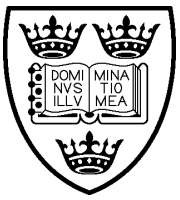
Scalable Reasoning for Description Logics



Rob Shearer

Linacre College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2010

Abstract

Description logics (DLs) are knowledge representation formalisms with well-understood model-theoretic semantics and computational properties. The DL $\mathcal{SROIQ}$ provides the logical underpinning for the semantic web language OWL 2, which is quickly becoming the standard for knowledge representation on the web.

A central component of most DL applications is an efficient and scalable reasoner, which provides services such as consistency testing and classification. Despite major advances in DL reasoning algorithms over the last decade, however, ontologies are still encountered in practice that cannot be handled by existing DL reasoners.

We present a novel reasoning calculus for the description logic $\mathcal{SROIQ}$ which addresses two of the major sources of inefficiency present in the tableau-based reasoning calculi used in state-of-the-art reasoners: unnecessary nondeterminism and unnecessarily large model sizes. Further, we describe a new approach to classification which exploits partial information about the subsumption relation between concept names to reduce both the number of individual subsumption tests performed and the cost of working with large ontologies; our algorithm is applicable to the general problem of deducing a quasi-ordering from a sequence of binary comparisons. We also present techniques for extracting partial information about the subsumption relation from the models generated by constructive DL reasoning methods, such as our hypertableau calculus.

Empirical results from a prototypical implementation demonstrate substantial performance improvements compared to existing algorithms and implementations.

Contents

I Foundations **1**

1 Introduction **2**

2 Description Logics **9**

2.1 Syntax and Semantics 10

II Model Generation **15**

3 Difficulties **16**

3.1 Traditional Tableau Algorithms 16

3.2 Or-Branching 17

3.3 And-Branching 20

3.4 Problems Due to Merging 23

3.5 Nominals, Inverses, and Number Restrictions 24

3.5.1 Promoting Blockable Individuals 25

3.5.2 The Traditional Tableau Solution 27

3.5.3 The *NI*-rule 29

3.5.4 Annotated Equalities 30

3.5.5 Nominals and Merging 31

3.5.6 The *NI*-Rule and Unraveling 33

4 Algorithm Overview **35**

4.1 Derivation Rules . . . 35

4.2 Calculus Overview . . . 37

4.3 Anywhere Pairwise Blocking . . . 38

4.4 Nominal Guard Concepts . . . 39

5 The Hypertableau Calculus for $\mathcal{SROIQ}$ **40**

5.1 Preprocessing . . . 40

5.1.1 Elimination of Role Inclusion Axioms . . . 44

5.1.2 Normalization . . . 51

5.1.3 Translation into DL-Clauses . . . 55

5.2 The Hypertableau Calculus for HT-Clauses . . . 59

6 Optimizations **86**

6.1 Caching Blocking Labels . . . 86

6.2 Single Blocking . . . 87

6.3 Subset Blocking . . . 92

6.4 The Number of Blockable Individuals . . . 94

6.5 The Number of Root Individuals . . . 97

7 Related Work **100**

7.1 Hypertableau vs. Absorption . . . 100

7.2 Relationship with Caching . . . 104

7.3 Relationship with First-Order Calculi . . . 105

III Classification and Retrieval **108**

8 Overview **109**

8.1 Difficulties . . . 110

8.2 Algorithm Summary 112

9 Deducing a Quasi-Ordering **114**

9.1 Preliminaries 114

9.2 Maximizing Partial Information 115

9.3 Taxonomy Construction and Searching 117

9.4 Example 120

10 Extracting Subsumption Information From Models **123**

10.1 Identifying Non-Subsumptions 123

10.2 Identifying Subsumptions 125

11 Related Work **129**

IV Evaluation **131**

12 Empirical Results **132**

12.1 Reasoning Performance 132

12.1.1 Implementing Anywhere Blocking 134

12.1.2 Test Procedure 135

12.1.3 Results 137

12.2 Classification Performance 141

12.2.1 Comparison with the Enhanced Traversal Method 141

12.2.2 Overall Performance 143

13 Conclusion **147**

13.1 Contributions 147

13.2 Significance 149

13.3 Future Work 149

A Reasoning Performance Data 151

References 169

Part I

Foundations

Chapter 1

Introduction

Description Logics (DLs) [Baader *et al.*, 2007] are a family of knowledge representation formalisms with well-understood formal properties. DLs have been applied to numerous problems in computer science such as information integration and metadata management, and have received extensive use in fields as diverse as biology [Sidhu *et al.*, 2005], medicine [Golbreich *et al.*, 2006], geography [Goodwin, 2005], astronomy [Derriere *et al.*, 2006], geology,[1] agriculture [Soergel *et al.*, 2004], and defense [Lacy *et al.*, 2005]. Furthermore, DLs have been used to develop several large biomedical ontologies, such as the Biological Pathways Exchange (BioPAX) ontology [Ruttenberg *et al.*, 2005], the GALEN ontology [Rector and Rogers, 2006], the Foundational Model of Anatomy (FMA) [Golbreich *et al.*, 2006], and the National Cancer Institute thesaurus [Hartel *et al.*, 2005]. Recent interest in DLs has been spurred by their application in the Semantic Web: the DL $\mathcal{SHOIQ}$ provides the logical underpinning for the Web Ontology Language (OWL) [Patel-Schneider *et al.*, 2004], and the DL $\mathcal{SROIQ}$ [Kutz *et al.*, 2006] is used in its successor OWL 2.

A central component of most DL applications is an efficient and scalable reasoner which allows interrogation of information implied by a knowledge base but not encoded explicitly. Such reasoners provide services such as *consistency checking* (searching for contradictions between information in the knowledge base), *member-*

[1] http://sweet.jpl.nasa.gov/ontology/

ship testing (determining whether a given instance is a member of a given concept expression), *realization* (discovering all concept names from the input knowledge base of which a given instance is a member), *conjunctive query answering* (identifying all tuples of instances from the input knowledge base which together match/instantiate a query pattern), *entailment checking* (determining whether one knowledge base is a logical consequence of another), and *explanation* (providing accessible information to users about the reasons for results provided by other services).

One of the core services provided by most reasoners is *classification*, the discovery of all subsumption relationships between concept names in the input knowledge base. In fact, classification is typically the primary (and often the only) reasoning service exposed by ontology engineering tools [Lutz *et al.*, 2006]. The Protégé-OWL editor [Knublauch *et al.*, 2004], for example, includes a "Reasoning" button which performs classification. The resulting hierarchy of subsumption relationships is used to organize concept names within all aspects of Protégé's interface, and the subsumption relationships which arise as implicit consequences of an ontology are the primary mechanism authors use to check that the axioms they write are consistent with their intuitions about the structure of the domain. Finally, other reasoning services, such as explanation and query answering, typically exploit a cached version of the classification results; classification is thus usually the first task performed by a reasoner.

In some cases (e.g. for less expressive ontology languages such as $\mathcal{EL}$ [Baader, 2003]) it is possible to implement high-level reasoning services such as classification directly. In most cases, however, high-level services are implemented by testing the consistency of a number of knowledge bases generated by extending the input KB with additional information. For example, to determine whether an instance a is a member of a concept C with respect to knowledge base $\mathcal{K}$ (which is known to be consistent), $\mathcal{K}$ is extended with the assertion that a is not a member of C; if the resulting knowledge base is inconsistent then it is a logical consequence of $\mathcal{K}$ that a

is a member of C. Consistency testing thus forms the core of most DL reasoners.

