every occurrence of 10 by 100, or by 1000, etc. However, the R$\exists_{\leq n}$-rule has to be applied to an increasing number of individuals causing a combinatorial explosion.

The signature calculus is advantageous since it offers a compact representation for large numbers of role successors caused by at-least concepts. Informally speaking, the new calculus is based on the idea to generate only a *single proxy role successor* for roles (and their "conjunctions") occurring in number restrictions. Due to the complexity caused by the combination of role hierarchies and number restrictions there is already a dramatic speed-up for values below of ten occurring in number restrictions.

The signature calculus for deciding the ABox consistency for $\mathcal{ALCNH}_{R^+}$ is introduced in the following. The presentation is very much oriented at the one given in Chapter 9. Some of the definitions and the completion rules dealing with role successors as described in Chapter 9 are replaced by new or adapted definitions or completion rules. The calculus is characterized by a set of tableaux or *completion* rules and by a particular *completion strategy* ensuring a specific order for applying the completion rules to assertional axioms of an ABox. The strategy is essential to guarantee the completeness of the ABox consistency algorithm. First, we have to introduce new assertional axioms needed to define the augmentation of an ABox.

Definition 10.1 (Additional ABox Assertions) Let $\mathsf{C} \in C$, $\mathsf{a}, \mathsf{b} \in O$, $RS \subseteq R$, $n \geq 0$, and $x \notin O$, then the following expressions are also assertional axioms:

- $\forall\, x \,.\, (x : \mathsf{C})$ *(universal concept assertion)*, and
- $(\mathsf{a}, \mathsf{b}) : \langle n, RS \rangle$ *(signature assertion)*.

In order to deal with the fact that a proxy individual represents a set of individuals, a slightly modified definition of an interpretation is used. The interpretation function $\cdot^{\mathcal{I}}$ of

the interpretation $\mathcal{I}$ can be extended by additionally mapping every individual name from O to a *subset* of $\Delta^{\mathcal{I}}$ in a way such that it satisfies

- $\|\mathsf{a}^{\mathcal{I}}\| = 1$ if $\mathsf{a} \in O_O$,
- $\mathsf{a}^{\mathcal{I}} \neq \mathsf{b}^{\mathcal{I}}$ if $\mathsf{a}, \mathsf{b} \in O_O$ and $\mathsf{a} \neq \mathsf{b}$ (*unique name assumption*),
- $\mathsf{a} : \mathsf{C}$ iff $\mathsf{a}^{\mathcal{I}} \subseteq \mathsf{C}^{\mathcal{I}}$,
- $(\mathsf{a}, \mathsf{b}) : \mathsf{R}$ iff $\forall\, x \in \mathsf{a}^{\mathcal{I}},\ y \in \mathsf{b}^{\mathcal{I}} : (x, y) \in \mathsf{R}^{\mathcal{I}}$,
- $\forall\, x\,.\,(x : \mathsf{C})$ iff $\mathsf{C}^{\mathcal{I}} = \Delta^{\mathcal{I}}$,
- $(\mathsf{a}, \mathsf{b}) : \langle n, RS \rangle$ iff $\forall x \in \mathsf{a}^{\mathcal{I}},\ y \in \mathsf{b}^{\mathcal{I}} : (x, y) \in \bigcap_{\mathsf{R} \in RS} \mathsf{R}^{\mathcal{I}}$ and $\|\mathsf{b}^{\mathcal{I}}\| = n$.

The completion rules for signature assertions require a dedicated operator '$\widetilde{\sqcap}$' for well-formed sets of role names.

Definition 10.2 (Role Set, Role Conjunction) A *well-formed role set* contains either a single role name or the direct parents of an anonymous (most specific) "role conjunction" whose canonical name is represented by this set. The set resulting from '$\widetilde{\sqcap}$' represents the "role conjunction" of its operands. Let $RS_1, RS_2 \subseteq R$ and $RS = RS_1 \cup RS_2$, then the role conjunction is defined as $RS_1 \,\widetilde{\sqcap}\, RS_2 = \{\mathsf{R} \in RS \mid \neg\exists\, \mathsf{S} \in RS : \mathsf{S} \in (\mathsf{R}^{\downarrow} \setminus \{\mathsf{R}\})\}$, i.e. the "role conjunction" is the union of both role sets without implied superroles.

For instance, consider the role hierarchy example introduced above. The "role conjunction" $\{R\} \,\widetilde{\sqcap}\, \{S_1, S_2\}$ yields $\{\mathsf{S_1}, \mathsf{S_2}\}$, i.e. R is not a member of the new set because R is already implied by at least one member of the new set (e.g. $\mathsf{S_1} \sqsubseteq \mathsf{R}$), while $\{S_1\} \,\widetilde{\sqcap}\, \{S_2, S_3\}$ yields $\{\mathsf{S_1}, \mathsf{S_2}, \mathsf{S_3}\}$. We are now ready to define an augmented ABox as input to the new tableaux rules.

Definition 10.3 (Augmented ABox) For an initial ABox $\mathcal{A}$ w.r.t a TBox $\mathcal{T}$ we define its *augmented* ABox $\mathcal{A}'$ by applying the following rules to $\mathcal{A}$. 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. Every assertion of the form $(\mathsf{a}, \mathsf{b}) : \mathsf{R}$ is replaced by $(\mathsf{a}, \mathsf{b}) : \langle 1, \{\mathsf{R}\}\rangle$ and every pair of assertions of the form $(\mathsf{a}, \mathsf{b}) : \langle 1, RS_1\rangle, (\mathsf{a}, \mathsf{b}) : \langle 1, RS_2\rangle$, $RS_1 \neq RS_2$ is replaced by $(\mathsf{a}, \mathsf{b}) : \langle 1, RS_1 \,\widetilde{\sqcap}\, RS_2\rangle$ as long as possible. From this point on, if we refer to an initial ABox $\mathcal{A}$ we always mean its augmented ABox.

The tableaux rules also require the notion of *blocking* their applicability. This is based on so-called concept sets, an ordering for new individuals, and blocking individuals (see Definitions 9.10-9.12).

10.4.1 Completion Rules

The completion rules create and merge signature assertions. If a signature assertion $(\mathsf{a},\mathsf{b}):\langle n, RS\rangle$ is element of an ABox $\mathcal{A}$, we say it represents n "identical" role successors whose proxy is named as b.

Definition 10.4 (Potential R-successors) Given an ABox $\mathcal{A}$, $\sharp(\mathsf{a},\mathsf{R})_\mathcal{A}$ defines the number of potential R-successors for an individual a mentioned in $\mathcal{A}$.

$$\sharp(\mathsf{a},\mathsf{R})_\mathcal{A} = \sum_{\alpha\in\mathcal{A}} \mathrm{count}(\mathsf{a},\mathsf{R},\alpha)$$

$$\mathrm{count}(\mathsf{a},\mathsf{R},\alpha) = \begin{cases} n & \text{if } \alpha = (\mathsf{a},\mathsf{b}):\langle n, RS\rangle,\ \mathsf{R}\in RS^{\uparrow}, \\ 0 & \text{otherwise.}\end{cases}$$

Definition 10.5 (Lower and Upper Bound for R-successors) For a given ABox $\mathcal{A}$, $\min(\mathsf{a},\mathsf{R})_\mathcal{A}$ defines the minimal number of required and $\max(\mathsf{a},\mathsf{R})_\mathcal{A}$ the maximal number of allowed R-successors for an individual a mentioned in $\mathcal{A}$.

$$\begin{aligned}\min(\mathsf{a},\mathsf{R})_\mathcal{A} = \max(&\{0\}\cup\{n\,|\,\mathsf{a}:\exists_{\geq n}\,\mathsf{S}\in\mathcal{A},\ \mathsf{S}\in\mathsf{R}^{\downarrow}\}\cup\\ &\{1\,|\,\exists\,\mathsf{S}\in\mathsf{R}^{\downarrow}:\mathsf{a}:\exists\,\mathsf{S}\,.\,\mathsf{C}\in\mathcal{A}\}\cup\\ &\{\|\{\mathsf{b}\in O_O\,|\,(\mathsf{a},\mathsf{b}):\langle 1, RS\rangle\in\mathcal{A},\ \mathsf{R}\in RS^{\uparrow}\}\|\})\\ \max(\mathsf{a},\mathsf{R})_\mathcal{A} = \min(&\{\infty\}\cup\{n\,|\,\mathsf{a}:\exists_{\leq n}\,\mathsf{S}\in\mathcal{A},\ \mathsf{S}\in\mathsf{R}^{\uparrow}\})\end{aligned}$$

Due to the unique name assumption for old individuals, the number of old R-successors has to be considered by the definition of $\min(\mathsf{a},\mathsf{R})_\mathcal{A}$.

We are now ready to define the *completion rules* that are intended to generate a so-called completion (see also below) of an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$.

Definition 10.6 (Completion Rules)

SigEmpty The signature cleanup rule.
if $(\mathsf{a},\mathsf{b}):\langle 0, RS\rangle\in\mathcal{A}$
then $\mathcal{A}' = \mathcal{A}\setminus(\{(\mathsf{a},\mathsf{b}):\langle 0, RS\rangle\}\cup\{\mathsf{b}:\mathsf{C}\,|\,\mathsf{b}:\mathsf{C}\in\mathcal{A}\})$

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. $\exists\,\mathsf{b} \in O : (\mathsf{a},\mathsf{b}){:}\langle n, RS\rangle \in \mathcal{A},\ \mathsf{R} \in RS^{\uparrow}$, 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. $\exists\,\mathsf{b} \in O,\ \mathsf{T} \in \mathsf{R}^{\downarrow} \cap RS^{\uparrow} \cap T : (\mathsf{a},\mathsf{b}){:}\langle n, RS\rangle \in \mathcal{A}$, 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 (for signatures).
if 1. $\mathsf{a}{:}\exists\,\mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}$, and
2. a is not blocked, and
3. $\neg\exists\,\mathsf{b} \in O : \{(\mathsf{a},\mathsf{b}){:}\langle n, RS\rangle,\ \mathsf{b}{:}\mathsf{C}\} \subseteq \mathcal{A},\ \mathsf{R} \in RS^{\uparrow}$
then $\mathcal{A}' = \mathcal{A} \cup \{(\mathsf{a},\mathsf{b}){:}\langle 1, \{\mathsf{R}\}\rangle,\ \mathsf{b}{:}\mathsf{C}\},\ \mathsf{b} \in O_N$ new in $\mathcal{A}$.

R∃$_{\geq n}$ The number restriction exists rule (for signatures).
if 1. $\mathsf{a}{:}\exists_{\geq n}\,\mathsf{R} \in \mathcal{A}$, and
2. a is not blocked, and
3. $\sharp(\mathsf{a},\mathsf{R})_{\mathcal{A}} < n$
then $\mathcal{A}' = \mathcal{A} \cup \{(\mathsf{a},\mathsf{b}){:}\langle n, \{\mathsf{R}\}\rangle\},\ \mathsf{b} \in O_N$ new in $\mathcal{A}$.

SigMerge The signature merge rule.

if 1. $\exists\, \mathsf{a}$ mentioned in $\mathcal{A}$, $\mathsf{R} \in P : \sharp(\mathsf{a},\mathsf{R})_{\mathcal{A}} > \max(\mathsf{a},\mathsf{R})_{\mathcal{A}}$, and
2. $\mathcal{M}_{\mathsf{R}} = \{\alpha \in \mathcal{A} \mid \alpha = (\mathsf{a},\mathsf{b}) : \langle n, RS\rangle, \mathsf{R} \in RS^{\uparrow}\}$

then select $\{(\mathsf{a},\mathsf{b}_1) : \langle n_1, RS_1\rangle,\ (\mathsf{a},\mathsf{b}_2) : \langle n_2, RS_2\rangle\} \subseteq \mathcal{M}_{\mathsf{R}}$ such that
1. $\mathsf{b}_1 \neq \mathsf{b}_2$, and
2. if $RS_1 = RS_2$ then $\forall\, \mathsf{S} \in RS_1 : \sharp(\mathsf{a},\mathsf{S})_{\mathcal{A}} > \min(\mathsf{a},\mathsf{S})_{\mathcal{A}}$, and
3. either $\mathsf{b}_1, \mathsf{b}_2 \in O_N$ or $\mathsf{b}_1 \in O_N$, $\mathsf{b}_2 \in O_O$

if $\mathsf{b}_1, \mathsf{b}_2 \in O_N$
then
if $\exists\, i, j \in 1..2 : RS_i \subseteq RS_j,\ i \neq j$
then (case a: *decrement superrole signature*)
$\mathcal{A}' = (\mathcal{A} \setminus \{(\mathsf{a},\mathsf{b_i}) : \langle n_i, RS_i\rangle\}) \cup \{(\mathsf{a},\mathsf{b_i}) : \langle n_i - 1, RS_i\rangle\} \cup \{\mathsf{b_j} : \mathsf{C} \mid \mathsf{C} \in \sigma(\mathcal{A},\mathsf{b_i})\}$
elsif $\exists\, \mathsf{c} \in O_N : (\mathsf{a},\mathsf{c}) : \langle n, RS_1 \widetilde{\sqcap} RS_2\rangle \in \mathcal{M}_{\mathsf{R}}$
then (case b: *increment common subrole signature*)
$\mathcal{A}' = (\mathcal{A} \setminus \{(\mathsf{a},\mathsf{b}_1) : \langle n_1, RS_1\rangle, (\mathsf{a},\mathsf{b}_2) : \langle n_2, RS_2\rangle, (\mathsf{a},\mathsf{c}) : \langle n, RS_1 \widetilde{\sqcap} RS_2\rangle\}) \cup$
$\{(\mathsf{a},\mathsf{b}_1) : \langle n_1 - 1, RS_1\rangle, (\mathsf{a},\mathsf{b}_2) : \langle n_2 - 1, RS_2\rangle, (\mathsf{a},\mathsf{c}) : \langle n + 1, RS_1 \widetilde{\sqcap} RS_2\rangle\} \cup$
$\{\mathsf{c} : \mathsf{C} \mid \mathsf{C} \in \sigma(\mathcal{A},\mathsf{b}_1) \cup \sigma(\mathcal{A},\mathsf{b}_2)\}$
else (case c: *create common subrole signature*)
$\mathcal{A}' = (\mathcal{A} \setminus \{(\mathsf{a},\mathsf{b}_1) : \langle n_1, RS_1\rangle, (\mathsf{a},\mathsf{b}_2) : \langle n_2, RS_2\rangle\}) \cup$
$\{(\mathsf{a},\mathsf{b}_1) : \langle n_1 - 1, RS_1\rangle, (\mathsf{a},\mathsf{b}_2) : \langle n_2 - 1, RS_2\rangle, (\mathsf{a},\mathsf{c}) : \langle 1, RS_1 \widetilde{\sqcap} RS_2\rangle\} \cup$
$\{\mathsf{c} : \mathsf{C} \mid \mathsf{C} \in \sigma(\mathcal{A},\mathsf{b}_1) \cup \sigma(\mathcal{A},\mathsf{b}_2)\}$, $\mathsf{c} \in O_N$ new in $\mathcal{A}$
else (case d: *merge with old individual*)
$\mathcal{A}' = (\mathcal{A} \setminus \{(\mathsf{a},\mathsf{b}_1) : \langle n, RS_1\rangle, (\mathsf{a},\mathsf{b}_2) : \langle 1, RS_2\rangle\}) \cup$
$\{(\mathsf{a},\mathsf{b}_1) : \langle n - 1, RS_1\rangle, (\mathsf{a},\mathsf{b}_2) : \langle 1, RS_1 \widetilde{\sqcap} RS_2\rangle\} \cup \{\mathsf{b}_2 : \mathsf{C} \mid \mathsf{C} \in \sigma(\mathcal{A},\mathsf{b}_1)\}$

Empty signatures and the corresponding concept assertions for their role successors are removed by the clean-up rule SigEmpty. We call the rules R$\sqcup$ and SigMerge *nondeterministic* rules since they can be applied in different ways to the same set of assertions. 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 the only rules that increase in an ABox the total number of role successors of already existing individuals. If the signature merge rule has been applied to an individual a and a non-transitive role R in an ABox $\mathcal{A}$, then it holds that $\sharp(\mathsf{a},\mathsf{R})_{\mathcal{A}'} = \sharp(\mathsf{a},\mathsf{R})_{\mathcal{A}} - 1$.

The rule SigMerge is quite complex and needs an explanation. The rule SigMerge is applicable to an individual a if a has more potential R-successors than allowed by applicable at most restrictions and there exists a set $\mathcal{M}_{\mathsf{R}}$ containing the signature assertions for the descendants of the role R. Then, a pair of assertions is indeterministically selected such that the following conditions hold. If the role sets of the two signature assertions are equal, it is necessary that the number of successors for all members of the role sets is greater than the required minimum. Otherwise the rule SigMerge (case a) would decrease the number of successors and violate a minimum restriction. The second condition restricts the set of eligible pairs to assertions where the proxy individuals are either both new individuals or one is a new individual and the second one is an old individual. Due to the unique name

assumption two signature assertions for old "proxy" individuals may never be merged.

If these conditions are met for a selected pair of signature assertions, then the SigMerge rule distinguishes four mutually exclusive cases (a-d). If both proxy individuals are new ones, then the cases a-c are considered, otherwise the case d is applicable.