The theoretical complexity of the consistency-testing problem varies with the ontology language used: knowledge bases that make use of only conjunction and existential restriction (and thus can be encoded in the $\mathcal{EL}$ language) can be tested for consistency in polynomial time [Baader, 2003], while the use of negation and universal restrictions (as allowed by the $\mathcal{ALC}$ language, which subsumes $\mathcal{EL}$) pushes complexity to ExpTime [Schmidt-Schauß and Smolka, 1991]. In this thesis we primarily address reasoning over knowledge bases encoded in the $\mathcal{SROIQ}$ language, which in turn subsumes $\mathcal{ALC}$; consistency testing for $\mathcal{SROIQ}$ is N2ExpTime [Kazakov, 2008]. Despite these high worst-case complexities, reasoning procedures have been developed which allow real-world performance on human-generated knowledge bases sufficient for a wide range of applications.

Tableau Reasoning

Modern reasoners typically show consistency of a knowledge base by building (an abstraction of) a model consistent with that knowledge base. There is an extremely large space of potential models, and most modern reasoners, such as Pellet [Parsia and Sirin, 2004], FaCT++ [Tsarkov and Horrocks, 2006], and RACER [Haarslev and Möller, 2001c], perform an exhaustive search of a representative subset of this space using algorithms based on tableau calculi [Baader and Nutt, 2007]. Such a calculus constructs partial models incrementally by repeatedly applying a set of *expansion rules*, each of which elaborates the model with further constraints implied by the axioms of the knowledge base. Numerous optimizations have been developed in an effort to reduce the size of the search space [Horrocks, 2007]. Despite major advances in tableau reasoning algorithms, however, ontologies are still encountered in practice that cannot be handled by existing DL reasoners. Such limitations have been attributed to two primary sources of inefficiency in tableau calculi [Donini, 2007].

This first well-known source of inefficiency is called *or-branching*: given a disjunctive assertion in the knowledge base, a tableau algorithm nondeterministically guesses which of the disjuncts holds. To show the unsatisfiability of a knowledge base, *every* possible guess must lead to a contradiction: if the initial guess leads to a contradiction, the algorithm must backtrack and explore another disjunct, which can give rise to exponential behavior. In DL knowledge bases, terminological axioms are the main source of disjunctions: to ensure that such an axiom holds, a tableau algorithm adds a disjunction to every individual introduced in the model.

While it is possible to encode genuinely complex constraint-satisfaction problems in description logic knowledge bases, few real-world ontologies require such general-purpose treatment of disjunction: only a tiny minority of nondeterministic guesses interact in practice, and in many cases the processing of certain types of disjunction can be deferred until a potential interaction is detected. Various *absorption* optimizations [Horrocks, 1998; Tsarkov and Horrocks, 2004; Hudek and Weddell, 2006; Horrocks, 2007] have been developed to reduce the nondeterminism in tableau calculi. We discuss this problem in more detail in Section 3.2.

The second well-known source of inefficiency in tableau calculi is called *and-branching*: the expansion of a model due to existential quantifiers can generate very large models. Such large models can quickly exceed the primary storage available to implementations, and they also greatly magnify problems resulting from or-branching: the cost of revising nondeterministic guesses grows with the size of the model, as does the number of individuals to which GCIs are applied; the problem of large models is further described in Section 3.2. Our results (presented in Chapter 12) demonstrate that much of the extra work introduced by and-branching is redundant: the large models typically produced by tableau reasoners contain multiple copies of identical sub-models.

We further identify a third (more esoteric) source of inefficiency in tableau calculi,

which we call *at-most branching*. It is possible for three different language features—number restrictions, inverse roles, and nominals—to interact in a complex way which requires special handling; we defer the details to Section 3.5. The standard solution to this problem involves making a nondeterministic guess as to the number of neighbors of certain individuals in a model labeled with at-most restrictions, and then instantiating these neighbors. Such an approach leads to the problems introduced by or-branching as well as those caused by and-branching: a high degree of nondeterminism combined with large models. As we show in Section 3.5, such complexity in unnecessary in most cases.

Finally, our informal experience is that traditional tableau algorithms are notoriously difficult to implement and optimize. Execution typically involves complex interlacing of many different expansion rules, resulting in performance dependent upon a large and complex inner loop. Efficient implementation of such an architecture requires a robust understanding of the overall algorithm, which is difficult to break down into independent sub-components. What is more, the specialized nature of tableau algorithms means that there are few related fields from which well-known optimization techniques can be drawn.

Key Contributions

Our goal in this thesis is to develop a novel approach to description logic reasoning which addresses the above problems with traditional tableau methods and allows for the implementation of reasoners that exhibit superior performance on the types of ontologies encountered in practice. To this end, we present a number of novel contributions:

- A new *hypertableau* reasoning calculus which reduces or-branching, generalizing and extending many absorption-style optimizations. This new calculus is able to process Horn knowledge bases entirely deterministically, and typically exhibits

very little nondeterminism on Horn knowledge bases extended with a small number of non-Horn axioms. This calculus also exhibits a number of appealing implementation properties and is particularly amenable to optimization. [Motik *et al.*, 2007]

- The application of a new blocking strategy (*pairwise anywhere blocking*) which reduces and-branching and reduces the size of generated models. This blocking strategy is applicable to traditional tableau algorithms as well as our new hypertableau calculus. [Motik *et al.*, 2007; Motik *et al.*, 2009]

- A novel *NI-rule* which handles the combination of nominals, inverse roles, and number restrictions far more efficiently than standard tableau rules. This rule can also be adapted to traditional tableau algorithms. [Motik *et al.*, 2008]

- A model-caching technique which re-uses work from one consistency test in order to speed tests for similar knowledge bases. [Shearer *et al.*, 2008; Motik *et al.*, 2009]

- A new general-purpose classification algorithm which reduces the number of individual comparisons necessary in order to compute a quasi-ordering of elements, and also allows for maximal exploitation of partial prior knowledge of the quasi-ordering. [Shearer and Horrocks, 2009; Shearer *et al.*, 2009]

- A novel technique for extracting subsumption and non-subsumption information from Tarski-style models, as generated by tableau and hypertableau calculi. On realistic test data, this approach extracts the vast majority of (non)subsumption information from concept satisfiability tests, eliminating the need for individual subsumption tests. [Shearer and Horrocks, 2009; Shearer *et al.*, 2009]

In order to evaluate the efficacy of our techniques, we have implemented them in a prototypical reasoner called HermiT [Shearer *et al.*, 2008]. This implementation

demonstrates substantial performance improvements over traditional tableau reasoners in our tests.

Chapter Guide

In the remainder of Part I, we formally introduce Description Logics, as well as the various DL languages and reasoning problems addressed in the body of this thesis.

Part II presents a new algorithm for searching for a model of a knowledge base in the description logic $\mathcal{SROIQ}$. Chapter 3 summarizes a few of the difficulties inherent in developing such an algorithm and summarizes a few of our techniques for overcoming them. Chapter 4 provides an informal overview of our algorithm, and Chapter 5 describes our new hypertableau calculus and blocking strategy in detail. Chapter 6 addresses implementation and optimization issues for this new calculus, including its efficient application to less expressive logics as well as worst-case performance characteristics. Chapter 7 compares our calculus to other reasoning techniques.

Part III addresses the efficient implementation of classification-based services for description logics. After describing a few of the problems that arise in classification in Chapter 8, we present our new general-purpose algorithm for deducing a quasi-ordering in Chapter 9. Chapter 10 describes techniques for extracting partial information about the subsumption quasi-ordering between concept names in DL knowledge bases from Tarski-style models, such as those generated by the algorithm presented in Part II. Related work in the field of classification is reviewed in Chapter 11.

Finally, Part IV discusses the impact of our contributions. Chapter 12 evaluates the performance of a prototypical implementation of our techniques and compares it with several other widely-used description logic reasoners, and Chapter 13 summarizes our results and their implications for future research in the field.

Chapter 2

Description Logics

Description Logics (DLs) are a family of knowledge representation formalisms with formally-defined syntax and semantics. Most commonly-used description logics are decidable fragments of first-order logic, restricted to only unary and binary predicates and carefully bounded forms of existential quantification. DLs are closely related to modal logics [Andrka *et al.*, 1998] and trace their history to attempts at standardizing semantics for human-centered knowledge representations such as semantic networks and frames [Lehmann, 1992].

Intuitively, DLs are used to describe a domain of *individuals* and a set of binary relations, called *roles*, between those individuals. These descriptions are constructed by defining sets of individuals with particular semantic characteristics; such sets are called *concepts*. The expressiveness of a particular DL is primarily a product of the richness of the language provided for defining concepts. Most expressive DLs further allow the imposition of simple global semantic restrictions (called *axioms*) on concepts and roles (e.g. that one concept or role is subsumed by another), as well as *assertions* about the concept memberships of, and role relationships between, specific individuals.

This thesis is concerned with efficiently identifying the implicit semantic consequences arising from combinations of concept definitions, axioms, and assertions. In Part II we present an algorithm that can be used to detect implicit contradictions

in such knowledge bases, and in Part III we show how the same algorithm can be adapted to identify implicit subsumption relationships between concepts.

In this chapter, we formally introduce Description Logics as well as the reasoning problems addressed by the remainder of the thesis.

2.1 Syntax and Semantics

We now define the syntax and the semantics of the description logic $\mathcal{SROIQ}$.

Definition 1 (Concepts, Roles, Individuals, and Interpretations) A *signature* is a triple $\Sigma = (N_R, N_C, N_I)$ consisting of mutually disjoint sets of *atomic roles* N_R, *atomic concepts* N_C, and *individuals* N_I. We additionally distinguish a subset $N_{sR} \subseteq N_R$ of *simple atomic roles*.

The set of *roles* over Σ is $N_R \cup \{R^- \mid R \in N_R\}$; roles of the form R^- are *inverse roles*. The function $\mathsf{inv}(\cdot)$ is defined on the set of roles as follows, where R is an atomic role: $\mathsf{inv}(R) = R^-$ and $\mathsf{inv}(R^-) = R$. A role R is *simple* iff either R or $\mathsf{inv}(R)$ is a simple atomic role.

We define the set of *concepts* over Σ inductively. Let R be a role, S a simple role, a an individual, A an atomic concept, n a positive integer, and C and D concepts. Then a concept is of one of the following forms:

$\top$	(the *top concept*)	$\bot$	(the *bottom concept*)
A	(an atomic concept)	$\neg C$	(a *negation*)
$C \sqcap D$	(a *conjunction*)	$C \sqcup D$	(a *disjunction*)
$\{a\}$	(a *nominal*)	$\exists S.\mathsf{Self}$	(a *local reflexivity restriction*)
$\exists R.C$	(an *existential restriction*)	$\forall R.C$	(a *universal restriction*)
$\geq n\, S.C$	(an *at-least restriction*)	$\leq n\, S.C$	(an *at-most restriction*)

At-least and at-most restrictions are types of *number restrictions*.

An *interpretation* over Σ is a tuple $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a nonempty set (the *domain* of $\mathcal{I}$) and $\cdot^\mathcal{I}$ is an *interpretation function* which assigns an element $a^\mathcal{I} \in \Delta^\mathcal{I}$ to each individual a, a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ to each atomic concept A, and a relation $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ to each atomic role R. The interpretation function $\cdot^\mathcal{I}$ is extended to concepts and roles as follows:

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \qquad \bot^{\mathcal{I}} = \emptyset$$

$$(R^-)^{\mathcal{I}} = \{\langle y, x\rangle \mid \langle x, y\rangle \in R^{\mathcal{I}}\} \qquad (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$$

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$

$$\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\} \qquad (\exists S.\mathsf{Self})^{\mathcal{I}} = \{x \mid \langle x, x\rangle \in S^{\mathcal{I}}\}$$

$$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y : \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$

$$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y : \langle x, y\rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$$

$$(\geq n\, S.C)^{\mathcal{I}} = \{x \mid \sharp\{y \mid \langle x, y\rangle \in S^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n\}$$

$$(\leq n\, S.C)^{\mathcal{I}} = \{x \mid \sharp\{y \mid \langle x, y\rangle \in S^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}$$

Two interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and $\mathcal{I}' = (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ *coincide on* a signature Σ iff $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$ and $x^{\mathcal{I}} = x^{\mathcal{I}'}$ for every $x \in \Sigma$.

A concept is in *negation-normal form* if negation appears only in front of atomic concepts, nominals, and local reflexivity restrictions. For any concept C, there exists a concept $\mathsf{nnf}(C)$ in negation-normal form such that $C^{\mathcal{I}} = \mathsf{nnf}(C)^{\mathcal{I}}$ for any interpretation $\mathcal{I}$. The concept $\mathsf{nnf}(C)$ can be computed from C in time linear in the size of C by pushing negation inward using the following dualities:

$$\neg\neg C \leftrightarrow C \qquad \neg(C \sqcap D) \leftrightarrow \neg C \sqcup \neg D \qquad \neg(C \sqcup D) \leftrightarrow \neg C \sqcap \neg D$$

$$\neg\exists R.C \leftrightarrow \forall R.\neg C \qquad \neg(\forall R.C) \leftrightarrow \exists R.\neg C \qquad \neg(\leq n\, S.C) \leftrightarrow \geq (n+1)\, S.C$$

$$\neg(\geq 1\, S.C) \leftrightarrow \forall S.\neg C \qquad \neg(\geq (n+1)\, S.C) \leftrightarrow \leq n\, S.C$$

We use $\dot{\neg}C$ to denote $\mathsf{nnf}(\neg C)$. △

Definition 2 (TBox, ABox, and RBox) A *TBox* $\mathcal{T}$ is a finite set of *general concept inclusions* (GCIs) $C \sqsubseteq D$ for C and D concepts. An interpretation $\mathcal{I}$ *satisfies* a GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it satisfies a TBox $\mathcal{T}$ iff it satisfies all GCIs of $\mathcal{T}$.

An *ABox* $\mathcal{A}$ is a finite set of assertions of the form $C(a)$ (*concept assertion*), $R(a,b)$ (*role assertion*), $a \approx b$ (*equality assertion*), or $a \not\approx b$ (*inequality assertion*), where C is a concept, R is a role, and a and b are individuals. An interpretation $\mathcal{I}$ satisfies a concept assertion $C(a)$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$, a role assertion $R(a,b)$ iff $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$, an equality $a \approx b$ iff $a^{\mathcal{I}} = b^{\mathcal{I}}$, and an inequality $a \not\approx b$ iff $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. An interpretation satisfies an ABox $\mathcal{A}$ iff it satisfies all assertions of $\mathcal{A}$.

An *RBox* $\mathcal{R}$ is a finite set of axioms of the form $\mathsf{Dis}(S_1, S_2)$ (*role disjointness*), $\mathsf{Ref}(R)$ (*reflexivity*), $\mathsf{Irr}(S_1)$ (*irreflexivity*), $S_1 \sqsubseteq S_2$ (*simple role inclusion*), or $\omega \sqsubseteq U$ (*complex role inclusion*), where S_1 and S_2 are simple roles, R is any role, ω is a finite string $R_1 ... R_n$ of roles, and U is a role which is not simple.