Case a: If one role set is a subset or equal to the second role set, then the counter of the (super)role signature (with the smaller set) is decremented by 1 and the concept assertions for the proxy individual of the (super)role signature are added to the proxy individual of the (sub)role signature.

Case b: If there already exists a role conjunction signature, then decrement the counters of the signature pair, increment the counter of the role conjunction signature, and add the concept assertions of the proxy individuals of the signature pair to the proxy individual of the role conjunction signature.

Case c: This case corresponds to case b but a new role conjunction signature with counter value 1 is added.

Case d: The proxy individual b_2 is an old individual and due to the unique name assumption the counter of the signature assertion with the proxy individual b_1 is decremented and the concept assertions for b_1 are added to b_2.

Given an initial ABox $\mathcal{A}$, more than one rule might be applicable to $\mathcal{A}$. This is controlled by a completion strategy in accordance with the ordering for new individuals (see Definition 9.11).

Definition 10.7 (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 with the following order (decreasing precedence). A rule may only be applied if no rule with a higher precedence is applicable.
  1. Apply the SigEmpty rule as long as possible.
  2. If possible, apply a nongenerating rule (R$\sqcap$, R$\sqcup$, R$\forall$C, R$\forall_+$C, R$\forall_x$, SigMerge) and return to step 1.
  3. If possible, apply a generating rule (R$\exists$C, R$\exists_{\geq n}$) and return to step 1.

In the following we always assume that rules are applied in accordance with this strategy. It ensures that the rules are applied to new individuals w.r.t. the ordering '$\prec$'. The strategy guarantees the tableaux expansion w.r.t. role successors in breadth-first order.

The calculus also has to check whether so-called clash triggers are applicable.

Definition 10.8 (Clash Triggers) We assume the same naming conventions as used above. Let $\mathcal{A}$ be an ABox and $\mathcal{A}'$ be its augmented ABox. The ABoxes $\mathcal{A}, \mathcal{A}'$ are called *contradictory* if one of the following *clash triggers* is applicable to $\mathcal{A}'$. If none of the clash triggers is applicable to $\mathcal{A}'$, then $\mathcal{A}$ and $\mathcal{A}'$ are called *clash-free*.

- *Primitive clash*: $\mathsf{a}:\bot \in \mathcal{A}'$ or $\{\mathsf{a}:\mathsf{A}, \mathsf{a}:\neg\mathsf{A}\} \subseteq \mathcal{A}'$, where A is a concept name
- *Number restriction clash*: $\exists\,\mathsf{a}$ mentioned in $\mathcal{A}'$, $\mathsf{R} \in P : \min(\mathsf{a},\mathsf{R})_{\mathcal{A}'} > \max(\mathsf{a},\mathsf{R})_{\mathcal{A}'}$

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 whose augmented ABox contains a clash is obviously inconsistent. The purpose of the calculus is to generate a completion for an initial ABox $\mathcal{A}$ in order to prove the consistency of $\mathcal{A}$ or the inconsistency of $\mathcal{A}$ if no completion can be derived.

10.4.2 Decidability of the ABox Consistency Problem

A careful analysis of the new tableaux rules presented in this section supports the conjecture that this new calculus is sound, complete, and terminates for any initial ABox. The formal proof of this conjecture is subject to ongoing work (see [Haarslev et al., 2000a]).

10.4.3 Further Enhancements

The actual implementation of the signature calculus in RACE is more advanced than described above. The signature merge rules are only applied to partitioned ABoxes (see below). The generating rules create no "redundant" signature assertions (e.g. if the assertions $\{\mathsf{a}:\exists_{\geq 2}\,\mathsf{R}, \mathsf{a}:\exists_{\geq 3}\,\mathsf{R}\}$ are a subset of an ABox, it is sufficient to create only a signature assertion for the second concept assertion). The signature calculus also takes care of dependency-directed backtracking whenever it is applicable.

The signature calculus can safely split a "subtableaux ABox" $\mathcal{A}$ (mentioning no old individuals) into independent *partitions* or subtableaux[7] w.r.t. concept assertions of the form $\mathsf{a}:\exists\,\mathsf{S}\,.\,\mathsf{C}$ or $\mathsf{a}:\exists_{\geq n}\,\mathsf{S}$, if a concept assertion of the form $\mathsf{a}:\exists_{\leq n}\,\mathsf{R}$ with $\mathsf{R} \in \mathsf{S}^{\uparrow}$ is present. A partition $P_{\hat{\mathsf{R}}}$ is constrained by a "root role" $\hat{\mathsf{R}} \in \hat{R}$ where $\hat{R}$ is defined as follows.

$$\hat{R} = \{\mathsf{R} \in R \mid \mathsf{a}:\exists_{\leq m}\,\mathsf{R} \in \mathcal{A}, \sharp(\mathsf{a},\mathsf{R})_{\mathcal{A}} > \max(\mathsf{a},\mathsf{R})_{\mathcal{A}}\}$$

[7] See also Section 10.2.3 on subtableaux caching.

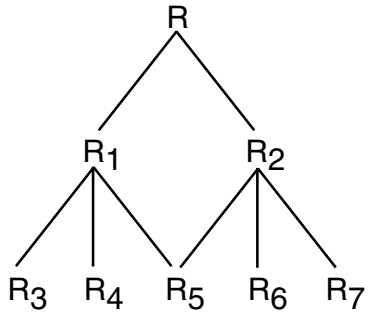


Figure 10.4: Role hierarchy.

A *partition* $P_{\hat{\mathsf{R}}}$ of an ABox $\mathcal{A}$ with $\hat{\mathsf{R}} \in \hat{R}$ can be defined as

$$P_{\hat{\mathsf{R}}} = \{\alpha \in \mathcal{A} \mid \exists\, \mathsf{S} \in \hat{\mathsf{R}}^{\downarrow} \colon (\alpha = \mathsf{a} \colon \exists\, \mathsf{S} \,.\, \mathsf{C}) \vee (\alpha = \mathsf{a} \colon \exists_{\geq n}\, \mathsf{S})\}.$$

The partition scheme is illustrated by the following ABox $\mathcal{A}$ with a role hierarchy as given in Figure 10.4.

$$\mathcal{A} = \left\{ \begin{array}{l} a \colon \exists_{\geq 2}\, \mathsf{R}_3, a \colon \exists_{\geq 3}\, \mathsf{R}_4, a \colon \exists_{\geq 4}\, \mathsf{R}_5, a \colon \exists_{\geq 3}\, \mathsf{R}_6, a \colon \exists_{\geq 2}\, \mathsf{R}_7, \\ a \colon \exists_{\leq n}\, \mathsf{R}_1, a \colon \exists_{\leq m}\, \mathsf{R}_2, \\ a \colon \forall\, \mathsf{R}_3 \,.\, \mathsf{P}_1, a \colon \forall\, \mathsf{R}_4 \,.\, \mathsf{P}_1, a \colon \forall\, \mathsf{R}_5 \,.\, \neg \mathsf{P}_1, a \colon \forall\, \mathsf{R}_6 \,.\, \mathsf{P}_1, a \colon \forall\, \mathsf{R}_7 \,.\, \neg \mathsf{P}_1 \end{array} \right\}$$

This ABox is consistent for values of $n, m \geq 7$ and inconsistent otherwise. The partitioning scheme applied to $\mathcal{A}$ results in $\hat{R} = \{\mathsf{R}_1, \mathsf{R}_2\}$ and the following partitions $P_{\mathsf{R}_1}, P_{\mathsf{R}_2}$.

$$P_{\mathsf{R}_1} = \left\{ a \colon \exists_{\geq 2}\, \mathsf{R}_3, a \colon \exists_{\geq 3}\, \mathsf{R}_4, a \colon \exists_{\geq 4}\, \mathsf{R}_5 \right\}$$
$$P_{\mathsf{R}_2} = \left\{ a \colon \exists_{\geq 4}\, \mathsf{R}_5, a \colon \exists_{\geq 3}\, \mathsf{R}_6, a \colon \exists_{\geq 2}\, \mathsf{R}_7 \right\}$$

The following two ABoxes $\mathcal{A}_1$ and $\mathcal{A}_2$ correspond to the partitions $P_{\mathsf{R}_1}, P_{\mathsf{R}_2}$ and can be proven independently from each other. The ABox $\mathcal{A}$ is consistent iff both $\mathcal{A}_1$ and $\mathcal{A}_2$ are consistent.

$$\mathcal{A}_1 = \left\{ \begin{array}{l} a \colon \exists_{\geq 2}\, \mathsf{R}_3, a \colon \exists_{\geq 3}\, \mathsf{R}_4, a \colon \exists_{\geq 4}\, \mathsf{R}_5, a \colon \exists_{\leq n}\, \mathsf{R}_1, \\ a \colon \forall\, \mathsf{R}_3 \,.\, \mathsf{P}_1, a \colon \forall\, \mathsf{R}_4 \,.\, \mathsf{P}_1, a \colon \forall\, \mathsf{R}_5 \,.\, \neg \mathsf{P}_1 \end{array} \right\}$$
$$\mathcal{A}_2 = \left\{ \begin{array}{l} a \colon \exists_{\geq 4}\, \mathsf{R}_5, a \colon \exists_{\geq 3}\, \mathsf{R}_6, a \colon \exists_{\geq 2}\, \mathsf{R}_7, a \colon \exists_{\leq m}\, \mathsf{R}_2, \\ a \colon \forall\, \mathsf{R}_5 \,.\, \neg \mathsf{P}_1, a \colon \forall\, \mathsf{R}_6 \,.\, \mathsf{P}_1, a \colon \forall\, \mathsf{R}_7 \,.\, \neg \mathsf{P}_1 \end{array} \right\}$$

10.4.4 Evaluation

In order to evaluate the effectiveness of an implemented version of the signature calculus, a set of four dedicated benchmark problems has been generated. The increased difficulty of the problems is caused by exponentially increasing the size of numbers used in at-least

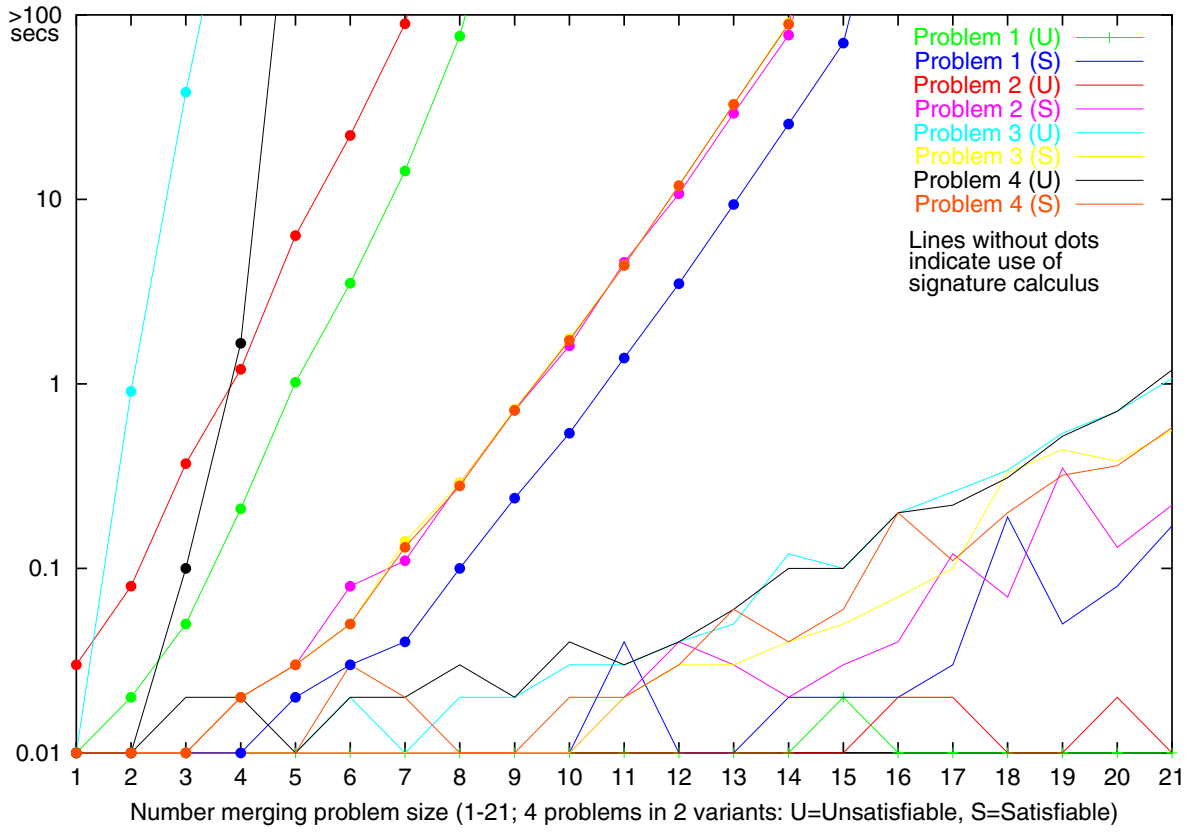


Figure 10.5: RACE: benchmark problems w/out signature calculus.

and at-most concepts which, in turn, cause an exponentially growing number of new concept and role assertions. Each of the four problems exists in two variants (a 'test concept' is satisfiable vs. unsatisfiable). The problems use role hierarchies and number restrictions. A problem basically employs concept terms of the form $\exists_{\leq n}\,\mathsf{R} \sqcap \exists_{\geq m_1}\,\mathsf{R}_1 \sqcap \exists_{\geq m_2}\,\mathsf{R}_2 \sqcap \exists_{\geq m_3}\,\mathsf{R}_3 \sqcap \forall\,\mathsf{R}_2\,.\,\mathsf{C} \sqcap \forall\,\mathsf{R}_3\,.\,\neg\mathsf{C}$ with $\mathsf{R}_i \sqsubseteq \mathsf{R}, i \in 1..3$. The (un)satisfiability of these terms has to be proven. A term is made satisfiable by choosing values for n, m_i such that $\max(m_1, m_2 + m_3) \leq n$ or unsatisfiable if $\max(m_1, m_2 + m_3) > n$. For instance, numbers occurring in at-least or at-most concepts of problem size 3 are usually below 7. Figure 10.5 (using a logarithmic scale) demonstrates the result of this benchmark with RACE. The use of the signature calculus indicates a dramatic performance gain of several orders of magnitude.

10.4.5 Summary and Discussion

In this chapter we presented an $\mathcal{ALCNH}_{R^+}$ calculus for dealing with number restrictions more efficiently. The signature calculus has been implemented in the ABox description logic system RACE. The signature calculus provides a better encoding of the same information

as the encoding given in Chapter 9 and, as a consequence, reduces the number of assertions the completion rules have to deal with. This, in turn, limits the combinatorial explosion caused by the completion rule which tries to merge role successors.

In the literature, some other techniques for dealing with number restriction have been discussed. In [Ohlbach and Köhler, 1999] an algebraic technique is introduced. The idea is to use an algorithm for solving linear inequalities in order to to check whether number constraints can be fulfilled. The algorithm requires that all possible role conjunctions in the context of a role hierarchy are properly considered. For all role conjunctions, the concept restrictions have to be computed. If, for a certain role conjunction, the imposed concept constraints cannot be satisfied, then an additional equation is derived and the system of inequalities might become unsatisfiable. Thus, in order to show the *satisfiability* of a concept term an exponential number of concept satisfiability tests are required even in the average case. To the best of our knowledge, an empirical analysis of the algorithm has not been presented. Experiences with description logic applications indicate that most satisfiability tests (e.g. for computing the concept hierarchy) return "satisfiable", and therefore, the costs of the appealing approach presented in [Ohlbach and Köhler, 1999] must not be underestimated.

It would be interesting to investigate a combination of both approaches: If the consistency of the concept restrictions for all role conjunctions has been computed by applying the rules of our signature calculus, then the approach presented in [Ohlbach and Köhler, 1999] could be used to show that backtracking does not lead to a solution. However, in [Ohlbach and Köhler, 1999] neither transitive roles nor ABoxes (nor axioms) are considered. Thus, the adaption of the techniques presented in [Ohlbach and Köhler, 1999] and the integration into the RACE architecture is left for future work.

We would like to emphasize that the signature calculus is also relevant in the context of propositional modal logics (for a subclass of graded modalities and for reasoning with nominals). We conjecture an easy adaption for dealing with so-called qualified number restrictions that correspond to graded modalities in modal logics.

10.5 Role Path Contraction

In order to make ABox realization as fast as possible we devised a transformation technique for ABoxes that maximizes the effect of caching techniques. Informally, the 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 10.6. The individual a is represented by a gray circle. A some-concept "representing" a contracted role path is

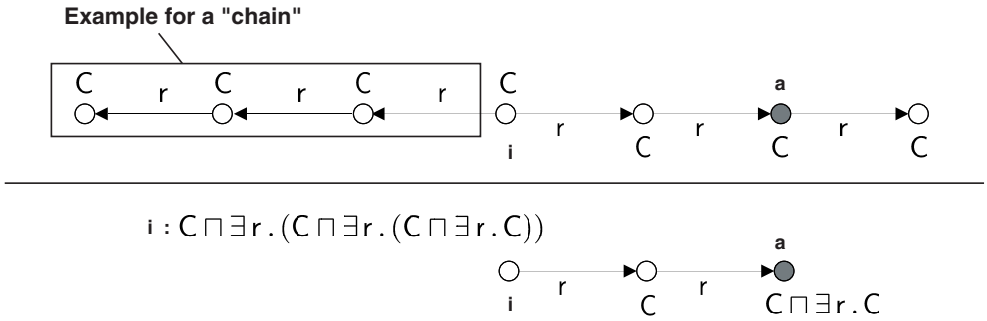


Figure 10.6: Example for ABox chain contraction (see text). The upper ABox is transformed into the lower one.

added as a concept assertion to the individual i 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 10.1 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.