We use $\mathsf{Sym}(R)$ as a shorthand for $R^- \sqsubseteq R$ (*symmetry*) and $\mathsf{Tra}(U)$ as a shorthand for $UU \sqsubseteq U$ (*transitivity*).

An interpretation $\mathcal{I}$ satisfies a role disjointness axiom $\mathsf{Dis}(S_1, S_2)$ iff $S_1^{\mathcal{I}} \cap S_2^{\mathcal{I}} = \emptyset$, a reflexivity axiom $\mathsf{Ref}(R)$ iff $\langle a, a \rangle \in R^{\mathcal{I}}$ for all $a \in \Delta^{\mathcal{I}}$, a irreflexivity axiom $\mathsf{Irr}(S)$ iff $\langle a, a \rangle \notin S^{\mathcal{I}}$ for all $a \in \Delta^{\mathcal{I}}$, and a simple role inclusion $S_1 \sqsubseteq S_2$ iff $S_1^{\mathcal{I}} \subseteq S_2^{\mathcal{I}}$. An interpretation $\mathcal{I}$ satisfies a role inclusion axiom $R_1 ... R_n \sqsubseteq U$ iff $R_1^{\mathcal{I}} \circ ... \circ R_2^{\mathcal{I}} \subseteq U^{\mathcal{I}}$, where $\circ$ stands for the composition of binary relations. An interpretation satisfies an RBox $\mathcal{R}$ iff it satisfies all axioms of $\mathcal{R}$. $\triangle$

Definition 3 (Regular RBoxes) A *strict partial order* $\prec$ on a set A is an irreflexive and transitive relation on A. A strict partial order $\prec$ on the set of roles $N_R \cup \{R^- \mid R \in N_R\}$ is a *regular order* iff $R_1 \prec R_2$ whenever $\mathsf{inv}(R_1) \prec R_2$ for all R_1 and R_2. An RBox $\mathcal{R}$ is *regular* if there exists a regular order $\prec$ such that for every simple role inclusion axiom $S_1 \sqsubseteq S_2$ in $\mathcal{R}$ either $S_1 = \mathsf{inv}(S_2)$ or $S_1 \prec S_2$, and for every complex role inclusion axiom $\omega \sqsubseteq U$ in $\mathcal{R}$:

1. $\omega = UU$, or

2. $\omega = \mathsf{inv}(U)$, or

3. $\omega = R_1...R_n$ and $R_i \prec U$ for all $1 \leq i \leq n$, or

4. $\omega = UR_1...R_n$ and $R_i \prec U$ for all $1 \leq i \leq n$, or

5. $\omega = R_1...R_nU$ and $R_i \prec U$ for all $1 \leq i \leq n$. $\triangle$

Definition 4 (Logics, knowledge bases, and inference problems) A $\mathcal{SROIQ}$ *knowledge base* $\mathcal{K}$ is a a triple $(\mathcal{T}, \mathcal{A}, \mathcal{R})$, where $\mathcal{T}$ is a TBox, $\mathcal{A}$ is an ABox, and $\mathcal{R}$ is a regular RBox. With $|\mathcal{K}|$ we denote the size of $\mathcal{K}$—that is, the number of symbols required to encode $\mathcal{K}$ on the input tape of a Turing machine (numbers can be coded in binary). [Kutz *et al.*, 2006]

An interpretation $\mathcal{I}$ is a *model* of $\mathcal{K}$, written $\mathcal{I} \models \mathcal{K}$, if it satisfies the TBox, ABox, and RBox of $\mathcal{K}$. The basic inference problem for $\mathcal{SROIQ}$ that we address is checking whether $\mathcal{K}$ is *satisfiable*—that is, checking whether a model of $\mathcal{K}$ exists. A concept C *subsumes* a concept D, written $\mathcal{K} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for each model $\mathcal{I}$ of $\mathcal{K}$. It is easy to see that $\mathcal{K} \models C \sqsubseteq D$ if and only if $\mathcal{K} \cup \{(C \sqcap \neg D)(a)\}$ is unsatisfiable, where a is an individual that does not occur in $\mathcal{K}$ [Baader and Nutt, 2007].

An $\mathcal{ALCHOIQ}^+$ knowledge base is a $\mathcal{SROIQ}$ KB which contains no complex role inclusion axioms. In the absence of complex role inclusion axioms, there is no distinction between simple and non-simple roles. Hence, in the context of $\mathcal{ALCHOIQ}^+$ we assume that all roles are simple unless otherwise stated and, without loss of generality, we treat $\exists R.B$ as a syntactic shortcut for $\geq 1\, R.B$.

Analogously, the logic $\mathcal{SRIQ}$ is obtained from $\mathcal{SROIQ}$ by disallowing nominals, $\mathcal{SROQ}$ is obtained from $\mathcal{SROIQ}$ by disallowing inverse roles, and $\mathcal{SHOIQ}$, $\mathcal{SHIQ}$, and $\mathcal{SHOQ}$ are obtained from $\mathcal{SROIQ}$, $\mathcal{SRIQ}$, and $\mathcal{SROQ}$, respectively, by restricting the RBox to only simple role inclusion axioms and complex role inclusions of the form $UU \sqsubseteq U$ (transitivity). Finally, $\mathcal{SHOI}$ is obtained from $\mathcal{SHOIQ}$ by disallowing number restrictions. $\triangle$

Horn logic is a fragment of first-order logic in which formulae are restricted to clauses containing at most one positive literal. While Horn logic cannot represent disjunctive information, practical query answering procedures for Horn knowledge bases are known. We next define a subset of DL knowledge bases which can be encoded in Horn logic; the exact encoding is given in Chapter 5.

Definition 5 (Horn-DL) Let C^+, C^-, and $C^\bullet$ be three classes of concepts whose grammar is defined as follows, where A is an atomic concept, R is a role, and n is a positive integer:

$$C^+ ::= \top \mid \bot \mid A \mid C^+ \sqcap C^+ \mid C^+ \sqcup C^+ \mid \exists R.C^+ \mid \exists R.\mathsf{Self}$$

$$C^- ::= \top \mid \bot \mid \neg A \mid C^- \sqcap C^- \mid C^- \sqcup C^- \mid \forall R.C^-$$

$$C^\bullet ::= \top \mid \bot \mid A \mid \neg A \mid C^\bullet \sqcap C^\bullet \mid C^- \sqcup C^\bullet$$
$$\mid \exists R.C^\bullet \mid \forall S.C^\bullet \mid \forall R.C^+ \mid \geq n\, R.C^\bullet \mid \leq 1\, R.C^-$$

The intuition behind these three classes is that any concept C^+ can be encoded in first-order clauses containing only positive literals, any concept C^- can be encoded in clauses containing only negative literals, and any concept $C^\bullet$ can be encoded in clauses which each contain at most one positive literal.

A DL knowledge base $\mathcal{K}$ is *Horn* if all GCIs of $\mathcal{K}$ are of the form $\top \sqsubseteq C^\bullet$. △

Part II

Model Generation

Chapter 3

Difficulties

In this chapter, we review the traditional tableau approach to demonstrating satisfiability of a DL knowledge base $\mathcal{K}$ by constructing (an abstraction of) a model of $\mathcal{K}$. We highlight a few of the main difficulties encountered in such model constructions and discuss major sources of inefficiency.

While we defer a formal description of our new hypertableau reasoning calculus to Chapter 5, this chapter summarizes the key differences between our approach and traditional tableau methods.