Proof. (Sketch) 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. It is easy to see that, 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 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**.

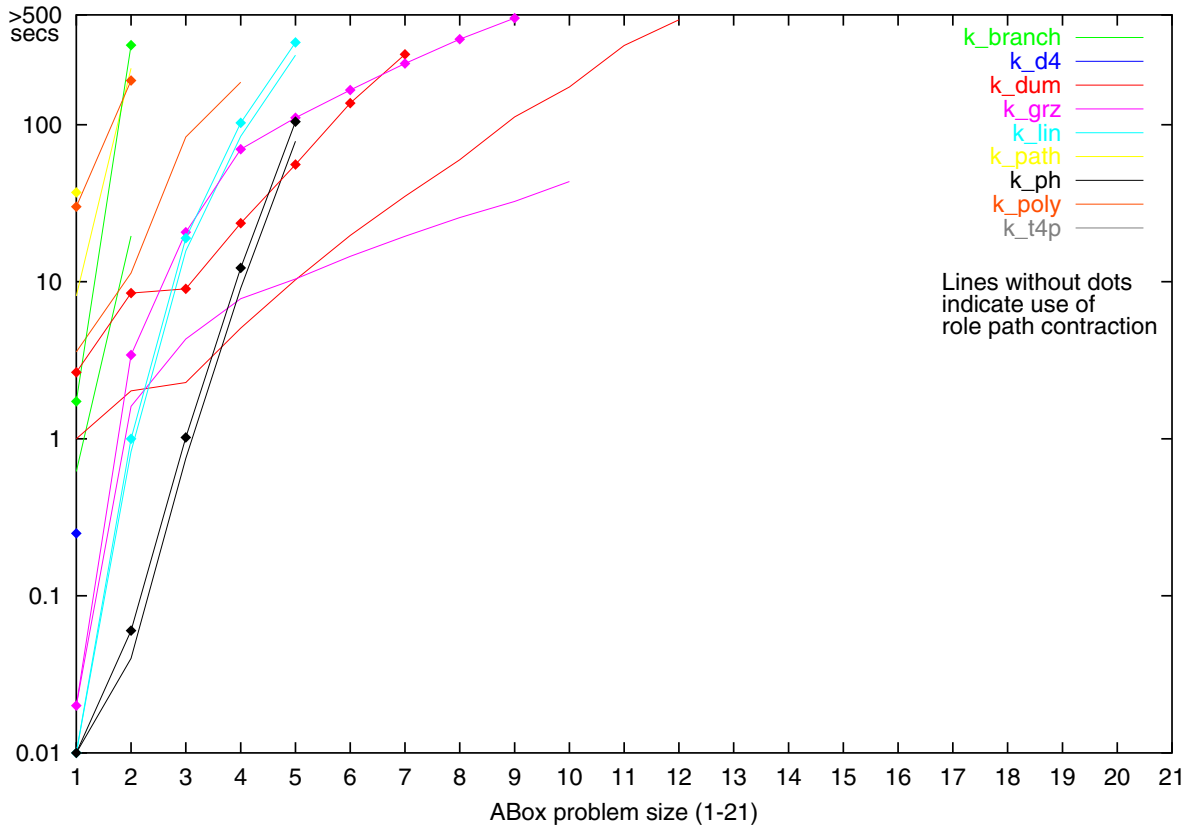


Figure 10.7: RACE: ABox realization w/out role path contraction.

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

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 10.2 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.

Proof. (Sketch) Given the proof sketch 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 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:

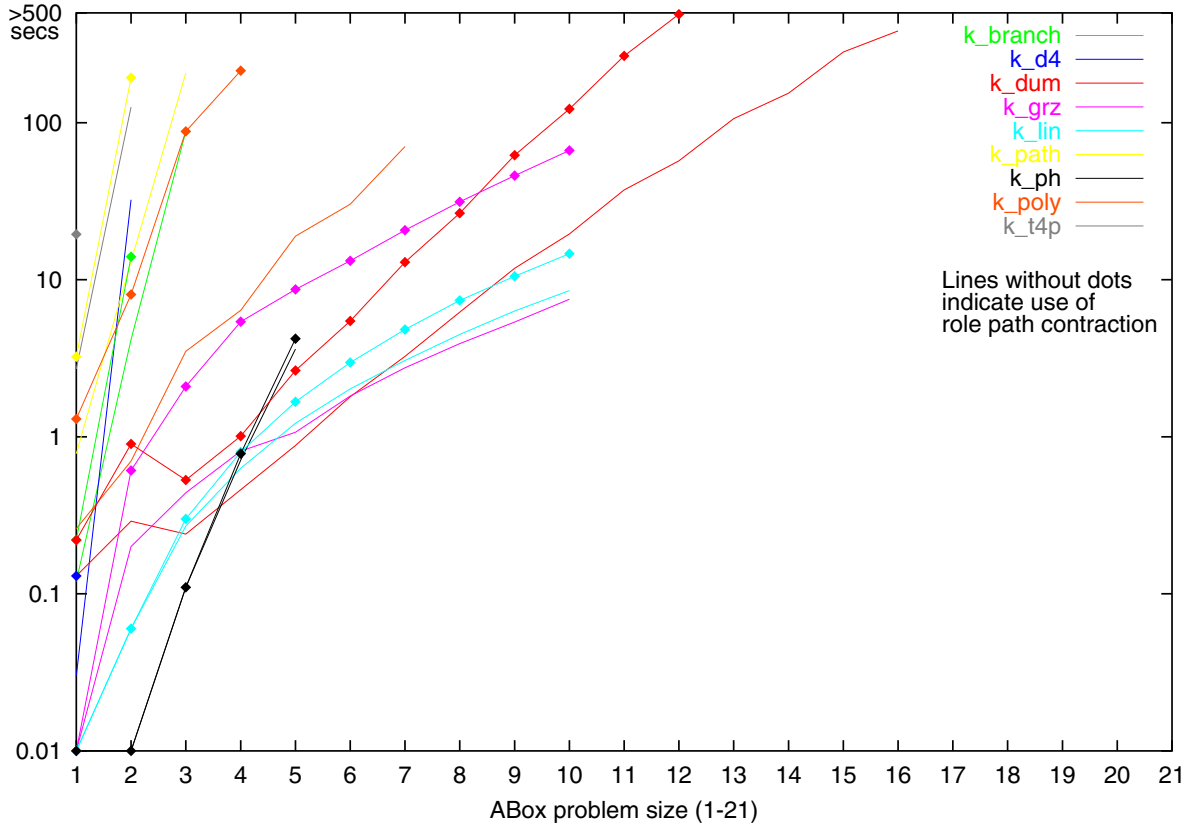


Figure 10.8: RACE: ABox instance checking w/out role path contraction.

1. The number of individuals and role assertions is reduced.

2. The some-concepts resulting from the contraction are automatically subject to the subtableaux caching (see Section 10.2.3) and pseudo model caching and merging techniques (see next section).

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.

The empirical tests with instance checking problems [Haarslev et al., 1999a] indicate that the role path contraction technique is very effective. The graph in Figure 10.7 shows the execution times for different problems with ABox realization and in Figure 10.8 with instance checking. The problems have a size increasing from 1 to 21. The runtime is supposed to grow exponentially if the problem size is incremented. It can be observed that the number of ABox assertions is reduced by several orders of magnitude due to the contraction technique. The improvement by this technique is indicated in Figures 10.7 and 10.8. It clearly shows that the contraction technique can give a speed gain of about one order of magnitude. However, many of these problems are still too hard and ask for

the design of even 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., 1994b] are not able to check ABox consistency (within 500 secs) for any of these problems except the simplest level (size 1) and even time out for some problems of this level.

10.6 Exploiting Deep Pseudo Models for TBox Reasoning

Given a set of concepts representing a conjunction whose satisfiability is to be checked, the *model merging strategy* tries to avoid a satisfiability test which relies on the "expensive" tableaux technique. This idea was first introduced in [Horrocks, 1997] for the logic $\mathcal{ALCH}f_{R^+}$. A model merging test is designed to be a "cheap" test operating on cached "concept models." It is a *sound* but incomplete satisfiability tester for a set of concepts. The achievement of minimal computational overhead and the avoidance of any indeterminism are important characteristics of such a test. If the test returns false, a tableaux calculus based on the rules as defined in Section 9.4 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.

For testing whether the conjunction of a set of concepts $\{\mathsf{C}_1, \ldots, \mathsf{C}_n\}$ is satisfiable, we present and analyze a technique called *deep model merging* that that generalizes the original model merging approach [Horrocks, 1997] in two ways.

- We extend the model merging technique to the logic $\mathcal{ALCNH}_{R^+}$, i.e. this technique also deals with number restrictions.
- We introduce *deep* pseudo models (e.g. for the concepts $\{\mathsf{C}_1, \ldots, \mathsf{C}_n\}$) which are *recursively* traversed and checked for possible clashes.

The deep model merging test is also applicable to every subsumption test. A concept C does not subsume a concept D if the pmodels of $\neg\mathsf{C}$ and D can be merged. The application of this technique is especially justified for computing the subsumption hierarchy of TBoxes. For instance, in order to insert a concept C into the subsumption lattice of the (atomic) concepts $\{\mathsf{D}_1, \ldots, \mathsf{D}_n\}$ a sequence of concept satisfiability tests for $\neg\mathsf{C} \sqcap \mathsf{D}_i$ and $\mathsf{C} \sqcap \neg\mathsf{D}_i$ $(i \in 1..n)$ might be required. However, these concept conjunctions are proven to be satisfiable in most cases, i.e. the satisfiability test is replaced by a model merging test. In case of the application TBoxes from Section 10.2.6 less than 5% of all subsumption tests are successful, i.e. discover a subsumption.

Since pseudo models for concepts are computed on demand, it might be possible that in order to employ a merging test, it is necessary to first compute the model of a certain concept which is given as one of the input concepts to the model merging test. However, since it is a good heuristic to assume that cached concept models will be heavily reused when

computing the subsumption hierarchy, exploiting the model merging optimization strategy is very effective compared to using an "expensive" rule-based tableaux computation. This is indicated by empirical tests presented in [Horrocks, 1997] and in this chapter.

Definition 10.9 (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. The consistency of $\mathcal{A} = \{\mathsf{a}:\mathsf{C}\}$ is tested. If $\mathcal{A}$ is inconsistent, the *pseudo model*[8] of C is defined as $\bot$. If $\mathcal{A}$ is consistent, then there exists a set of completions $\mathcal{C}$. A completion $\mathcal{A}' \in \mathcal{C}$ is selected and a pmodel 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.

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

The set M^{A} ($M^{\neg\mathsf{A}}$) contains all (negated) atomic concept names occurring in the assertions for an individual a in a completion $\mathcal{A}'$. Analogously, the set $M^{\exists}$ ($M^{\forall}$) contains all exists- and at-least concepts (at-most-, all-concepts, and exists-concepts for features). This guarantees the correct treatment of features.

The procedure **mergable** shown in Procedure 8 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 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 $\mathsf{R}' \in RS$.

[8] For brevity a pseudo model is called a *pmodel*.