3.1 Traditional Tableau Algorithms

To show that a knowledge base $\mathcal{K} = (\mathcal{R}, \mathcal{T}, \mathcal{A})$ is satisfiable, a tableau algorithm constructs a *derivation*—a sequence of ABoxes $\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n$ where $\mathcal{A}_0 = \mathcal{A}$ and each $\mathcal{A}_i$ is obtained from $\mathcal{A}_{i-1}$ by an application of one *derivation rule*.[1] The derivation rules make the information implicit in the axioms of $\mathcal{R}$ and $\mathcal{T}$ explicit, and thus evolve the ABox $\mathcal{A}$ towards a (representation of a) model of $\mathcal{K}$. The algorithm terminates either if no derivation rule is applicable to some $\mathcal{A}_n$, in which case $\mathcal{A}_n$ represents a model of $\mathcal{K}$, or if $\mathcal{A}_n$ contains an obvious contradiction, in which case the model construction has failed. The following derivation rules are commonly used in DL tableau calculi.

[1] Some formalizations of tableau algorithms work on *completion graphs* [Horrocks and Sattler, 2007], which have a natural correspondence to ABoxes.

- $\sqcup$-rule: Given $(C_1 \sqcup C_2)(s)$, derive either $C_1(s)$ or $C_2(s)$.
- $\sqcap$-rule: Given $(C_1 \sqcap C_2)(s)$, derive $C_1(s)$ and $C_2(s)$.
- $\exists$-rule: Given $(\exists R.C)(s)$, derive $R(s,t)$ and $C(t)$ for t a fresh individual.
- $\forall$-rule: Given $(\forall R.C)(s)$ and $R(s,t)$, derive $C(t)$.
- $\sqsubseteq$-rule: Given a GCI $C \sqsubseteq D$ and an individual s, derive $(\neg C \sqcup D)(s)$.

Such a ruleset exhibits two types of nondeterminism. First, when multiple different rule applications are possible an implementation of this calculus must choose which to apply at each step. With the exception of occasional precedence rules which must be respected, termination or the detection of contradictions is independent of such choices in tableau calculi; we thus call such choices *don't-care nondeterminism*.

A second type of nondeterminism is introduced by the $\sqcup$-rule. If $(C_1 \sqcup C_2)(s)$ is true, then $C_1(s)$ or $C_2(s)$ or both are true. Therefore, implementations of tableau calculi make a nondeterministic guess and choose either C_1 or C_2. If one choice leads to a contradiction, the implementation must try the other choice. Thus, $\mathcal{K}$ is unsatisfiable only if exhaustive search of all combinations of choices lead to a contradiction; in practice this is implemented by means of backtracking search. We call this *don't-know nondeterminism*.

We next discuss two sources of complexity inherent in standard tableau derivation rules.

3.2 Or-Branching

Handing disjunctions through reasoning by case is often called *or-branching*. The $\sqsubseteq$-rule adds a disjunction for each GCI to each individual in an ABox and is thus a major source of or-branching and inefficiency [Horrocks, 2007]. Consider, for example, the

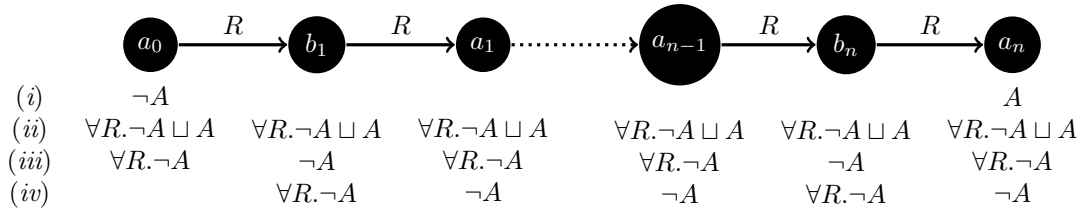


Figure 3.1: Or-Branching Example

knowledge base $\mathcal{K}_1 = (\mathcal{T}_1, \mathcal{A}_1, \emptyset)$, with $\mathcal{T}_1$ and $\mathcal{A}_1$ specified as follows:

$$\begin{aligned} \mathcal{T}_1 &= \{\exists R.A \sqsubseteq A\} \\ \mathcal{A}_1 &= \{\neg A(a_0),\ R(a_0, b_1),\ R(b_1, a_1), \ldots,\ R(a_{n-1}, b_n),\ R(b_n, a_n),\ A(a_n)\} \end{aligned} \tag{3.1}$$

The ABox $\mathcal{A}_1$ is graphically shown in Figure 3.1. The individuals occurring in the ABox are represented as black dots, an assertion of the form $A(a_0)$ is represented by placing A next to the individual a_0, and an assertion of the form $R(a_0, b_1)$ is represented as an R-labeled arrow from a_0 to b_1. Initially, $\mathcal{A}_1$ contains only the concept assertions shown in line (i).

To satisfy the GCI in $\mathcal{T}_1$, a tableau algorithm applies the $\sqsubseteq$-rule, thus adding the assertions shown in line (ii) of Figure 3.1. Tableau algorithms are usually free to choose the order in which they process the assertions in an ABox; in fact, finding an order that exhibits good performance in practice requires advanced heuristics [Tsarkov and Horrocks, 2005b]. Let us assume that the algorithm chooses to process the assertions on a_i before those on b_j. Hence, by applying the derivation rules to all a_i, a tableau algorithm derives the assertions shown in line (iii) of Figure 3.1; after that, by applying the derivation rules to all b_i, the algorithm derives the assertions shown in line (iv) of Figure 3.1. The ABox now contains both $A(a_n)$ and $\neg A(a_n)$, which is a contradiction. Thus, the algorithm needs to backtrack its most recent choice, so it flips its guess on b_{n-1} to $A(b_{n-1})$. This generates a contradiction on b_{n-1}, so the algorithm backtracks from all guesses for b_i, changes the guess on a_n to $A(a_n)$, and repeats the work for all b_i. This also leads to a contradiction, so the algorithm must revise its guess for a_{n-1}; but then, two guesses are again possible for

a_n. In general, after revising a guess for a_i, all possibilities for a_j, $i < j \leq n$, must be reexamined, which results in exponential behavior. None of the standard backtracking optimizations [Horrocks, 2007] are helpful: the problem arises because the order in which the individuals are processed makes the guesses on a_i independent from the guesses on a_j for $i \neq j$.

The GCI $\exists R.A \sqsubseteq A$, however, is not inherently nondeterministic: it is equivalent to the Horn clause $\forall x, y : [R(x, y) \wedge A(y) \rightarrow A(x)]$, which can be applied bottom-up to derive the assertions $A(b_n), A(a_{n-1}), \ldots, A(a_0)$ and eventually reveal a contradiction on a_0. These inferences are deterministic,[2] so we can conclude that $\mathcal{K}_1$ is unsatisfiable without any backtracking. This example suggests that the processing of GCIs in tableau algorithms can be "unnecessarily" nondeterministic. Hustadt *et al.* [2005] have identified a class of knowledge bases without "unnecessary" nondeterminism: $\mathcal{SHIQ}$ knowledge bases which are Horn can always be translated into Horn clauses, suggesting that reasoning without any nondeterminism is possible in principle. Ideally, a practical DL reasoning procedure should exhibit no nondeterminism on Horn knowledge bases.

In the context of tableau calculi, various *absorption* optimizations [Horrocks, 2007] have been developed to control the nondeterminism arising in the application of GCIs. These optimizations are closely related to Horn transformations, but they do not eliminate all nondeterminism from Horn KBs. We discuss these optimizations in depth in Section 7.1.

We address the problem of or-branching in our calculus by first encoding complex concepts and GCIs as *DL-clauses*—universally quantified implications containing DL concepts and roles as predicates—and then replacing the tableau $\sqcap$-, $\sqcup$-, and $\sqsubseteq$-rules with a *hyperresolution rule* (*Hyp*-rule) that matches the antecedents of clauses to the ABox, deriving the clauses' consequents. This approach allows reasoning to proceed

[2] More precisely, each inference is deterministic, but the order in which the inferences are performed is don't-care nondeterministic.

(a) Ancestor Blocking (b) Anywhere Blocking

Figure 3.2: And-Branching Example

with far less nondeterminism in many cases, and entirely deterministically in the case of Horn KBs. We describe DL-clauses and the *Hyp*-rule informally in Section 4.1, and provide full definitions in Chapter 5.

3.3 And-Branching

The introduction of new individuals in the $\exists$-rule is often called *and-branching*, and it is another major source of inefficiency in tableau algorithms [Donini, 2007]. Consider, for example, the (satisfiable) knowledge base $\mathcal{K}_2 = (\mathcal{T}_2, \mathcal{A}_2, \emptyset)$, with $\mathcal{T}_2$ and $\mathcal{A}_2$ specified as follows (where n and m are integers):

$$\begin{aligned} \mathcal{T}_2 &= \{ \quad A_1 \sqsubseteq \geq 2\, S.A_2,\ \ldots,\ A_{n-1} \sqsubseteq \geq 2\, S.A_n,\ A_n \sqsubseteq A_1, \\ &\qquad A_i \sqsubseteq (B_1 \sqcup C_1) \sqcap \ldots \sqcap (B_m \sqcup C_m) \text{ for } 1 \leq i \leq n \ \} \\ \mathcal{A}_2 &= \{ \quad A_1(a) \ \} \end{aligned} \tag{3.2}$$

At-least restrictions are dealt with in tableau algorithms by the $\geq$-rule, which is quite similar to the $\exists$-rule: from $(\geq n\, R.C)(s)$, the $\geq$-rule derives $R(s, t_i)$ and $C(t_i)$ for $1 \leq i \leq n$, and $t_i \not\approx t_j$ for $1 \leq i < j \leq n$. Thus, the assertion $A_1(a)$ implies the existence of at least two individuals in A_2, which imply the existence of at least two individuals in A_3, and so on. Given $\mathcal{K}_2$, a tableau algorithm thus constructs a binary tree, shown in Figure 3.2a, in which each individual is labeled with some A_i and an element of $\Pi = \{B_1, C_1\} \times \ldots \times \{B_m, C_m\}$. All individuals in the tree at depth n are instances of A_n; because of the GCI $A_n \sqsubseteq A_1$, these individuals must be instances

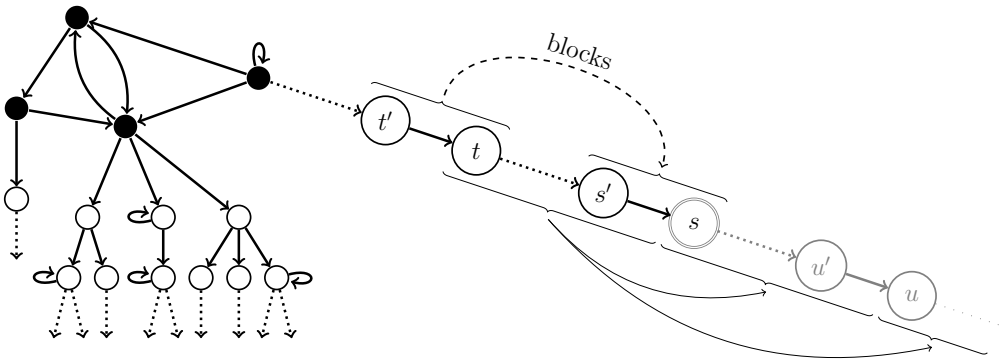


Figure 3.3: Forest-Like Shape of ABoxes

of A_1 as well, so we can repeat the whole construction and generate an even deeper tree. Clearly, a naïve application of the tableau rules does not terminate if the TBox contains existential quantifiers in cycles.

To ensure termination in such cases, tableau algorithms employ *blocking* [Buchheit *et al.*, 1993; Baader and Nutt, 2007], which is based on an important observation about the shape of ABoxes that can be derived from some input ABox $\mathcal{A}$. The individuals in $\mathcal{A}$ are called *named* (shown as black circles), and they can be connected by role assertions in an arbitrary way. The individuals introduced by the $\exists$- and $\geq$-rules are called *blockable* (shown as white circles). For example, if $\exists R.C(a)$ is expanded into $R(a, s)$ and $C(s)$, then s is called a blockable individual and it is an *R-successor* of a. It is not difficult to see that, if the knowledge base does not contain nominals, no tableau derivation rule can connect s with an arbitrary named individual: the individual s can participate only in inferences that derive an assertion of the form $D(s)$ with D a concept, create a new successor of s, connect s to an existing predecessor or successor, or, in the presence of (local) reflexivity, connect s to itself. Hence, each ABox $\mathcal{A}'$ obtained from $\mathcal{A}$ can be seen as a "forest" of the form shown in Figure 3.3: each named individual can be arbitrarily connected to other named individuals and to a tree of blockable successors. The *concept label* $\mathcal{L}_{\mathcal{A}}(s)$ is defined as the set of all concepts C such that $C(s) \in \mathcal{A}$, and the *edge label* $\mathcal{L}_{\mathcal{A}}(s, s')$ as the set of all atomic roles such that $R(s, s') \in \mathcal{A}$.

The forest-like structure of ABoxes enables blocking. Description logics such as $\mathcal{SHIQ}$ and $\mathcal{SROIQ}$ allow for the combination of inverse roles and number restrictions, which has been handled in the literature by *ancestor pairwise blocking* [Horrocks *et al.*, 2000b]: for individuals s, s', t, and t' occurring in an ABox $\mathcal{A}$ as shown in Figure 3.3, t blocks s (shown by a double border on s) if and only if $\mathcal{L}_{\mathcal{A}}(s) = \mathcal{L}_{\mathcal{A}}(t)$, $\mathcal{L}_{\mathcal{A}}(s') = \mathcal{L}_{\mathcal{A}}(t')$, $\mathcal{L}_{\mathcal{A}}(s, s') = \mathcal{L}_{\mathcal{A}}(t, t')$, and $\mathcal{L}_{\mathcal{A}}(s', s) = \mathcal{L}_{\mathcal{A}}(t', t)$.[3] In tableau algorithms, the $\exists$- and $\geq$-rules are applicable only to nonblocked individuals, which ensures termination: the number of different concept and edge labels is exponential in $|\mathcal{K}|$, so an exponentially long branch in a forest-like ABox must contain a blocked individual, thus limiting the length of each branch in an ABox. Let $\mathcal{A}$ be an ABox as in Figure 3.3 to which no tableau derivation rule is applicable, and in which s is blocked by t. We can construct a model from $\mathcal{A}$ by *unraveling*—that is, by replicating the fragment between s and t infinitely often. Intuitively, blocking ensures that the part of the ABox between s and s' "behaves" just like the part between t and t', so unraveling indeed generates a model. If our logic were able to connect blockable individuals in a non-tree-like way, then unraveling would not generate a model; in fact, the notion of ancestors, descendants, and blocking would itself be ill-defined.