Procedure 8 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
```

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 $\{\text{get\_pmodel}(\mathsf{C}) \mid \mathsf{C} \in Q\}$. 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. Note that mergable depends on the clash triggers (see Definition 9.15) of the particular tableaux calculus chosen since it has to detect potential clashes in a set of pmodels. The structure and composition of the completion rules might vary as long as the clash triggers do not change and the calculus remains sound and complete.

Proposition 10.3 (Soundness of mergable) Let $D?$ have either the value *true* or *false*, $CS = \{\mathsf{C}_1, \ldots, \mathsf{C}_n\}$, $M_{C_i} = \text{get\_pmodel}(\mathsf{C}_i)$, 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 satisfiable.

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 8. 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 8) 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 $CS' = \{\mathsf{C_{i_1}}, \ldots, \mathsf{C_{i_{k_2}}}\} \subseteq CS$ such that $\exists_{\leq m}\mathsf{R} \in \cup_{C_i \in CS'} M_{C_i}^{\forall}$ and $\sum_{\mathsf{E'} \in N} \mathrm{num}_{RS}(\mathsf{E'}) \geq l$, with $N = \cup_{C_i \in CS'} M_{C_i}^{\exists}$ and $RS = (\cup_{\mathsf{j} \in \mathsf{1..k_1}} \mathsf{S_j}^{\uparrow}) \cap \mathsf{R}^{\downarrow}$. However, the call of mergable$(PM, \emptyset, D?)$ returned *true* due to our assumption. 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 8) 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 graph we know that the node resembling $\mathsf{a_{n-1}}$ of this call graph chain contains the call mergable$(PM', VM', \mathit{true})$ 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*.

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

(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*.

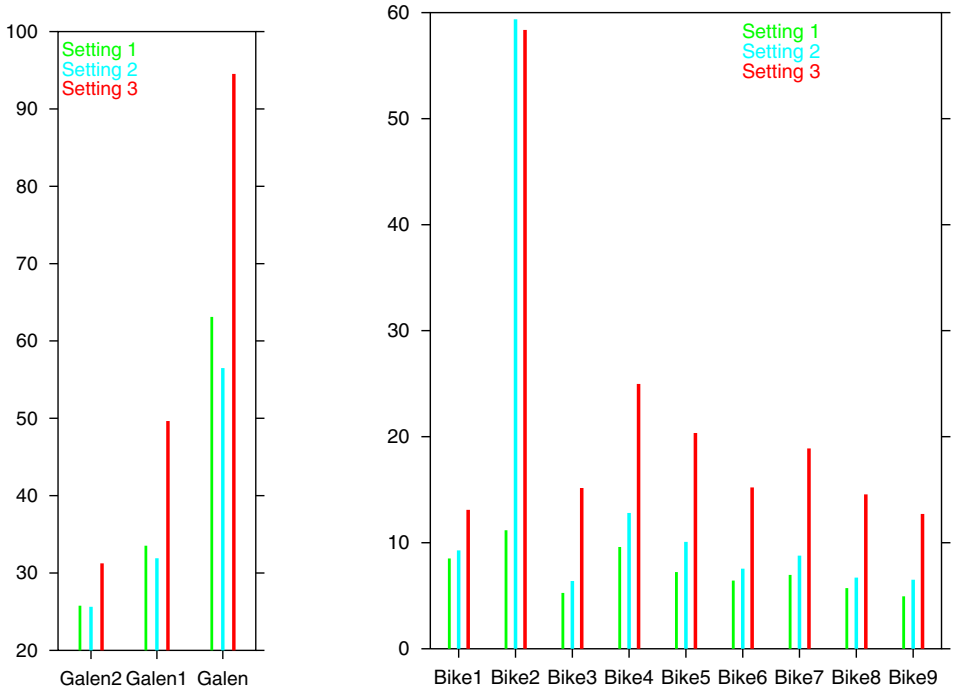
It is easy to see that this proof also holds in the case the value of $D?$ is *false* since the "flat mode" is more conservative than the "deep" one, i.e. it will always return *false* instead of possibly *true* if the set of collected pmodels M' is not empty (line 12 in Procedure 8) □

As shown above the so-called *flat* model merging technique is a special case of the deep model merging technique. Instead of recursively traversing collected sets of pseudo submodels only the first level is considered. It is important to note that in an optimized implementation it is sufficient to perform a *pairwise* test of pmodels for possible clashes or role interactions.

If the flat model merging test returns *false* because of interacting all- and some-concepts, this test might be too conservative. This is illustrated with a small example ($\mathsf{C}, \mathsf{D} \in C$, $\mathsf{R}, \mathsf{S} \in R$). For instance, the deep model merging test starts with the pseudo models $\langle \emptyset, \emptyset, \{\exists\,\mathsf{R}\,.\,\exists\,\mathsf{S}\,.\,\mathsf{C}\}, \emptyset\rangle$ and $\langle \emptyset, \emptyset, \emptyset, \{\forall\,\mathsf{R}\,.\,\forall\,\mathsf{S}\,.\,\mathsf{D}\}\rangle$. Due to the interaction on the role R, the test is recursively applied to the pmodels $\langle \emptyset, \emptyset, \{\exists\,\mathsf{S}\,.\,\mathsf{C}\}, \emptyset\rangle$ and $\langle \emptyset, \emptyset, \emptyset, \{\forall\,\mathsf{S}\,.\,\mathsf{D}\}\rangle$. Eventually, the deep model merging test succeeds with the pmodels $\langle \{\mathsf{C}\}, \emptyset, \emptyset, \emptyset\rangle$ and $\langle \{\mathsf{D}\}, \emptyset, \emptyset, \emptyset\rangle$ and returns true.

The advantage of the deep vs. the flat mode of the model merging technique is demonstrated by empirical tests using the 'Galen' and 'Bike' TBoxes. Figure 10.9 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, the second one the subtableaux caching technique [Horrocks and Patel-Schneider, 1999; Haarslev and Möller, 1999b] 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 in RACE if a "subtableaux" is tested for consistency. 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-up 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. However, the BCS3 TBox introduced in Section 10.2.6 can be classified within less than 10 seconds of runtime if subtableaux caching is enabled, but cannot be classified within 10000 seconds



(a) Galen TBoxes (b) Bike TBoxes

Figure 10.9: Evaluation of pseudo model merging techniques (3 runs for each TBox; Setting 1: all optimizations enabled, Setting 2: subtableaux caching disabled, Setting 3: both subtableaux caching and the deep mode of model merging disabled).

of runtime if subtableaux caching is disabled and the flat or deep mode of model merging is enabled.

10.7 Exploiting Flat Pseudo 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.

Definition 10.10 (Individual Pseudo Model) 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.

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

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.

Proposition 10.4 (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($PM, \emptyset, \mathit{false}$) 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 mergable($\{M_a, M_{\neg C}\}, \emptyset, \mathit{false}$) is called and 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 additionally 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. 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 $(\mathsf{a},\mathsf{b}_1){:}\mathsf{R}_1, \ldots, (\mathsf{b}_n,\mathsf{b}){:}\mathsf{R}_{n+1}, n \geq 0$ 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 8).

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 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 8) since $\mathsf{D} \in (M_a^{\mathsf{A}} \cap M_{\neg C}^{\neg\mathsf{A}}) \cup (M_a^{\neg\mathsf{A}} \cap M_{\neg C}^{\mathsf{A}})$.

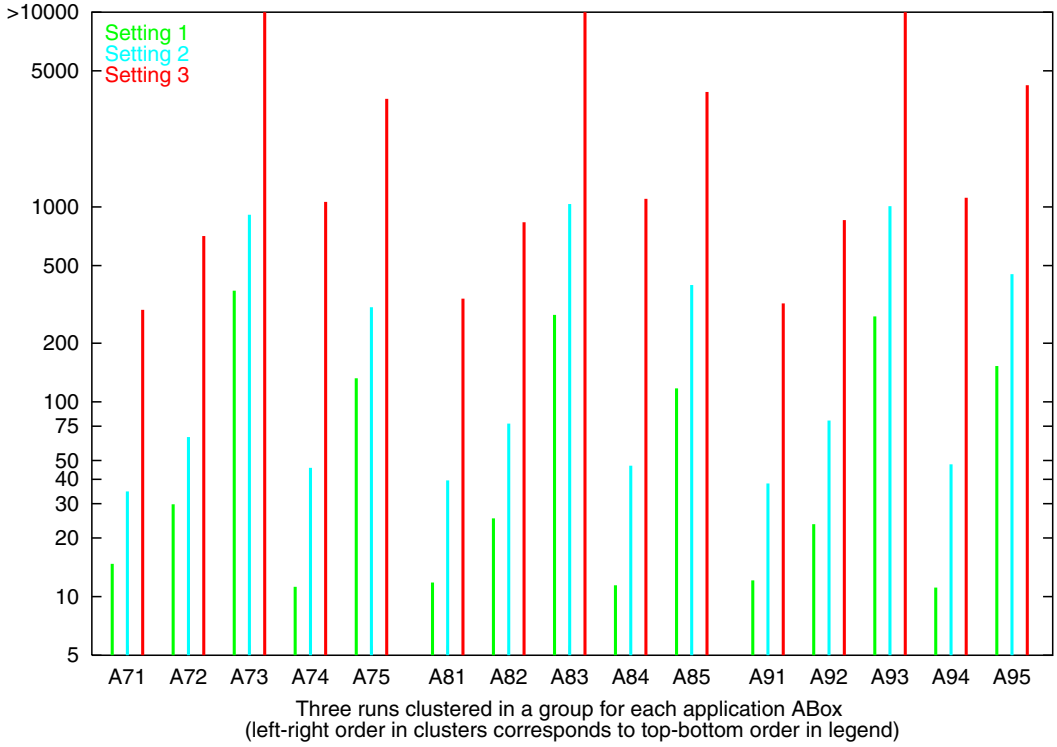


Figure 10.10: Evaluation of the individual model merging technique (3 runs for each TBox; Setting 1: all optimizations enabled, Setting 2: 'told disjoints' disabled, Setting 3: both 'told disjoints' and individual model merging disabled).

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 $\exists_{\geq n_j}\,\mathsf{S_j}$,[10] $\mathsf{S_j} \in \mathsf{R}^{\downarrow}$ with $\mathsf{j} \in \mathsf{1..k}$, such that $\sum_{\mathsf{E}' \in N} \mathrm{num}_{RS}(\mathsf{E}') \geq l$ with $RS = (\cup_{\mathsf{j} \in \mathsf{1..k}}\,\mathsf{S_j}^{\uparrow}) \cap \mathsf{R}^{\downarrow}$. This contradicts the assumption that the call of mergable$(\{M_a, M_{\neg C}\}, \emptyset, \mathit{false})$ returned *true* since mergable called critical_at_most (lines 6-8 in Procedure 8) which returned *true* since the condition $\sum_{\mathsf{E}' \in N} \mathrm{num}_{RS}(\mathsf{E}') \geq l > m$ was satisfied. □

The performance gain by the individual model merging technique is 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 w.r.t. the number of disjointness declarations for atomic concepts. Figure 10.10 shows the runtimes for the realization of the ABoxes 1-5. Each ABox is realized with three different settings. The first setting has all optimization techniques enabled, in the second one an optimization technique[11] is disabled that exploits the dis-

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

[11] 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.

AKB Name	Gen. Lev.	Solv. Lev.	No. of Inds	Time Opt.	Solv. Lev.	No. of Inds	Time No Opt.
k_branch_n	4	3	126	43.25s	3	126	77.25s
k_d4_n	4	2	128	35.77s	2	128	40.57s
k_dum_n	21	13	145	391.19s	13	145	395.68s
k_grz_n	19	16	292	433.68s	15	264	414.41s
k_lin_n	11	11	2	151.70s	6	19	339.34s
k_path_n	4	3	94	350.95s	2	36	111.03s
k_ph_n	7	5	9	3.52s*	5	9	37.70s*
k_poly_n	8	6	73	470.62s	5	65	384.34s
k_t4p_n	6	2	117	284.43s	2	117	319.13s

*TBox of next level timed out after 500 secs.

Figure 10.11: Comparison for synthetic ABox benchmarks.

jointness between concepts, and the third setting has additionally disabled the individual model merging technique. 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.

We also evaluated the individual model merging technique with a set of synthetic ABox benchmarks [Haarslev and Möller, 1999b] (timeout of 500 secs). The results (see Figure 10.11) indicate a speed gain of one order of magnitude for two third of the problems and only a minor speed-up for the other third. However, we like to remark that these ABox benchmarks are rather artificial since they were generated from models of concepts. The ABoxes contain no role joins or cycles and the individuals are members of many concepts which is rather unusual for realistic ABoxes.

An enhanced version of the individual model merging technique can be developed, which additionally exploits the use of deep models. This is immediately possible if only ABoxes containing no joins for role assertions are encountered. In case an ABox $\mathcal{A}$ contains a join (e.g. $\{(\mathsf{a},\mathsf{c}):\mathsf{R},(\mathsf{b},\mathsf{c}):\mathsf{R}\} \subseteq \mathcal{A}$), one has to consider a graph-like instead of a tree-like traversal of pseudo models reflecting the dependencies caused by joins. By analogy to the comparison between flat and deep model merging for classification we conjecture a moderate speed gain by using the proposed enhanced individual model merging technique.

It is explained in Section 11.4.

Chapter 11

High Performance Reasoning with Very Large TBoxes

In application projects it is often necessary to deal with TBoxes containing thousands of simple axioms. These TBoxes are usually automatically derived from databases or dictionaries. For instance, in the medical domain already exist very large databases describing anatomical knowledge. 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 (or primitive concept definitions). The question arises whether the optimization techniques introduced in the previous chapter scale up and allow one to cope with these TBoxes. Furthermore, it has been argued that only systems based on incomplete calculi can deal with knowledge bases containing more than 100,000 axioms of this kind.

It is shown in this chapter that these techniques still work quite well. However, new optimization techniques are required which address the characteristics of these TBoxes. It is also shown that DL 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 10.2.5) and if there exist (almost) no defined concepts. The effectiveness of the new techniques is demonstrated by an empirical analysis of the performance of RACE applied to TBoxes of this size.

As an example we consider a reconstruction of important parts of the UMLS (Unified Medical Language System) [McCray and Nelson, 1995] using description logic representation techniques. The reconstruction is described in [Schulz and 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 metathesaurus. 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 and Hahn, 2000] for details):

ana_heart $\sqsubseteq$ ana_heart_s $\sqcap$ ana_hollow_viscus $\sqcap$ umls_body_part_organ_or_organ_component

ana_heart_s $\sqsubseteq$ ana_hollow_viscus_s $\sqcap$ ana_cardiovascular_system_p

ana_heart_p $\sqsubseteq$ $\neg$ana_heart $\sqcap$ ana_heart_s $\sqcap$ $\exists_{\geq 1}$ anatomical_part_of_ana_heart

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

anatomical_part_of_ana_heart $\sqsubseteq$ 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.

Modern DL systems such as RACE offer (at least) two operations for TBoxes: classification and coherence checking. Classification is the process of computing the most-specific subsumption relationships ("parents" and "children") of every concept name to other concept names mentioned in a TBox. Coherence checking determines that no concept name is unsatisfiable. The following subsections introduce three techniques to speed up the classification or coherence test of very large TBoxes.

11.1 Topological Sorting for Achieving Quasi Definition Order

For TBox classification the RACE system employs the marking and propagation techniques introduced in [Baader et al., 1994a]. The parents and children of a certain concept name are computed in so-called 'top search' and 'bottom search' traversal phases, respectively. For large knowledge bases it is particularly important to avoid as many traversals as possible. Let us assume, a TBox can be transformed such that all GCIs can be absorbed but cyclic (primitive) concept definitions may still 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., 1994a]. According to [Baader et al., 1994a] 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. The relation 'uses' is the transitive closure of 'directly uses'. If A uses B then B comes before A 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. 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 uses non-modal' similar to 'directly uses' but with references occurring in the scope of quantifiers not considered. Again 'uses non-modal' is the transitive closure of 'directly uses non-modal'. For acyclic concepts the 'uses non-modal' 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".

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 primitive concept that is not a member of a set S_i, we claim that the bottom search can be disabled. The 'uses non-modal' relation and the quasi definition order serialization ensures that either all concepts which 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.

Topological sorting is of order $n + e$ where e is the number of given 'uses non-modal' 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., 1994a] no experiments are discussed that involve the computation of a serialization given a TBox with axioms not already in (strict) definition order.

11.2 TBox Clustering

A problem with large TBoxes is that the set of children of some concept names can get very large (e.g. some concepts in the UMLS TBoxes have more than 20000 children). Thus, the top search and bottom search procedures each exhibit worst case performance, i.e. the optimization techniques presented in [Baader et al., 1994a] and in the previous chapter are not effective enough in this case. Therefore, a clustering technique had to be developed in order to keep the number of traversals in the concept hierarchy and the number of needed subsumption tests as small as possible. The technique works as follows.

If a certain concept name gets more than θ children assigned, these 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 inserted into the subsumption lattice with $\mathsf{A}_1 \ldots \mathsf{A}_\theta$ being the children of $\mathsf{A}_{\mathsf{new}}$. Note that bucket concepts are virtual concepts in the sense that they are not referred to 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 $\mathsf{A_i}$ ($\mathsf{i} \in \{1..\theta\}$) subsumes A during the top search phase, our findings suggest that it is more effective to initially test whether $\mathsf{A_{new}}$ does not subsume A using the model merging technique. Since in most cases, no subsumption relation can be found between any $\mathsf{A_i}$ and A, one test possibly replaces θ tests. On the other hand, if a subsumption relation indeed exists, then clustering introduces some overhead. However, in the case of the UMLS TBoxes only primitive concept definitions are included in the TBox for almost all concept names. Thus, the pseudo model of $\neg\mathsf{A_{new}}$ being used for model merging is very simple because the pseudo model basically consists only of a set of negated concept names (see Section 10.6 and Definition 10.9).

For best performance, the number of concepts to be kept in a bucket should depend on the number of children of the concept. However, this 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 has to be created for a certain concept A and there are already σ buckets clustering the children of A, then two buckets (those buckets with the smallest number of children) are merged. Merging the buckets $\mathsf{A_{new}} \doteq \mathsf{A_1} \sqcup \ldots \sqcup \mathsf{A_n}$ and $\mathsf{B_{new}} \doteq \mathsf{B_1} \sqcup \ldots \sqcup \mathsf{B_m}$ means that the bucket $\mathsf{A_{new}}$ is "redefined" as $\mathsf{A_{new}} \doteq \mathsf{A_1} \sqcup \ldots \sqcup \mathsf{A_n} \sqcup \mathsf{B_1} \sqcup \ldots \sqcup \mathsf{B_m}$ and the bucket $\mathsf{B_{new}}$ is reused for the new bucket to be created (see above).[1] Whether hierarchical clustering techniques lead to performance improvements is subject to further research.