Consider now an "unlucky" run of a tableau algorithm with ancestor pairwise blocking on $\mathcal{K}_2$. The number of elements in Π is exponential in $|\mathcal{K}_2|$, so it can happen that blocking comes into effect only after the algorithm constructs an exponentially deep tree; since the tree is binary, it is doubly exponential in total. In a "lucky" run, the algorithm can always pick B_j instead of C_j; then, the algorithm constructs a polynomially deep binary tree, so the tree is exponential in total. Thus, the and-branching caused by the $\exists$- and $\geq$-rules can lead to unnecessary generation of an ABox that is doubly exponential in the size of the input, which limits the scalability of tableau algorithms in practice.

[3] Our blocking definition must include both edge labels in both directions because, unlike in some other tableau formalizations, our edge labels include only atomic roles.

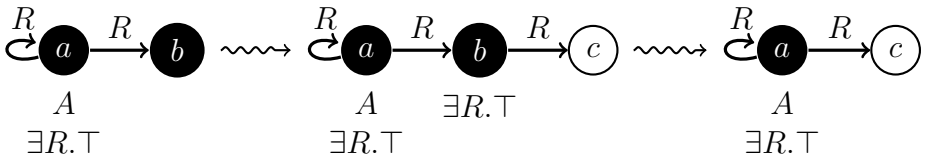


Figure 3.4: A Yo-Yo Example

We address the problem of and-branching with a more aggressive blocking strategy which limits the total size of a tree in the generated model to $|\Pi|$ by extending the set of potential blockers for an individual to non-ancestors, illustrated in Figure 3.2b. We discuss this *anywhere pairwise blocking* technique in Section 4.3.

3.4 Problems Due to Merging

If a logic allows number restrictions or (certain types of) equalities, then the reasoning calculus must be extended with an additional $\approx$-rule which *merges* the two individuals a and b in the presence of $a \approx b$. Merging can easily lead to termination problems even for very simple DLs, as shown in the following example. For simplicity, we present the TBox of $\mathcal{K}_3$ as a set of DL-clauses $\mathcal{C}_3$.

$$\begin{array}{rcl} \mathcal{A}_3 & = & \{\ A(a), \quad \exists R.\top(a), \quad R(a,b), \quad R(a,a)\ \} \\ \mathcal{C}_3 & = & \{\ R(x,y_1) \wedge R(x,y_2) \rightarrow y_1 \approx y_2, \qquad A(x) \wedge R(x,y) \rightarrow \exists R.\top(y)\ \} \end{array} \tag{3.3}$$

Consider now a derivation on $\mathcal{A}_3$ and $\mathcal{C}_3$, illustrated in Figure 3.4: from the second clause, our *Hyp*-rule derives $\exists R.\top(b)$, which the $\exists$-rule expands to $R(b,c)$; then, by the first clause, we derive $b \approx a$, so the $\approx$-rule merges b into a. Clearly, the resulting ABox is isomorphic to the original one (that c is a blockable and b a named individual is not relevant here), so we can repeat the same sequence of inferences, which leads to nontermination. To the best of our knowledge, this problem was first identified by Baader and Sattler [2001], and it is commonly known as a "yo-yo."

This problem arises because, due to merging, a can have an unbounded number of blockable R-successors: the blockable individual c is created as an R-successor of

b, but merging b into a makes c a blockable R-successor of a. This, in turn, allows us to apply the clauses from $\mathcal{C}_3$ to a an arbitrary number of times, which leads to nontermination.

This problem can be solved by always merging a descendant s into its ancestor t, and *pruning* s before merging—that is, by removing all assertions containing a blockable descendant of s and thus ensuring that t does not "inherit" new successors.[4] Pruning is formally defined in Definition 13 on page 60.

Thus, before merging b into a in our example, we prune b—that is, we remove the assertion $R(b, c)$. Merging then produces an ABox that represents a model of $\mathcal{A}_3$ and $\mathcal{C}_3$, so the algorithm terminates. Note that pruning is well-defined only because our ABoxes are forest-shaped, cf. Figure 3.3: if connections between individuals were arbitrary and, in particular, cyclic, it would not be clear which part of the ABox should be pruned.

3.5 Nominals, Inverses, and Number Restrictions

With nominals, it is possible to derive ABoxes that are not forest-like, as the following simple example demonstrates. For presentation purposes, we use the concept $\exists R.\{c\}$ in the DL-clauses even though such concepts would be further decomposed in our algorithm.

$$\begin{array}{rcl} \mathcal{A}_4 & = & \{\ A(a), \quad A(b)\ \} \\ \mathcal{C}_4 & = & \{\ A(x) \to (\exists R.B)(x),\ \ B(x) \to (\exists R.C)(x),\ \ C(x) \to (\exists S.\{c\})(x)\ \} \end{array} \tag{3.4}$$

Successive applications of the *Hyp*- and $\exists$-rules to $\mathcal{A}_4$ and $\mathcal{C}_4$ can produce the ABox $\mathcal{A}_4'$ shown on the left-hand side of Figure 3.5. This ABox is clearly not forest-shaped: the two paths of role atoms in $\mathcal{A}_4'$ start at the named individuals a and b and end in a named individual c. Nevertheless, if role relations between blockable individuals remain forest-like, termination of the derivation can be ensured using blocking. Some

[4] Horrocks *et al.* [2000b] do not physically remove successors, but mark them as "not present" by setting the relevant edge labels to $\emptyset$. This has exactly the same effect as pruning.

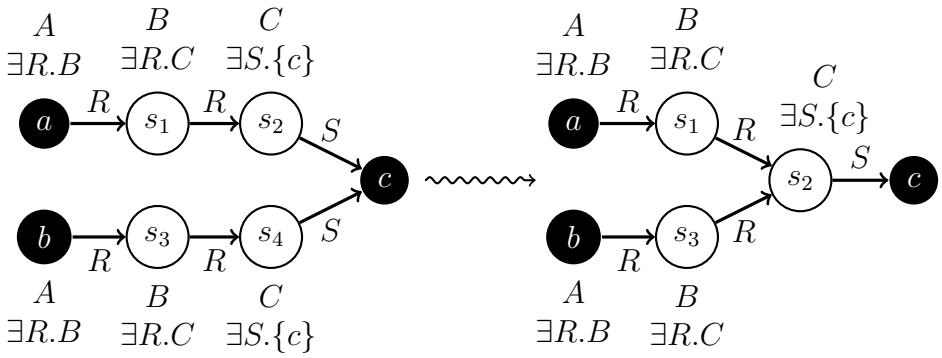


Figure 3.5: Non-Tree-Like Structures Due to Merging

DLs that include nominals produce only such *extended forest-like* ABoxes [Horrocks and Sattler, 2001].

If a DL includes inverse roles, number restrictions, and nominals, the shape of an ABox becomes much more involved. To this end, assume now that we extend $\mathcal{C}_4$ with the DL-clause $S(y_1, x) \wedge S(y_2, x) \rightarrow y_1 \approx y_2$ (which axiomatizes S to be inverse-functional and effectively introduces number restrictions). On $\mathcal{A}_4'$, the *Hyp*-rule then derives $s_2 \approx s_4$. Note that both s_2 and s_4 are blockable individuals; furthermore, neither individual is an ancestor of the other, so we can merge, say, s_4 into s_2. This produces the ABox $\mathcal{A}_4''$ shown on the right-hand side of Figure 3.5, in which the assertion $R(s_3, s_2)$ makes $\mathcal{A}_4''$ not forest-shaped. By extending the example, it is possible to use nominals, inverse roles, and number restrictions to arrange blockable individuals in cycles. The derived ABoxes are thus not forest-shaped, which makes defining suitable notions of pruning and unraveling difficult and prevents us from using blocking to ensure termination of the calculus.