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

11.3 Dealing with Domain and Range Restrictions

In order to avoid disjunctions, GCIs for domain restrictions are dealt with by RACE 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 an assertion $\mathsf{i}\!:\!\exists\,\mathsf{R}\,.\,\mathsf{C}$ for an arbitrary $\mathcal{ALCNH}_{R^+}$ concept term is found in an ABox.[2] If lazy unfolding is applied, domain restrictions have to be considered w.r.t. the 'directly uses non-modal' relation in a special way.

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 of the

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

[2] 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 cannot guarantee the uniqueness of the encoding result (which is essential for clash detection).

form $\neg\mathsf{ana\_heart\_p} \sqsubseteq \forall\,\mathsf{anatomical\_part\_of\_ana\_heart}\,.\,\bot$, lazy unfolding cannot be easily applied if an inclusion axiom for $\mathsf{ana\_heart\_p} \sqsubseteq \ldots$ also exists. Indeed, this is the case for UMLS. Hence, in order to apply the topological sorting optimization, the incorporation of domain restrictions into the tableaux calculus was necessary because all GCIs need to be absorbed for topological sorting to be a valid optimization.

Note that, in principle, RACE also supports the absorption of GCIs of the form $\neg\mathsf{A} \sqsubseteq \mathsf{C}_1$ (but only if no inclusion axiom $\mathsf{A} \sqsubseteq \mathsf{C}_2$ or definition $\mathsf{A} \doteq \mathsf{C}_2$ also exists). Some knowledge bases can only be handled effectively when the absorption of GCIs of the form $\neg\mathsf{A} \sqsubseteq \mathsf{C}_1$ is supported.

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 only "considered" if an existential restriction for R or a subrole of R is imposed for a certain individual i. These cases can also be easily detected.

11.4 Exploiting Disjointness Declarations

As has been discussed in [Baader et al., 1994a], 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 above, $\mathsf{ana\_heart}$ is computed as a told disjoint concept of $\mathsf{ana\_heart\_p}$ 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., 1994a] 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., 1994a].

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 $\mathsf{i}:\mathsf{A}$ 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 Section 10.7 and Figure 10.10 for experimental results).

11.5 Caching Policies

RACE supports different subtableaux caching policies (see also Section 10.2.3 for a discussion of subtableaux caching). Two types of caches are provided which can be used together

or alternatively. Both cache types are accessed via keys constructed from a set of concepts representing a subtableau. The first cache (called equal cache) contains entries about the satisfiability status of a subtableaux already encountered. This cache only returns a hit if the search key exactly matches (i.e. is equal to) the key of a known entry.

The second cache type consists of a pair of caches. One cache contains only entries for satisfiable subtableaux while the other one stores unsatisfiable subtableaux. These caches support queries concerning already encountered supersets and subsets of a given search key. This technique was inspired by [Hoffmann and Köhler, 1999]. For the UMLS benchmarks the (additional) equal cache had to be disabled in order to reduce space requirements.[3]

11.6 Empirical Results for TBox Classifications

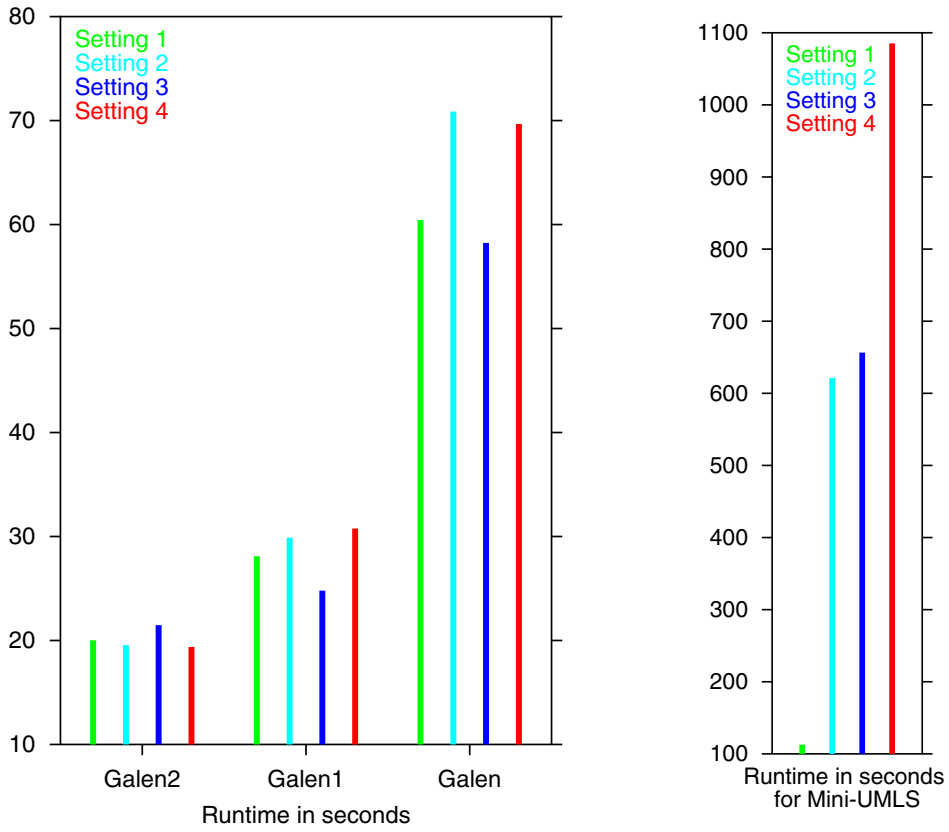
The performance of the RACE system is evaluated with different versions of the UMLS knowledge base and the 'Galen' TBoxes. UMLS-1 is a preliminary version that contains many inconsistent concept names. UMLS-1 consists of approximately 100,000 concept names and for almost all of them there exists a primitive 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 we used for our empirical tests uses approximately 160,000 concept names and 80,000 roles.

Originally, the UMLS knowledge base has been developed with Loom 4.0 [MacGregor, 1994]. If Loom is given 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, in [Schulz and Hahn, 2000] the cycle-causing concepts are placed in a so-called `:implies` clause, i.e. these restrictions are only asserted to 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 and to test more than one TBox with RACE, for each of the knowledge base versions, UMLS-1 and UMLS-2, three different subversions are 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.

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

[3] If the equal cache is enabled, it is the first reference. Only if an equal cache lookup fails, the superset or the subset caches are consulted. All retrieval results from the superset or subset caches are also entered into the equal cache.



(a) Galen TBoxes (b) Mini-UMLS TBox

Figure 11.1: Evaluation of the topological sorting and clustering techniques for selected TBoxes (4 runs for each TBox; Setting 1: all optimizations enabled, Setting 2: clustering disabled, Setting 3: topological sorting disabled, Setting 4: both topological sorting and clustering disabled).

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

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

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

Figure 11.1a shows the evaluation result for the 'Galen' TBoxes. There is a minor variation in the runtimes. The settings 2 and 4 are usually slower than setting 1. The improvement for the Mini-UMLS TBox (see Figure 11.1b) which is a small fragment of the UMLS-2c TBox (see below) is dramatic. Setting 4 is one order of magnitude slower than settings 1. However, there is a smaller performance gain for the full UMLS TBoxes.

For the detailed UMLS performance evaluation six different TBoxes have been tested. Without clustering and topological sorting, classifying UMLS-1a can be done in approxi-

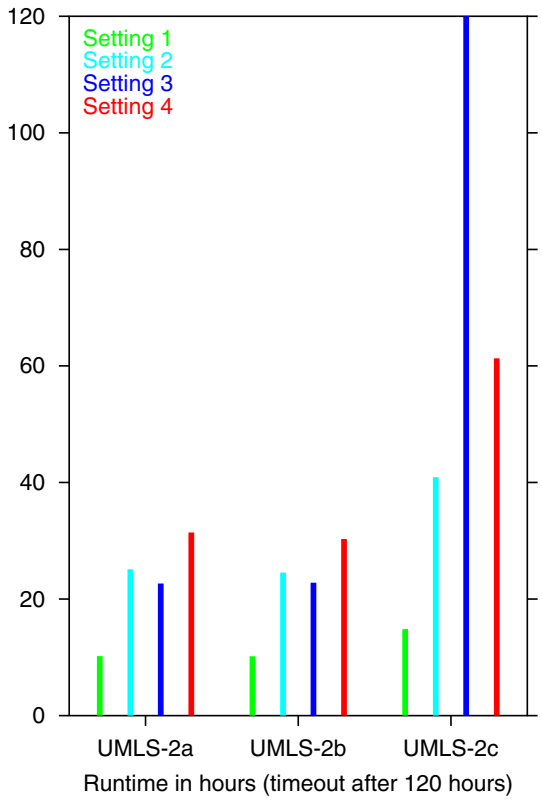


Figure 11.2: Evaluation of the topological sorting and clustering techniques for UMLS2 (4 runs for each TBox; Setting 1: all optimizations enabled, Setting 2: clustering disabled, Setting 3: topological sorting disabled, Setting 4: both topological sorting and clustering disabled).

mately 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 UMLS-1b version requires 3.6 hours (with optimization) and 6.1 hours (without optimization). The reason for the enhanced performance with more constraints is that in this version already 47855 concepts are inconsistent. With domain and range restrictions added even 60246 concepts were classified as inconsistent. 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 (see above) 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 becomes even more difficult 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 concepts "disappear" as synonyms of the bottom concept. For UMLS-2, checking TBox coherence (see above) requires between 15 and 50 minutes (2a: 16 mins, 2b: 19 mins, 2c: 51 mins).

UMLS	**Sorting**	**Clustering**	**Runtime**	**NST** ($\times 10^6$)	**MaxNC**
2a	on	on	10:13	232	26,874
	on	off	25:06	2,341	”
	off	on	22:40	1,256	”
	off	off	31:26	2,796	”
2b	on	on	10:11	232	26,874
	on	off	24:33	2,341	”
	off	on	22:48	1,256	”
	off	off	30:18	2,796	”
2c	on	on	14:53	222	21,298
	on	off	40:54	3,723	”
	off	on	>120:00	?	”
	off	off	61:18	5,814	”

Figure 11.3: Evaluation of the classifications of the UMLS-2 knowledge bases (runtime is given in hours : minutes, NST = number of subsumption tests, MaxNC = maximal number of children).

Performance evaluations of the TBox classifications for UMLS-2 are presented in Figure 11.2 and 11.3. In order to provide a machine-independent evaluation, not only the runtimes are given but also the number of subsumption tests. It should be noted that the tableaux algorithm is needed only for computing pseudo models (see Section 10.6). In other words, all subsumption tests are decided 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 runtimes for setting 2 increase at least by a factor of two, the number of subsumption tests even by one order of magnitude. The result for setting 3 (topological sorting disabled) and UMLS-2a supports the fact that topological sorting is also very effective or even essential. The runtimes also increase at least by a factor of two, the number of subsumption test by a factor of 5. The runtime result for setting 3 and UMLS-3c is caused by removed buckets. A bucket has to be removed if a member of this bucket gets a new parent assigned. This situation[4] is very likely if topological sorting is disabled since named concepts will be selected in the “wrong” order for sorting them into the subsumption hierarchy. This evaluation for setting 3 and UMLS-3c timed out after five days of runtime. At this time only 80,000 of the 160,000 concepts had been sorted into the subsumption hierarchy, i.e. the induced overhead due to “bucket thrashing” is dramatic. One can expect an estimated runtime of at least ∼300 hours for this setting.

If, as in setting 4, both clustering and topological sorting are disabled, the runtimes increase only to a limited extent compared to the settings 2-3. Moreover, according to the evaluation

[4]For instance, in setting 3 (topological sorting disabled) and UMLS-2b more than 60,000 buckets had to be removed.

results, the UMLS-2b version does not require more computational resources than UMLS-2a (see the discussion about `:implies` from above). Only the incorporation of domain and range restrictions cause runtimes to increase. For UMLS-2 up to 800 MBytes of main memory are required. For other benchmark TBoxes (e.g. Galen with approx. 3000 concepts) the results (see Figure 11.1a) demonstrate that there is no significant overhead imposed by the clustering and a slight performance gain might be observed.

In summary, the results for the UMLS TBoxes clearly demonstrate that clustering is only 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.

Part VI

Summary and Outlook

The work reported in this monograph is based on two major lines of research, visual languages and description logics. For both areas we addressed theoretical as well as practical aspects. In the following we summarize the advances achieved in both areas and outline ongoing and future research.

We presented a new logic-based approach to visual language theory. It employs description logic for the specification and recognition of diagrammatical notations. It was practically applied in the generic editor GenEd offering tools for editing and parsing diagrams. The advantages of this approach are its pure declarativeness, high expressiveness, and its meta-reasoning capabilities.

We applied visual languages to spatial query languages in the GIS domain. This is exemplified by the development of a visual spatial query language demonstrating the advantages of this approach. A query is described by a diagram showing an example configuration of query elements. The diagrammatic query language naturally expresses spatial constraints in form of spatial relationships between these elements. The query semantics is supported by using a metaphor from "naive physics." With this language we developed solutions for typical problems with diagrammatical query languages. The explicitness of diagrams is often a disadvantage. As a remedy we integrated into the query language means for expressing relaxations and "don't cares" for spatial constraints. These issues were addressed by the development of VISCO which partially implements this query language. A first proposal for the logical specification of the semantics of parts of VISCO's query language using a decidable logic is presented in this monograph. This approach is based on the idea to translate a diagram (or sketch) into a propositional form represented as an $\mathcal{ALCRP}(\mathcal{D})$ ABox reflecting the spatial constraints given by such a diagram. We offered two solution for query answering. An ABox can be either reduced to a concept term (via an abstraction process) or it can be handled by ABox pattern mechanisms.

Our work on visual languages and examples for the GIS domain led us to investigate the integration of spatial domains into description logics. Due to the lack of proper theories and implementations we first pursued a simplistic approach with GenEd which used the CLASSIC system for TBox classification and ABox realization. The missing reasoning capabilities were emulated by the spatial (geometric) reasoning component of GenEd. The derived spatial knowledge was asserted in the ABox using primitive roles. However, this did not resolve the incompleteness w.r.t. to TBox and ABox reasoning. The practical experience with VISCO and CLASSIC showed that its $\mathcal{ALEN}$-like DL, which offers no full negation, no disjunction, and no qualified existential restriction, was too weak for appropriately describing diagrams or diagrammatic queries. This experience culminated in three research topics for description logics, the integration of spatial reasoning, the development of expressive DLs, and the design and evaluation of optimization techniques.

We advanced the research on integrating spatial reasoning into description logic theory in several ways. As a first step we integrated reasoning about qualitative spatial relations into a $\mathcal{ALEN}$-like DL. This work was extended by developing the DL $\mathcal{ALCRP}(\mathcal{D})$ and

proving the decidability of the ABox consistency problem for a syntactically restricted form of $\mathcal{ALCRP}(\mathcal{D})$. The $\mathcal{ALCRP}(\mathcal{D})$ approach is based on the notion of concrete domains. It provides general mechanisms for integrating reasoning about proper domains. Examples for these domains were discussed in this monograph (e.g. $\mathcal{R}$, $\mathcal{S}_2$) or elsewhere [Haarslev et al., 1999b] (e.g. ALLEN). However, $\mathcal{ALCRP}(\mathcal{D})$ is only decidable if the corresponding syntax restrictions are enforced. This definitely diminishes its applicability and makes modeling much harder.

A first reasoner for $\mathcal{ALCRP}(\mathcal{D})$ was developed as a research prototype. Its evaluation showed the need for extending optimization techniques to description logics with concrete domains (e.g. $\mathcal{ALC}(\mathcal{D})$, $\mathcal{ALCRP}(\mathcal{D})$). A first treatment adapted two major optimization techniques, dependency-directed backtracking and model-merging and caching [Turhan, 2000; Turhan and Haarslev, 2000]. An implementation of these techniques and an empirical evaluation of their effectiveness is left to future work. However, the development of these optimization techniques already motivated the extension of $\mathcal{ALCNH}_{R^+}$ by concrete domains [Haarslev et al., 2000b].

Due to the syntactic restrictions for $\mathcal{ALCRP}(\mathcal{D})$ we decided to pursue alternative approaches. We are developing a DL without concrete domains but offering means for reflecting the semantics of qualitative spatial relations. As a first step the DL $\mathcal{ALCRA}$ was proposed [Wessel et al., 2000]. It is motivated by the way how spatial calculi (e.g. RCC-8 [Randell et al., 1992]) usually describe the composition of relations. A so-called role box in $\mathcal{ALCRA}$ partially reflects the information given by a composition table. Unfortunately, the undecidability of the concept satisfiability problem for $\mathcal{ALCRA}^-$ is shown in [Wessel, 2000]. The logic $\mathcal{ALCRA}^-$ is a variant of $\mathcal{ALCRA}$, where the requirement for the mutual exclusiveness between the roles in a role box is removed and the existence of a universal role is assumed. However, the decidability of $\mathcal{ALCRA}$ is an open problem. Furthermore, in case the decidability of $\mathcal{ALCRA}$ can be proven, the integration of inverse roles into $\mathcal{ALCRA}$, which is necessary to fully resemble a composition table, is still unresolved.

As part of our second research topic, the development of expressive DLs, we showed that the ABox consistency problem for $\mathcal{ALCNH}_{R^+}$ is decidable. Based on the calculus given in this proof we implemented the DL system RACE supporting TBox and ABox reasoning for $\mathcal{ALCNH}_{R^+}$. RACE is also used as a testbed for evaluating optimization techniques applicable to $\mathcal{ALCNH}_{R^+}$. RACE is available over the WWW and at the time of this writing we have recorded hundreds of downloads. RACE is used in a research collaboration with the University of Amsterdam (see [Areces et al., 1999]).

The third topic is concerned with the design and evaluation of optimization techniques for TBox and ABox reasoning. As reported in the preceding chapters, we developed the deep model merging and caching technique, new transformations on GCIs, a classification order defined by topologically sorting named concepts, and a technique for clustering large numbers of children in a TBox taxonomy. These techniques especially speed up the TBox classification phase. We also developed the signature calculus which addresses the inefficiency caused by big numbers occurring in number restrictions. We are currently working on a combination of the signature calculus and the techniques presented in [Ohlbach and

Köhler, 1999]. A special lazy unfolding technique handles domain and range restrictions for roles and avoids a naive treatment with GCIs.

The role path contraction technique transforms ABoxes into simpler ones that are semantically equivalent. The transformation reduces the number of assertions and individuals occurring in an ABox and replaces them with concept assertions. These assertions facilitate the usage of optimization techniques such as model merging, subtableaux caching, etc. The individual model merging technique optimizes special forms of ABox consistency tests usually occurring in the ABox realization phase.

In summary, this monograph has presented research about two areas, visual languages and description logics, which have been considered as disjoint in the past. As part of this study, we showed that both areas could give important impetus to each other. The application of description logics to visual languages resulted in a new logic based theory for visual languages. The practical application of description logic systems proved the need to develop more expressive description logics and corresponding optimization techniques. This experience initiated the development of the description logics $\mathcal{ALCRP}(\mathcal{D})$ and $\mathcal{ALCNH}_{R^+}$. The design, implementation, and empirical evaluation of optimization techniques within the RACE architecture resulted also from this experience. The DL system RACE was presented whose architecture is based on optimization techniques covering almost all aspects of the description logic $\mathcal{ALCNH}_{R^+}$. We do hope that the findings reported in this monograph allow further advances in related research areas.

Bibliography

Abel, D. and Ooi, B., editors (1993). *Advances in Spatial Databases, Third International Symposium, SSD'93, Singapore, June 23-25, 1993*, volume 692 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin.

Aiello, L., Doyle, J., and 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.

Areces, C., Bouma, W., and de Rijke, M. (1999). Description logics and feature interaction. In *Proc. of International Workshop on Description Logics, Linköping, Sweden*, pages 28–32.

Baader, F., Franconi, E., Hollunder, B., Nebel, B., and Profitlich, H. (1994a). 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., Franconi, E., Hollunder, B., Nebel, B., and Profitlich, H. (1994b). An empirical analysis of optimization techniques for terminological representation systems. *Applied Intelligence*, 2(4):109–138.

Baader, F. and Hanschke, P. (1991). 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. and Hollunder, B. (1995a). Embedding defaults into terminological representation systems. *J. Automated Reasoning*, 14:149–180.

Baader, F. and 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., and Franconi, E. (1992). An empirical analysis of optimization techniques for terminological representation systems, or:

Making 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. and Sattler, U., editors (2000). *Proceedings of the International Workshop on Description Logics (DL'2000), August 17 - August 19, 2000, Aachen, Germany.*

Bennett, B. (1995). Modal logics for qualitative spatial reasoning. *Bull. of the IGPL*, 3:1–22.

Borgida, A. (1995). Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671–682.

Borgida, A., Isbell, C., and McGuinness, D. (1996). Reasoning with black boxes: Handling test concepts in classic. In [Padgham et al., 1996], pages 87–91. Technical Report WS-96-05.

Borgida, A. and Patel-Schneider, P. (1994). A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308.

Borgo, S., Guarino, N., and Masolo, C. (1996). A pointless theory of space based on strong connection and congruence. In [Aiello et al., 1996], pages 220–229.

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., McGuinness, D., Patel-Schneider, P., Resnick, L., and Borgida, A. (1991). Living with CLASSIC: When and how to use a KL-ONE-like language. In [Sowa, 1991], pages 401–456.

Brachman, R. and Schmolze, J. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, pages 171–216.

Buchheit, M., Donini, F., and Schaerf, A. (1993). Decidable reasoning in terminological knowledge representation systems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence, IJCAI-93*, pages 704–709, Chambery (France).

Calcinelli, D. and Mainguenaud, M. (1994). Cigales, a visual query language for a geographical information system: the user interface. *Journal of Visual Languages and Computing*, 5(2):113–132.

Catarci, T., Costabile, M., Levialdi, S., and Batini, C. (1997). Visual query systems for databases: A survey. *Journal of Visual Languages and Computing*, 8(2):215–260.

Citrin, W., Doherty, M., and Zorn, B. (1994). Formal semantics of control in a completely visual programming language. In [VL'94, 1994], pages 208–215.

Clarke, B. (1981). A calculus of individuals based on 'connection'. *Notre Dame Journal of Formal Logic*, 22(3):204–218.

Clarke, B. (1985). Individuals and points. *Notre Dame Journal of Formal Logic*, 26(1):204–218.

Clementini, E. and Di Felice, P. (1997). Approximate topological relations. *International Journal of Approximate Reasoning*, 16:173–204.

Clementini, E., Di Felice, P., and van Oosterom, P. (1993). A small set of formal topological relationships suitable for end-user interaction. In [Abel and Ooi, 1993], pages 277–295.

Cohn, A. (1997). Qualitative spatial representation and reasoning techniques. In Brewka, G., Habel, C., and Nebel, B., editors, *Proceedings, KI-97: Advances in Artificial Intelligence, 21st Annual German Conference on Artificial Intelligence, Freiburg, Germany*, volume 1303 of *Lecture Notes in Artificial Intelligence*, pages 1–30. Springer Verlag, Berlin.

Cohn, A., Bennett, B., Gooday, J., and Gotts, N. (1997). Representing and reasoning with qualitative spatial relations. In [Stock, 1997], pages 97–134.

Cohn, A., Giunchiglia, F., and Selman, B., editors (2000). *Proceedings of Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000), Breckenridge, Colorado, USA, April 11-15, 2000.*

Cohn, A. and Gooday, J. (1994). Defining the syntax and the semantics of a visual programming language in a spatial logic. In *AAAI-94, Spatial and Temporal Reasoning Workshop.*

Cohn, T., Schubert, L., and Shapiro, S., editors (1998). *Proceedings of Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy, June 2-5, 1998.*

Costagliola, G., Tomita, M., and Chang, S. (1991). A generalized parser for 2-D languages. In *1991 IEEE Workshop on Visual Languages, Kobe, Japan, Oct. 8-11*, pages 98–104. IEEE Computer Society Press, Los Alamitos.

Crimi, C., Guercio, A., Nota, G., Pacini, G., Tortora, G., and Tucci, M. (1991). Relation grammars and their application to multi-dimensional languages. *Journal of Visual Languages and Computing*, 2(4):333–346.

De Giacomo, G. and Lenzerini, M. (1996). TBox and ABox reasoning in expressive description logics. In [Aiello et al., 1996].

Del Bimbo, A., Vicario, E., and Zingoni, D. (1994). A spatial logic for symbolic description of image contents. *Journal of Visual Languages and Computing*, 5(3):267–286.

Egenhofer, M. (1991). Reasoning about binary topological relations. In Günther, O. and Schek, H.-J., editors, *Advances in Spatial Databases, Second Symposium, SSD'91, Zurich, Aug. 28-30, 1991*, volume 525 of *Lecture Notes in Computer Science*, pages 143–160. Springer Verlag, Berlin.

Egenhofer, M. (1992). Why not SQL! *International Journal on Geographical Information Systems*, 6(2):71–85.

Egenhofer, M. (1996). Spatial-query-by-sketch. In [VL'96, 1996], pages 60–67.

Egenhofer, M. (1997). Query processing in spatial-query-by-sketch. *Journal of Visual Languages and Computing*, 8(4):403–424.

Franconi et al., E., editor (1998). *Proceedings of the International Workshop on Description Logics (DL'98), June 6-8, 1998, Trento, Italy.*

Freeman, J. (1995). *Improvements to propositional satisfiability search algorithms.* PhD thesis, University of Pennsylvania, Computer and Information Science.

Glasgow, J., Narayanan, N., and Chandrasekaran, B., editors (1995). *Diagrammatic Reasoning: Cognitive and Computational Perspectives.* AAAI Press / The MIT Press, Menlo Park.

Golin, E. (1991). Parsing visual languages with picture layout grammars. *Journal of Visual Languages and Computing*, 2(4):371–393.

Gooday, J. and Cohn, A. (1996). Using spatial logic to describe visual programming languages. *Artificial Intelligence Review*, 10:171–186.

Göttler, H. (1989). Graph grammars, a new paradigm for implementing visual languages. In *Rewriting Techniques and Applications, 3rd International Conference, RTA-89, 3-5 April 1989, Chapel Hill, NC*, pages 152–166. Springer Verlag, Berlin.

Graf, M. (1990). Visual programming and visual languages: Lessons learned in the trenches. In *Visual Programming Environments: Applications and Issues*, pages 452–454, Los Alamitos, California. IEEE Computer Society Press.

Grigni, M., Papadias, D., and Papadimitriou, C. (1995). Topological inference. In Mellish, C., editor, *Fourteenth International Joint Conference on Artificial Intelligence, Montreal, Quebec, Canada, Aug. 20-25, 1995*, pages 901–906.

Haarslev, V. (1995). Formal semantics of visual languages using spatial reasoning. In [VL'95, 1995], pages 156–163.

Haarslev, V. (1996a). A fully formalized theory for describing visual notations (extended abstract). In *Proceedings of the International Workshop on Theory of Visual Languages, held in conjunction with AVI'96, May 30, 1996, Gubbio, Italy.* 9 pages.

Haarslev, V. (1996b). Using description logic for reasoning about diagrammatical notations. In [Padgham et al., 1996], pages 124–128. Technical Report WS-96-05.

Haarslev, V. (1998a). A fully formalized theory for describing visual notations. In [Marriott and Meyer, 1998b], pages 261–292.

Haarslev, V. (1998b). A logic-based formalism for reasoning about visual representations (extended abstract). In *Proceedings, AAAI Workshop on Formalizing Reasoning with Visual and Diagrammatic Representations, AAAI Fall Symposium Series 1998, October 23-25, Orlando, Florida/USA. Technical Report FS-98-04*, pages 57–66. AAAI Press.

Haarslev, V. (1999). A logic-based formalism for reasoning about visual representations. *Journal of Visual Languages and Computing*, 10(4):421–445.

Haarslev, V., Horrocks, I., Möller, R., and 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., and Möller, R. (1998a). Foundations of spatioterminological reasoning with description logics. In [Cohn et al., 1998], pages 112–123.

Haarslev, V., Lutz, C., and 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. and 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. and Möller, R. (1997b). Spatioterminological reasoning: Subsumption based on geometrical inferences. In [Rousset et al., 1997], pages 74–78.

Haarslev, V. and 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. and 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. and 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. and Möller, R. (1999d). RACE system description. In [Lambrix et al., 1999], pages 130–132.

Haarslev, V. and Möller, R. (1999e). RACE System Download Page. Available at URL http://kogs-www.informatik.uni-hamburg.de/~race/.

Haarslev, V. and Möller, R. (2000a). Consistency testing: The RACE experience. In Dyckhoff, R., editor, *Proceedings, Automated Reasoning with Analytic Tableaux and Related Methods, University of St Andrews, Scotland, 4-7 July, 2000*, pages 57–61. Springer Verlag, Berlin.

Haarslev, V. and Möller, R. (2000b). Expressive ABox reasoning with number restrictions, role hierarchies, and transitively closed roles. In [Cohn et al., 2000], pages 273–284.

Haarslev, V. and Möller, R. (2000c). High performance reasoning with very large knowledge bases. In [Baader and Sattler, 2000], pages 143–152.

Haarslev, V. and Möller, R. (2000d). Optimizing TBox and ABox reasoning with pseudo models. In [Baader and Sattler, 2000], pages 153–162.

Haarslev, V., Möller, R., and Schröder, C. (1994). Combining spatial and terminological reasoning. In Nebel, B. and 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., and Tobies, S. (2000a). Signature calculus: Optimizing reasoning with number restrictions. In preparation.

Haarslev, V., Möller, R., and Turhan, A.-Y. (1998b). HAM-ALC. In [Franconi et al., 1998], pages 64–65. Benchmark results for DL'98 comparison.

Haarslev, V., Möller, R., and Turhan, A.-Y. (1998c). Implementing an $\mathcal{ALCRP}(\mathcal{D})$ ABox reasoner: Progress report. In [Franconi et al., 1998], pages 82–86.

Haarslev, V., Möller, R., and 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., and Wessel, M. (1999d). On specifying semantics of visual spatial query languages. In [VL'99, 1999], pages 4–11.

Haarslev, V., Möller, R., and Wessel, M. (2000b). The description logic $\mathcal{ALCNH}_{R^+}$ extended with concrete domains: Revised version. Technical Report FBI-HH-M-290/00, University of Hamburg, Computer Science Department.

Haarslev, V., Möller, R., and Wessel, M. (2000c). Visual spatial query languages: A semantics using description logics. In [Olivier et al., 2000]. In print.

Haarslev, V. and Wessel, M. (1996). GenEd—an editor with generic semantics for formal reasoning about visual notations. In [VL'96, 1996], pages 204–211.

Haarslev, V. and Wessel, M. (1997). Querying GIS with animated spatial sketches. In *1997 IEEE Symposium on Visual Languages, Capri, Italy, Sep. 23-26*, pages 197–204. IEEE Computer Society Press, Los Alamitos.

Hanschke, P. (1996). *A Declarative Integration of Terminological, Constraint-based, Data-driven, and Goal-directed Reasoning*. Infix, Sankt Augustin.

Helm, R. and Marriott, K. (1991). A declarative specification and semantics for visual languages. *Journal of Visual Languages and Computing*, 2(4):311–331.

Hoffmann, J. and 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. and Baader, F. (1991). Qualifying number restrictions in concept languages. In Allen, J., Fikes, R., and 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.

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. and Patel-Schneider, P. (1999). Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293.

Horrocks, I. and 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., and Tobies, S. (1999). Practical reasoning for expressive description logics. In Ganzinger, H., McAllester, D., and 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., and Tobies, S. (2000a). Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–264.

Horrocks, I., Sattler, U., and Tobies, S. (2000b). Reasoning with individuals for the description logic $\mathcal{SHIQ}$. 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. and Tobies, S. (2000). Reasoning with axioms: Theory and practice. In [Cohn et al., 2000], pages 285–296.

Kahn, K. and Saraswat, V. (1990). Complete visualizations of concurrent programs and their executions. In *1990 IEEE Workshop on Visual Languages, Skokie, Illinois, Oct. 4-6*, pages 7–14. IEEE Computer Society Press, Los Alamitos.

Kahn, K., Saraswat, V., and Haarslev, V. (1991). Pictorial Janus: A Completely Visual Programming Language and its Environment (in German). In Encarnacao, J., editor, *GI-Fachgespräch Programmieren multimedialer Anwendungen der GI-Jahrestagung 1991, Darmstadt, Oktober 1991*, pages 427–436. Springer Verlag, Berlin.

Lambrix et al., P., editor (1999). *Proceedings of the International Workshop on Description Logics (DL'99), July 30 - August 1, 1999, Linköping, Sweden.*

Lange, H. and Schröder, C. (1994). Analysis and interpretation of changes in aerial images: Knowledge interpretation and spatial reasoning. In Ebner, H., Heipke, C., and Eder, K., editors, *ISPRS Commision III Symposium – Spatial Information from Digital Photogrammetry and Computer Vision, Munich, Germany, Sep. 5–9, 1994*, volume 30 of *International Archives of Photogrammetry and Remote sensing*, pages 475–482.

Lee, Y. and Chin, F. (1995). An iconic query language for topological relationships in GIS. *International Journal on Geographical Information Systems*, 9(1):25–46.

Lemon, O. (1996). Semantical foundations of spatial logics. In [Aiello et al., 1996], pages 212–219.

Lewis, H. and Papadimitriou, C. (1981). *Elements of the Theory of Computation.* Prentice-Hall, Englewood Cliffs, New Jersey.

Lutz, C., Haarslev, V., and 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. and Möller, R. (1997). Defined topological relations in description logics. In [Rousset et al., 1997], pages 15–19.

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.

Marriott, K. (1994). Constraint multiset grammars. In [VL'94, 1994], pages 118–125.

Marriott, K. and Meyer, B. (1997). On the classification of visual languages by grammar hierarchies. *Journal of Visual Languages and Computing*, 8(4):375–402.

Marriott, K. and Meyer, B. (1998a). The CCMG visual language hierarchy. In [Marriott and Meyer, 1998b], pages 129–169.

Marriott, K. and Meyer, B., editors (1998b). *Visual Language Theory*. Springer Verlag, Berlin.

Marriott, K., Meyer, B., and Wittenberg, K. (1998). A survey of visual language specification and recognition. In [Marriott and Meyer, 1998b], pages 5–85.

McCray, A, T. and Nelson, S. (1995). The representation of meaning in the UMLS. *Methods of Information in Medicine*, 34(1/2):193–201.

Meyer, B. (1992). Pictures depicting pictures: On the specification of visual languages by visual grammars. In *1992 IEEE Workshop on Visual Languages, Seattle, Washington, Sept. 15-18*, pages 41–47. IEEE Computer Society Press, Los Alamitos.

Meyer, B. (1994). Pictorial deduction in spatial information systems. In [VL'94, 1994], pages 23–30.

Meyer, B. (1997). Formalization of visual mathematical notations. In *Proceedings of AAAI Symposium on Diagrammatic Reasoning, Boston/MA*, pages 23–30.

Möller, R., Haarslev, V., and 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., and Neumann, B. (1998). Semantics-based information retrieval. In *Proceedings of IT&KNOWS-98: International Conference on Information Technology and Knowledge Systems, 31. August- 4. September, Vienna, Budapest, 1998*, pages 48–61.

Möller, R. and Wessel, M. (1999). Terminological default reasoning about spatial information: A first step. In *Proc. of COSIT'99, International Conference on Spatial Information Theory, Stade*, pages 172–189. Springer Verlag, Berlin.

Najork, M. and Kaplan, S. (1993). Specifying visual languages with conditional set rewrite systems. In *1993 IEEE Symposium on Visual Languages, Bergen, Norway, Aug. 24-27*, pages 12–17. IEEE Computer Society Press, Los Alamitos.

Ohlbach, H. and Köhler, J. (1999). Modal logics, description logics and arithmetic reasoning. *Journal of Artificial Intelligence*, 1-2:1–31.

Olivier, P., Anderson, M., and Meyer, B., editors (2000). *Diagrammatic Representation and Reasoning*. Springer Verlag, Berlin. In print.

Padgham et al., L., editor (1996). *Proceedings of the International Workshop on Description Logics, Nov. 2-4, 1996, Cambridge, Massachusetts*. AAAI Press, Menlo Park. Technical Report WS-96-05.

Pratt, I. and Lemon, O. (1997). Ontologies for plane, polygonal mereotopology. *Notre Dame Journal of Formal Logic*, 2(38):225–245.

Pratt, I. and Schoop, D. (1997). A complete axiom system for polygonal mereotopology of the real plane. Technical Report UMCS-97-2-2, University of Manchester, Department of Computer Science.

Prien, C. (1998). Anwendung der Least-Common-Subsumer-Methode zur Bestimmung von Ähnlichkeiten beim Information-Retrieval mit Beschreibungslogiken. Master's thesis, Fachbereich Informatik, Universität Hamburg.

Randell, D., Cui, Z., and Cohn, A. (1992). A spatial logic based on regions and connections. In Nebel, B., Rich, C., and Swartout, W., editors, *Principles of Knowledge Representation and Reasoning, Cambridge, Mass., Oct. 25-29, 1992*, pages 165–176. Morgan Kaufman.

Reiter, R. and Mackworth, A. (1989). A logical framework for depiction and image interpretation. *Artificial Intelligence*, 41:125–155.

Rekers, J. and Schürr, A. (1995). A graph grammar approach to graphical parsing. In [VL'95, 1995], pages 195–202.

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.

Russ, T., MacGregor, R., Salemi, B., Price, K., and Nevatia, R. (1996). Veil: Combining semantic knowledge with image understanding. In *ARPA Image Understanding Workshop*.

Sattler, U. (1996). A concept language extended with different kinds of transitive roles. In Görz, G. and Hölldobler, S., editors, *20. Deutsche Jahrestagung für Künstliche Intelligenz*, number 1137 in Lecture Notes in Artificial Intelligence, pages 333–345. Springer Verlag, Berlin.

Schild, K. (1991). 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.

Schmidt-Schauss, M. and Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26.

Schröder, C. (1998). *Bildinterpretation durch Modellkonstruktion: Eine Theorie zur rechnergestützten Analyse von Bildern*. Dissertation, Universität Hamburg.

Schröder, C. and Neumann, B. (1996). On the logics of image interpretation: Model-construction in a formal knowledge-representation framework. In *Proceedings of the 1996 IEEE International Conference on Image Processing ICIP-96, Lausanne, September 16-19, 1996*, volume 2, pages 785–788. IEEE Computer Society Press, Los Alamitos.

Schulz, S. and Hahn, U. (2000). Knowledge engineering by large-scale knowledge reuse – Experience from the medical domain. In Cohn, A., Giunchiglia, F., and Selman, B., editors, *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000), Breckenridge, Colorado, USA, 2000*, pages 601–610. Morgan Kaufmann.

Serrano, J. (1995). The use of semantic constraints on diagram editors. In [VL'95, 1995], pages 211–216.

Shin, S.-J. (1994). *The Logical Status of Diagrams*. Cambridge University Press, Cambridge.

Soffer, A. and Samet, H. (1998). Pictorial query specification for browsing through spatially referenced image databases. *Journal of Visual Languages and Computing*, 9(6):567–596.

Sommerville, I. (1995). *Software Engineering*. Addison-Wesley, 5. edition.

Sowa, J., editor (1991). *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. Morgan Kaufmann Publishers, San Mateo.

Spanier, E. (1966). *Algebraic Topology*. McGraw-Hill Book Company, New York.

Stock, O., editor (1997). *Spatial and Temporal Reasoning*. Kluwer Academic Publishers, Dordrecht.

Tarski, A. (1951). *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley, CA.

Tessaris, S. and Gough, G. (1999). ABox reasoning with transitive roles and axioms. In [Lambrix et al., 1999], pages 101–104.

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). Optimization methods for the satisfiability test for description logics with concrete domains (in German). Master's thesis, University of Hamburg, Computer Science Department.

Turhan, A.-Y. and Haarslev, V. (2000). Adapting optimization techniques to description logics with concrete domains. In [Baader and Sattler, 2000], pages 247–256.

VL'94 (1994). *1994 IEEE Symposium on Visual Languages, St. Louis, Missouri, Oct. 4-7*. IEEE Computer Society Press, Los Alamitos.

VL'95 (1995). *1995 IEEE Symposium on Visual Languages, Darmstadt, Germany, Sep. 5-9*. IEEE Computer Society Press, Los Alamitos.

VL'96 (1996). *1996 IEEE Symposium on Visual Languages, Boulder, Colorado, USA, Sep. 3-6*. IEEE Computer Society Press, Los Alamitos.

VL'99 (1999). *1999 IEEE Symposium on Visual Languages, Tokyo, Japan, Sep. 13-16*. IEEE Computer Society Press, Los Alamitos.

Wang, D. and Lee, J. (1993a). Pictorial concepts and a concept-supporting graphical system. *Journal of Visual Languages and Computing*, 4(2):177–199.

Wang, D. and Lee, J. (1993b). Visual reasoning: its formal semantics and applications. *Journal of Visual Languages and Computing*, 4(4):327–356.

Wang, D., Lee, J., and Zeevat, H. (1995). Reasoning with diagrammatic representations. In [Glasgow et al., 1995], pages 339–393.

Wessel, M. (1996). Development of a concept-oriented generic graphic editor in Common Lisp (in German). Bachelor's Thesis (Studienarbeit).

Wessel, M. (1998). A visual language for defining (planar) spatial constellations (in German). Master's thesis, University of Hamburg, Computer Science Department.

Wessel, M. (2000). Obstacles on the way to spatial reasoning with description logics: Undecidability of $\mathcal{ALCRA}^{-}$. Technical Report FBI-HH-M-297/00, University of Hamburg, Computer Science Department.

Wessel, M. and Haarslev, V. (1998). VISCO: Bringing visual spatial querying to reality. In *1998 IEEE Symposium on Visual Languages, Halifax, Canada, Sep. 1-4*, pages 170–177. IEEE Computer Society Press, Los Alamitos.

Wessel, M., Haarslev, V., and Möller, R. (2000). $\mathcal{ALC}_{\mathcal{RA}}$ – $\mathcal{ALC}$ with role axioms. In [Baader and Sattler, 2000], pages 267–276.

Wittenburg, K. (1993). Adventures in multi-dimensional parsing: Cycles and disorders. In *1993 International Workshop on Parsing Technologies, Tilburg, Netherlands and Durbuy, Belgium, Aug. 8-10*.

Wittenburg, K., Weitzman, L., and Talley, J. (1991). Unification-based grammars and tabular parsing for graphical languages. *Journal of Visual Languages and Computing*, 2(4):347–370.

Woods, W. and Schmolze, J. (1992). The KL-ONE family. In Lehmann, F., editor, *Semantic Networks in Artificial Intelligence*, pages 133–177. Pergamon Press, Oxford.

Appendix A

Verifying Satisfiability in $\mathcal{ALCRP}(\mathcal{D})$: An Extended Example

We illustrate the satisfiability problem for an $\mathcal{ALCRP}(\mathcal{S}_2)$ concept with the example from Section 7.4.1. In order to prove that the concept unknown is subsumed by hh_border_district, the tableaux prover constructs an initial ABox and derives that every ABox in the set of ABoxes obtained by applying a set of rules will be "obviously contradictory," i.e. it will contain a clash. The rules are described in detail in [Haarslev et al., 1999b]. For the reader's convenience they are repeated in Appendix B. Note that the rules can be applied in arbitrary order but in the following we rely on an manually defined ordering.

We start with the ABox $\mathcal{A}_1$ and expand in ABox $\mathcal{A}_2$ the concept names from ABox $\mathcal{A}_1$.

$$\mathcal{A}_1 := \{\mathsf{x} : \mathsf{unknown} \sqcap \neg \mathsf{hh\_border\_district}\}$$

$$\mathcal{A}_2 := \left\{ \begin{array}{l} \mathsf{x} : \mathsf{district\_of\_hh} \sqcap \exists\, \mathsf{is\_spatially\_related} \,.\, \mathsf{federal\_state\_hh} \sqcap \\ \quad \exists\, \mathsf{is\_touching} \,.\, \mathsf{federal\_state\_sh} \sqcap \\ \quad \neg(\mathsf{district\_of\_hh} \sqcap \exists\, \mathsf{is\_t\_inside} \,.\, \mathsf{federal\_state\_hh}) \end{array} \right\}$$

If we fully expand the concept terms, we get the following ABox.

$$\mathcal{A}_3 := \left\{ \begin{array}{l} \mathsf{x} : \exists\, \mathsf{has\_area} \,.\, \mathsf{g\_inside}_{\mathsf{p_2}} \sqcap \exists\, \mathsf{has\_area} \,.\, \neg\mathsf{equal}_{\mathsf{p_2}} \sqcap \\ \quad \exists\, \mathsf{is\_spatially\_related} \,.\, \exists\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p_2}} \sqcap \\ \quad \exists\, \mathsf{is\_touching} \,.\, \exists\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p_4}} \sqcap \\ \quad \neg(\exists\, \mathsf{has\_area} \,.\, \mathsf{g\_inside}_{\mathsf{p_2}} \sqcap \exists\, \mathsf{has\_area} \,.\, \neg\mathsf{equal}_{\mathsf{p_2}} \sqcap \\ \qquad \exists\, \mathsf{is\_t\_inside} \,.\, \exists\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p_2}}) \end{array} \right\}$$

Then, we transform this ABox in negation normal form.

$$\mathcal{A}_4 := \left\{ \begin{array}{l} \mathsf{x} : \exists\, \mathsf{has\_area} \,.\, \mathsf{g\_inside}_{\mathsf{p}_2} \sqcap \exists\, \mathsf{has\_area} \,.\, \neg\mathsf{equal}_{\mathsf{p}_2} \sqcap \\ \quad \exists\, \mathsf{is\_spatially\_related} \,.\, \exists\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p}_2} \sqcap \\ \quad \exists\, \mathsf{is\_touching} \,.\, \exists\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p}_4} \sqcap \\ \quad (\exists\, \mathsf{has\_area} \,.\, \neg\mathsf{g\_inside}_{\mathsf{p}_2} \sqcup \forall\, \mathsf{has\_area} \,.\, \top \sqcup \exists\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p}_2} \sqcup \\ \quad \forall\, \mathsf{is\_t\_inside} \,.\, (\exists\, \mathsf{has\_area} \,.\, \neg\mathsf{equal}_{\mathsf{p}_2} \sqcup \forall\, \mathsf{has\_area} \,.\, \top)) \end{array} \right\}$$

We already proved that the TBox containing the axioms introduced in Section 7.4.1 complies to the restrictedness criterion. Therefore, the ABox $\mathcal{A}_4$ is based on a restricted terminology and we are allowed to apply the ABox rules of the calculus for $\mathcal{ALCRP}(\mathcal{D})$ (see Appendix B). First, we apply the *and rule* ($\mathbf{R}\sqcap$) and get the ABox $\mathcal{A}_6$. For convenience we use an auxiliary ABox $\mathcal{A}_5$.

$$\mathcal{A}_5 := \left\{ \begin{array}{l} \mathsf{x} : \exists\, \mathsf{has\_area} \,.\, \mathsf{g\_inside}_{\mathsf{p}_2} \\ \mathsf{x} : \exists\, \mathsf{has\_area} \,.\, \neg\mathsf{equal}_{\mathsf{p}_2} \\ \mathsf{x} : \exists\, \mathsf{is\_spatially\_related} \,.\, \exists\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p}_2} \\ \mathsf{x} : \exists\, \mathsf{is\_touching} \,.\, \exists\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p}_4} \end{array} \right\}$$

$$\mathcal{A}_6 := \mathcal{A}_5 \cup \left\{ \begin{array}{l} \mathsf{x} : (\exists\, \mathsf{has\_area} \,.\, \neg\mathsf{g\_inside}_{\mathsf{p}_2} \sqcup \forall\, \mathsf{has\_area} \,.\, \top \sqcup \\ \quad \exists\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p}_2} \sqcup \\ \quad \forall\, \mathsf{is\_t\_inside} \,.\, (\exists\, \mathsf{has\_area} \,.\, \neg\mathsf{equal}_{\mathsf{p}_2} \sqcup \forall\, \mathsf{has\_area} \,.\, \top)) \end{array} \right\}$$

Afterwards we obtain four alternative ABoxes ($\mathcal{A}_7$ - $\mathcal{A}_{10}$) by resolving the disjunctions in ABox $\mathcal{A}_6$ and by applying the *exists-in over predicates rule* ($\mathbf{R}\exists\mathbf{P}$) and/or the *all rule* ($\mathbf{R}\forall\mathbf{C}$).

$$\mathcal{A}_7 := \mathcal{A}_5 \cup \left\{ \begin{array}{l} \mathsf{x} : \exists\, \mathsf{has\_area} \,.\, \neg\mathsf{g\_inside}_{\mathsf{p}_2} \\ (\mathsf{x}, \mathsf{q}_2) : \mathsf{has\_area},\ \mathsf{q}_2 : \mathsf{g\_inside}_{\mathsf{p}_2},\ \mathsf{q}_2 : \neg\mathsf{g\_inside}_{\mathsf{p}_2} \end{array} \right\}$$

The ABox $\mathcal{A}_7$ contains a *concrete domain clash* because the concrete individual q_2 can not satisfy the conjunction ($\mathsf{g\_inside}_{\mathsf{p}_2} \wedge \neg\mathsf{g\_inside}_{\mathsf{p}_2}$).

$$\mathcal{A}_8 := \mathcal{A}_5 \cup \left\{ \begin{array}{l} \mathsf{x} : \forall\, \mathsf{has\_area} \,.\, \top \\ (\mathsf{x}, \mathsf{q}_2) : \mathsf{has\_area},\ \mathsf{q}_2 : \mathsf{g\_inside}_{\mathsf{p}_2},\ \mathsf{q}_2 : \top \end{array} \right\}$$

The ABox $\mathcal{A}_8$ contains an *all domain clash* because the concrete individual q_2 can not be a member of both the abstract ($\top$) and the concrete (e.g. $\mathsf{g\_inside}_{\mathsf{p}_2}$) domain.

$$\mathcal{A}_9 := \mathcal{A}_5 \cup \left\{ \begin{array}{l} \mathsf{x} : \exists\, \mathsf{has\_area} \,.\, \mathsf{equal}_{\mathsf{p}_2} \\ (\mathsf{x}, \mathsf{q}_2) : \mathsf{has\_area},\ \mathsf{q}_2 : \neg\mathsf{equal}_{\mathsf{p}_2},\ \mathsf{q}_2 : \mathsf{equal}_{\mathsf{p}_2} \end{array} \right\}$$

The ABox $\mathcal{A}_9$ contains a *concrete domain clash* because the concrete individual q_2 can not satisfy the conjunction ($\mathsf{equal}_{\mathsf{p}_2} \wedge \neg\mathsf{equal}_{\mathsf{p}_2}$).

$$\mathcal{A}_{10} := \mathcal{A}_5 \cup \left\{ \mathsf{x} : \forall\, \mathsf{is\_t\_inside} \,.\, (\exists\, \mathsf{has\_area} \,.\, \neg\mathsf{equal}_{\mathsf{p}_2} \sqcup \forall\, \mathsf{has\_area} \,.\, \top) \right\}$$

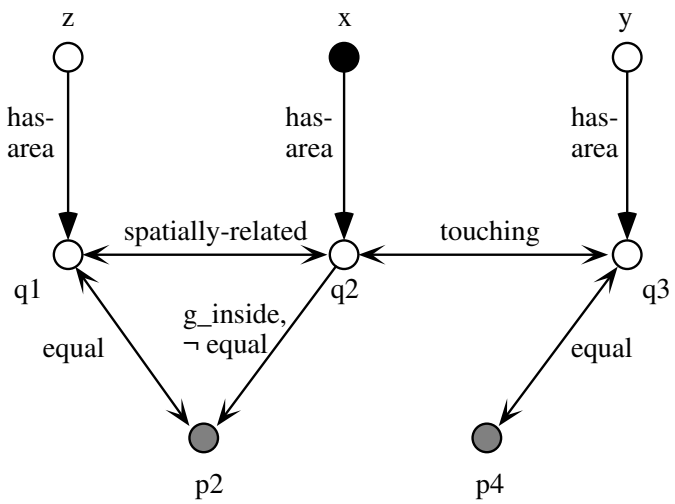


Figure A.1: Initial constraint network corresponding to ABox $\mathcal{A}_{12}$. For symmetric relations the arrows point in both directions. Inverse relations have been omitted.

The ABox $\mathcal{A}_{10}$ is subject to further rule application. The *exists-in over predicates rule* (**R**$\exists$**P**) and the *role-forming exists-in over predicates rule* (**Rr**$\exists$**P**) are applied and they create two abstract domain individuals y and z such that z is a filler of the role $\mathsf{is\_spatially\_related}$ and y is a filler of $\mathsf{is\_touching}$. Three concrete domain individuals q_2, q_3, and q_1 are also created that are associated with their corresponding abstract individuals via the attribute $\mathsf{has\_area}$. The rules also establish spatial relations that have to hold between concrete individuals. After firing all applicable rules except the *choose rule* (**RChoose**), we get ABox $\mathcal{A}_{12}$ whose spatial constraints are illustrated in Figure A.1. For convenience we use an auxiliary ABox $\mathcal{A}_{11}$.

$$\mathcal{A}_{11} := \mathcal{A}_5 \cup \left\{ \begin{array}{r} (\mathsf{x}, \mathsf{q}_2) : \mathsf{has\_area},\ \mathsf{q}_2 : \mathsf{g\_inside}_{\mathsf{p}_2},\ \mathsf{q}_2 : \neg\mathsf{equal}_{\mathsf{p}_2} \\ (\mathsf{x}, \mathsf{y}) : \exists\,(\mathsf{has\_area})(\mathsf{has\_area})\,.\,\mathsf{touching} \\ (\mathsf{q}_2, \mathsf{q}_3) : \mathsf{touching} \\ (\mathsf{y}, \mathsf{q}_3) : \mathsf{has\_area},\ \mathsf{q}_3 : \mathsf{equal}_{\mathsf{p}_4} \\ (\mathsf{x}, \mathsf{z}) : \exists\,(\mathsf{has\_area})(\mathsf{has\_area})\,.\,\mathsf{spatially\_related} \\ (\mathsf{q}_2, \mathsf{q}_1) : \mathsf{spatially\_related} \end{array} \right\}$$

$$\mathcal{A}_{12} := \mathcal{A}_{11} \cup \left\{ \begin{array}{r} \mathsf{x} : \forall\,\mathsf{is\_t\_inside}\,.\,(\exists\,\mathsf{has\_area}\,.\,\neg\mathsf{equal}_{\mathsf{p}_2} \sqcup \forall\,\mathsf{has\_area}\,.\,\top) \\ (\mathsf{z}, \mathsf{q}_1) : \mathsf{has\_area},\ \mathsf{q}_1 : \mathsf{equal}_{\mathsf{p}_2} \end{array} \right\}$$

In the next step, the *choose rule* (**RChoose**) has to decide whether the relation $\mathsf{t\_inside}$ or its negation holds between any two concrete individuals in ABox $\mathcal{A}_{12}$. Without loss of generality we can assume that only the following two alternative ABoxes ($\mathcal{A}_{13}$, $\mathcal{A}_{15}$) are created by selecting the concrete individuals $\mathsf{q}_1, \mathsf{q}_2$.

$$\mathcal{A}_{13} := \mathcal{A}_{11} \cup \left\{ \begin{array}{r} \mathsf{x} : \forall\,\mathsf{is\_t\_inside}\,.\,(\exists\,\mathsf{has\_area}\,.\,\neg\mathsf{equal}_{\mathsf{p}_2} \sqcup \forall\,\mathsf{has\_area}\,.\,\top) \\ (\mathsf{z}, \mathsf{q}_1) : \mathsf{has\_area},\ \mathsf{q}_1 : \mathsf{equal}_{\mathsf{p}_2} \\ (\mathsf{q}_2, \mathsf{q}_1) : \mathsf{t\_inside} \end{array} \right\}$$

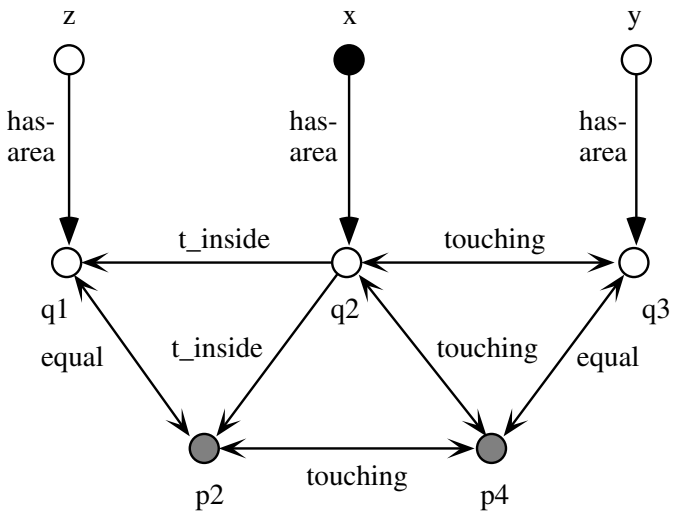


Figure A.2: Final constraint network (most of the implicit constraints are added) which corresponds to ABox $\mathcal{A}_{13}$. For symmetric relations the arrows point in both directions. Inverse relations have been omitted.

The spatial constraints that have to hold in ABox $\mathcal{A}_{13}$ are illustrated in Figure A.2. This ABox assumes that $\mathsf{t\_inside}(\mathsf{q}_2, \mathsf{q}_1)$ holds and makes the implicit spatial constraints from ABox $\mathcal{A}_{12}$ explicit (see also Figure A.1). Due to the last assertion in ABox $\mathcal{A}_{13}$, now the *all rule*($\mathbf{R}\forall\mathbf{C}$) is applicable to $\mathsf{is\_t\_inside}$ and creates the ABox $\mathcal{A}_{14}$.

$$\mathcal{A}_{14} := \mathcal{A}_{11} \cup \left\{ \begin{array}{r} \mathsf{x} : \forall\, \mathsf{is\_t\_inside} \,.\, (\exists\, \mathsf{has\_area} \,.\, \neg \mathsf{equal}_{\mathsf{p}_2} \sqcup \forall\, \mathsf{has\_area} \,.\, \top) \\ (\mathsf{z}, \mathsf{q}_1) : \mathsf{has\_area},\ \mathsf{q}_1 : \mathsf{equal}_{\mathsf{p}_2} \\ (\mathsf{q}_2, \mathsf{q}_1) : \mathsf{t\_inside} \\ \mathsf{z} : \exists\, \mathsf{has\_area} \,.\, \neg \mathsf{equal}_{\mathsf{p}_2} \sqcup \forall\, \mathsf{has\_area} \,.\, \top \end{array} \right\}$$

Caused by the disjunction in the last assertion, we get two descendants of ABox $\mathcal{A}_{14}$. However, both descendants contain clashes for the concrete individual q_1 (i.e. either an *all domain clash* or a *concrete domain clash*) and eliminate this branch.

It remains the ABox $\mathcal{A}_{15}$ as the second alternative descendant of ABox $\mathcal{A}_{12}$. The ABox $\mathcal{A}_{15}$ assumes that $\neg\mathsf{t\_inside}(\mathsf{q}_2, \mathsf{q}_1)$ holds. It has all spatial constraints expanded.

$$\mathcal{A}_{15} := \mathcal{A}_5 \cup \left\{ \begin{array}{r} (\mathsf{x}, \mathsf{q}_2) : \mathsf{has\_area},\ \mathsf{q}_2 : \mathsf{g\_inside}_{\mathsf{p}_2},\ \mathsf{q}_2 : \neg\mathsf{equal}_{\mathsf{p}_2} \\ (\mathsf{x}, \mathsf{y}) : \exists\, (\mathsf{has\_area})(\mathsf{has\_area}) \,.\, \mathsf{touching} \\ (\mathsf{q}_2, \mathsf{q}_3) : \mathsf{touching} \\ (\mathsf{y}, \mathsf{q}_3) : \mathsf{has\_area},\ \mathsf{q}_3 : \mathsf{equal}_{\mathsf{p}_4} \\ (\mathsf{x}, \mathsf{z}) : \exists\, (\mathsf{has\_area})(\mathsf{has\_area}) \,.\, \mathsf{spatially\_related} \\ (\mathsf{q}_2, \mathsf{q}_1) : \mathsf{spatially\_related} \\ \mathsf{x} : \forall\, \mathsf{is\_t\_inside} \,.\, (\exists\, \mathsf{has\_area} \,.\, \neg \mathsf{equal}_{\mathsf{p}_2} \sqcup \forall\, \mathsf{has\_area} \,.\, \top) \\ (\mathsf{z}, \mathsf{q}_1) : \mathsf{has\_area},\ \mathsf{q}_1 : \mathsf{equal}_{\mathsf{p}_2} \\ (\mathsf{q}_2, \mathsf{q}_1) : \neg\mathsf{t\_inside} \end{array} \right\}$$

ABox $\mathcal{A}_{15}$ contains a *concrete domain clash*. In order to check for concrete domain clashes, the calculus for $\mathcal{ALCRP}(\mathcal{S}_2)$ invokes the satisfiability test for $\mathcal{S}_2$ with the conjunction $\mathcal{C}_0$ of spatial predicate terms. This conjunction represents all spatial relations that have to hold between the concrete individuals in ABox $\mathcal{A}_{15}$.

$$\mathcal{C}_0 := \left\{ \begin{array}{c} \mathsf{g\_inside}(\mathsf{q}_2, \mathsf{p}_2) \wedge \neg\mathsf{equal}(\mathsf{q}_2, \mathsf{p}_2) \wedge \mathsf{touching}(\mathsf{q}_2, \mathsf{q}_3) \wedge \mathsf{equal}(\mathsf{q}_3, \mathsf{p}_4) \wedge \\ \mathsf{spatially\_related}(\mathsf{q}_2, \mathsf{q}_1) \wedge \mathsf{equal}(\mathsf{q}_1, \mathsf{p}_2) \wedge \neg\mathsf{t\_inside}(\mathsf{q}_2, \mathsf{q}_1) \end{array} \right\}$$

Some of the terms in the conjunction $\mathcal{C}_0$ contain references to the two concrete polygons p_2 and p_4 that have known positions (see Figure 6.5). During the satisfiability test the elementary spatial relation which holds between these polygons is automatically added. This results in the conjunction $\mathcal{C}_0 \wedge \mathsf{touching}(\mathsf{p}_2, \mathsf{p}_4)$ which is unsatisfiable in the spatial domain $\mathcal{S}_2$.

Appendix B

The Calculus for $\mathcal{ALCRP}(\mathcal{D})$

The following two sections are excerpts taken from [Haarslev et al., 1999b]. They are given here because Appendix A relies on the rules and clash triggers.

B.1 Completion Rules

Before the completion rules can be defined, we introduce some technical terms. Let $\mathcal{A}$ be an ABox, R be a role term, a and b be object names from O_A, γ be a symbol that is not element of O_D, u be a feature chain $f_1 \ldots f_k$, and let $u_1, \ldots, u_n$ and $v_1, \ldots, v_m$ (possibly with index) be arbitrary feature chains. For convenience we define three functions as follows:

$$\mathit{filler}_{\mathcal{A}}(a,u) := \begin{cases} x \text{ where } x \in \mathsf{O}_D \text{ such that} \\ \quad \exists b_1,\ldots,b_{k-1} \in \mathsf{O}_A\text{:} \\ \quad\quad ((a,b_1):f_1 \in \mathcal{A},\ldots,(b_{k-1},x):f_k \in \mathcal{A}) \\ \gamma \text{ if no such } x \text{ exists.} \end{cases}$$

$$\mathit{createchain}_{\mathcal{A}}(a,x,u) := \{(a,c_1):f_1,\ldots,(c_{k-1},x):f_k\}$$
$$\text{where } c_1,\ldots,c_{k-1} \in \mathsf{O}_A \text{ are not used in } \mathcal{A}.$$

$$\mathit{related}_{\mathcal{A}}(a,b,R) := \begin{cases} \mathit{true} \text{ if } (a,b):R \in \mathcal{A} \\ \mathit{true} \text{ if } R \text{ is of the form } \exists(u_1,\ldots,u_n)(v_1,\ldots,v_m).P, \\ \qquad \text{and } \exists x_1,\ldots,x_n,y_1,\ldots,y_m \in \mathsf{O}_D \text{ such that} \\ \qquad \mathit{filler}_{\mathcal{A}}(a,u_1)=x_1,\ \ldots\ ,\mathit{filler}_{\mathcal{A}}(a,u_n)=x_n, \\ \qquad \mathit{filler}_{\mathcal{A}}(b,v_1)=y_1,\ \ldots\ ,\mathit{filler}_{\mathcal{A}}(b,v_m)=y_m, \\ \qquad (x_1,\ldots,x_n,y_1,\ldots,y_m):P \in \mathcal{A} \\ \mathit{false} \text{ otherwise} \end{cases}$$

Let $\mathcal{A}$ be an ABox, f be a feature name, a, b, c be object names from O_A, and x, y be object names from O_D. If $\mathcal{A}$ contains the constraints $(a,b):f$ and $(a,c):f$ (resp. $(a,x):f$ and $(a,y):f$) then this pair of constraints is called a *fork* in $\mathcal{A}$. Since f is interpreted

as a partial function, b and c (resp. x and y) have to be interpreted as the same objects. Each ABox is checked for forks immediately after a completion rule was applied. If a fork is detected, all occurrences of c in $\mathcal{A}$ are replaced by b (resp. y by x). Before any rule is applied to the initial ABox $\mathcal{A}_0$, any forks in $\mathcal{A}_0$ have to be eliminated. It is easy to prove that fork elimination preserves (in)consistency by showing that a model $\mathcal{I}$ for an ABox $\mathcal{A}$ is also a model for an ABox $\mathcal{A}'$ which is obtained from $\mathcal{A}$ by fork elimination.

Definition B.1 The following *completion rules* will replace an ABox $\mathcal{A}$ by an ABox $\mathcal{A}'$ or by two ABoxes $\mathcal{A}'$ and $\mathcal{A}''$ (*descendants* of $\mathcal{A}$). In the following C and D denote concept terms, R denotes a role term, and P denotes a predicate name from $\Phi_{\mathcal{D}}$. Let $f_1, \ldots, f_n$ as well as $g_1, \ldots, g_n$ denote feature names, and $u_1, \ldots, u_m$ denote feature chains. a and b denote object names from O_A.

R⊓ The conjunction rule.
Premise: $a : C \sqcap D \in \mathcal{A}, \quad a : C \notin \mathcal{A} \vee a : D \notin \mathcal{A}$
Consequence: $\mathcal{A}' = \mathcal{A} \cup \{a : C,\ a : D\}$

R⊔ The disjunction rule.
Premise: $a : C \sqcup D \in \mathcal{A}, \quad a : C \notin \mathcal{A} \wedge a : D \notin \mathcal{A}$
Consequence: $\mathcal{A}' = \mathcal{A} \cup \{a : C\},\ \mathcal{A}'' = \mathcal{A} \cup \{a : D\}$

R∃C The concept exists restriction rule.
Premise: $a : \exists R.C \in \mathcal{A}, \quad \neg\exists b \in \mathsf{O}_A : (\mathit{related}_{\mathcal{A}}(a, b, R) \wedge b : C \in \mathcal{A})$
Consequence: $\mathcal{A}' = \mathcal{A} \cup \{(a, b) : R\ ,\ b : C\}$ where $b \in \mathsf{O}_A$ is not used in $\mathcal{A}$.
This rule may create a fork if R is a feature.

R∀C The concept value restriction rule.
Premise: $a : \forall R.C \in \mathcal{A}, \quad \exists b \in \mathsf{O}_A : (\mathit{related}_{\mathcal{A}}(a, b, R), \wedge b : C \notin \mathcal{A})$
Consequence: $\mathcal{A}' = \mathcal{A} \cup \{b : C\}$

R∃P The predicate restriction rule.
Premise: $a : \exists u_1, \ldots, u_n.P \in \mathcal{A}, \neg\exists x_1, \ldots, x_n \in \mathsf{O}_D :$
$(\mathit{filler}_{\mathcal{A}}(a, u_1) = x_1 \wedge \ \ldots\ \wedge \mathit{filler}_{\mathcal{A}}(a, u_n) = x_n \wedge$
$(x_1, \ldots, x_n) : P \in \mathcal{A})$
Consequence: $\mathcal{A}' = \mathcal{A} \cup \{(x_1, \ldots, x_n) : P\} \cup$
$\mathit{createchain}_{\mathcal{A}}(a, x_1, u_1) \cup \ldots \cup \mathit{createchain}_{\mathcal{A}}(a, x_n, u_n)$
where the $x_i \in \mathsf{O}_D$ are not used in $\mathcal{A}$.
This rule may create forks.

Rr∃P The role-forming predicates restriction rule.
Premise: $(a, b) : \exists(u_1, \ldots, u_n)(v_1, \ldots, v_m).P \in \mathcal{A},$
$\neg\exists x_1, \ldots, x_n, y_1, \ldots, y_m \in \mathsf{O}_D :$
$(\mathit{filler}_{\mathcal{A}}(a, u_1) = x_1 \wedge \ \ldots\ \wedge \mathit{filler}_{\mathcal{A}}(a, u_n) = x_n \wedge$
$\mathit{filler}_{\mathcal{A}}(b, v_1) = y_1 \wedge \ \ldots\ \wedge \mathit{filler}_{\mathcal{A}}(b, v_m) = y_m \wedge$
$(x_1, \ldots, x_n, y_1, \ldots, y_m) : P \in \mathcal{A})$

Consequence: $\mathcal{A}' = \mathcal{A} \cup \{(x_1, \ldots, x_n, y_1, \ldots, y_m) : P\} \cup$
$createchain_{\mathcal{A}}(a, x_1, u_1) \cup \ldots \cup createchain_{\mathcal{A}}(a, x_n, u_n) \cup$
$createchain_{\mathcal{A}}(b, y_1, v_1) \cup \ldots \cup createchain_{\mathcal{A}}(b, y_m, v_m)$
where the $x_i \in \mathsf{O}_D$ and $y_i \in \mathsf{O}_D$ are not used in $\mathcal{A}$.
This rule may create forks.

RChoose The choose rule.
Premise: $a : \forall(\exists(u_1, \ldots, u_n)(v_1, \ldots, v_m).P).C \in \mathcal{A}$,
$\exists b \in \mathsf{O}_A,\ x_1, \ldots, x_n, y_1, \ldots, y_m \in \mathsf{O}_D :$
$(filler_{\mathcal{A}}(a, u_1) = x_1 \wedge \ \ldots \ \wedge filler_{\mathcal{A}}(a, u_n) = x_n \wedge$
$filler_{\mathcal{A}}(b, v_1) = y_1 \wedge \ \ldots \ \wedge filler_{\mathcal{A}}(b, v_m) = y_m \wedge$
$(x_1, \ldots, x_n, y_1, \ldots, y_m) : P \notin \mathcal{A} \wedge$
$(x_1, \ldots, x_n, y_1, \ldots, y_m) : \overline{P} \notin \mathcal{A})$
Consequence: $\mathcal{A}' = \mathcal{A} \cup \{(x_1, \ldots, x_n, y_1, \ldots, y_m) : P\}$,
$\mathcal{A}'' = \mathcal{A} \cup \{(x_1, \ldots, x_n, y_1, \ldots, y_m) : \overline{P}\}$

B.2 Clash Rules

Termination of the algorithm applying the completion rules is proven in [Haarslev et al., 1999b]. The proof shows that after a finite number of rule applications a tree Υ of ABoxes is obtained for which one of the following conditions holds:

1. it contains an ABox $\mathcal{A}$ which is complete or
2. all leaf ABoxes in the tree contain a clash.

In both cases no more completion rules are applicable. In the following we formalize the notion "to contain a clash."

Definition B.2 Let the same naming conventions be given as in Definition B.1. Additionally, let f be a feature. An ABox $\mathcal{A}$ contains a *clash* if any of the following *clash triggers* are applicable:

Primitive Clash
$a : C \in \mathcal{A}, \quad a : \neg C \in \mathcal{A}$

Feature Domain Clash
$(a, x) : f \in \mathcal{A}, \quad (a, b) : f \in \mathcal{A}$

All Domain Clash
$(a, x) : f \in \mathcal{A}, \quad a : \forall f.C \in \mathcal{A}$

Concrete Domain Clash
$(x_1^{(1)}, \ldots, x_{n_1}^{(1)}) : P_1 \in \mathcal{A}, \ldots, (x_1^{(k)}, \ldots, x_{n_k}^{(k)}) : P_k \in \mathcal{A}$ and the corresponding conjunction $\bigwedge_{i=1}^{k} P_i(x^{(i)})$ is not satisfiable in $\mathcal{D}$. This can be decided because $\mathcal{D}$ is required to be admissible.