3.5.1 Promoting Blockable Individuals

To solve this problem, we need to extend the arbitrarily interconnected part of $\mathcal{A}_4''$ by changing the status of s_2 from a blockable into a *root individual*—that is, an individual similar to the named ones in that it can be arbitrarily interconnected. Our

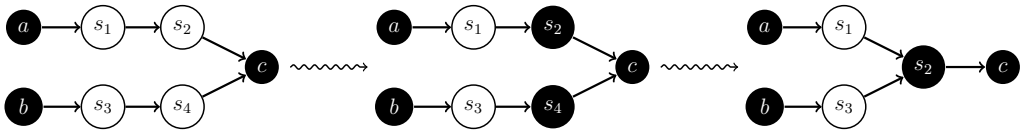


Figure 3.6: The Introduction of Root Individuals

extended forest-like ABoxes thus consist of a set of arbitrarily interconnected root individuals each of which can be the root of a "tree" (ignoring reflexive connections and connections back to root individuals) that otherwise consists entirely of blockable individuals (see Figure 3.3 on page 21). Named individuals are just the subset of the root individuals that occur in the input ABox. When we talk about individuals, we mean either root or blockable ones (see Definition 13 on page 60 for a formal definition).

Returning to our example, after changing the status of s_2 from a blockable into a root individual, only s_1 and s_3 are blockable in $\mathcal{A}_4''$, so the ABox has the extended forest-like shape and we can apply blocking and pruning as usual. This is schematically shown in Figure 3.6. More generally, we apply the following preliminary version of a *Nominal Introduction rule* (*NI*-rule), which we denote with (*) for easier reference:

> We change s into a root individual whenever $\mathcal{A}$ contains assertions $R(s, a)$ and $A(s)$ where a is a root or a named individual, s is a blockable individual that is not a successor of a, and a must satisfy an at-most restriction $\leq n\, R^-.A$.

Note that, if s is a successor of a, then the part of the ABox involving s and a is forest-shaped, so the *NI*-rule need not be applicable.

This solution, however, introduces another problem: the number of root individuals can now grow arbitrarily, as shown in the following example.

$$\begin{array}{rcl} \mathcal{A}_5 & = & \{\ A(b)\ \} \\ \mathcal{C}_5 & = & \left\{ \begin{array}{l} A(x) \rightarrow (\exists R.A)(x), \qquad A(x) \rightarrow (\exists S.\{a\})(x), \\ S(y_1, x) \wedge S(y_2, x) \wedge S(y_3, x) \rightarrow y_1 \approx y_2 \vee y_2 \approx y_3 \vee y_1 \approx y_3 \end{array} \right\} \end{array} \tag{3.5}$$

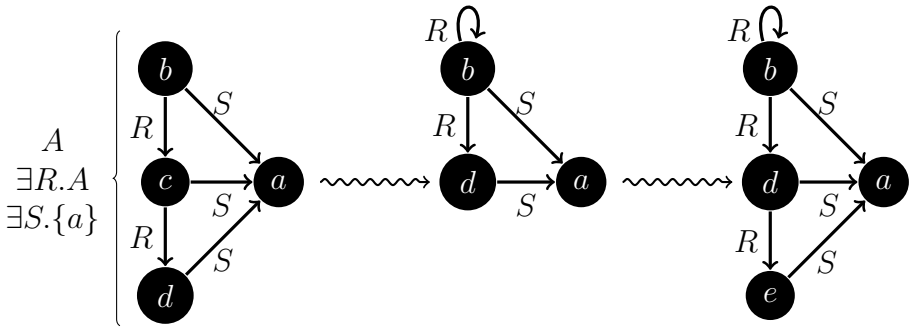


Figure 3.7: A Yo-Yo With Root Individuals

On $\mathcal{A}_5$ and $\mathcal{C}_5$, our calculus can produce the ABox $\mathcal{A}_5'$ shown on the left-hand side of Figure 3.7. ABox $\mathcal{A}_5'$ does not explicitly contain at-most restriction concepts, so the precondition of (*) cannot be checked directly; we shall discuss this issue shortly. For the moment, however, please note that the last DL-clause in $\mathcal{C}_5$ corresponds to the axiom $\top \sqsubseteq \leq 2\, S^-.\top$, so individuals c and d can be seen as satisfying the precondition of (*); therefore, we change them into root individuals. Furthermore, the third DL-clause from $\mathcal{C}_5$ is not satisfied, so the *Hyp*-rule derives $c \approx b$, and the $\approx$-rule can merge c into b. Since d is now not a blockable individual, we cannot prune it, so we obtain the ABox $\mathcal{A}_5''$ shown in the middle of Figure 3.7.[5] Since $\exists R.A(d)$ is not satisfied, we can extend $\mathcal{A}_5''$ with $R(d, e)$, $A(e)$, $\exists R.A(e)$, $\exists S.\{a\}(e)$, and $S(e, a)$ to produce the ABox $\mathcal{A}_5'''$ shown on the right-hand side of Figure 3.7. Individual e can be seen as satisfying the precondition of (*), so it is changed into a root individual. This ABox is isomorphic to $\mathcal{A}_5'$, so we can repeat the same inferences forever.

3.5.2 The Traditional Tableau Solution

To guarantee termination, the traditional $\mathcal{SHOIQ}$ tableau calculus [Horrocks and Sattler, 2007] uses an NN-rule that refines condition (*). We briefly summarize this rather complex tableau rule here; our simpler and more efficient solution is described in Section 3.5.3, below.

[5] To reduce clutter, we do not repeat the labels of individuals.

Assume that an ABox $\mathcal{A}$ contains an individual s that satisfies the precondition of (*)—that is, $\mathcal{A}$ contains assertions $R(s,a)$, $A(s)$, and $\leq n\, R^-.A(a)$, where s is a blockable individual that is not a successor of a root or named individual a. If $\mathcal{A}$ already contains root individuals $z_1, ..., z_n$ such that

$$\bigcup_{1\leq i\leq n} \{A(z_i), R(z_i,a)\} \cup \{z_i \neq z_j \mid 1 \leq i < j \leq n\} \subseteq \mathcal{A}$$

then the $\leq$-rule simply merges s into some z_i; no new root individual needs to be introduced. If $\mathcal{A}$ does not contain such $z_1, ..., z_n$, the NN-rule nondeterministically guesses the exact number $m \leq n$ of R^--neighbors of a that are members of A, generates m fresh root individuals $w_1, ..., w_m$, and extends $\mathcal{A}$ with the assertions

$$\{A(w_i), R(w_i,a) \mid 1 \leq i \leq m\} \cup \{w_i \not\approx w_j \mid 1 \leq i < j \leq m\} \cup \{\leq m\, R.A(a)\}.$$

This allows the NN-rule to be applied at most once for each concept of the form $\leq n\, R^-.A$ and each root individual, which ensures termination in the "yo-yo" case: the number of neighbors introduced for each root individual is clearly finite.

For example, Figure 3.8 shows applications of the NN- and $\leq$-rules to the ABox $\mathcal{A}_6 = \{\ \exists R.\exists R.\{c\}(a),\ \leq 3\, R^-.\top(c)\ \}$. In $\mathcal{A}_6{}^1$, shown in the left-hand side of the figure, the named individual c has a blockable R^--neighbor and must satisfy the restriction $\leq 3\, R^-.\top$. The NN-rule nondeterministically chooses whether to introduce one, two, or three fresh root individuals; the parallel branches of the derivation are shown in the center of the figure. The introduction of a single individual, shown in the top branch of Figure 3.8, results in deterministic application of the $\leq$-rule as the blockable individual b is merged into the fresh root individual z_1, shown in the upper right of the figure. For derivation paths on which more than one fresh root individual is introduced, application of the $\leq$-rule is nondeterministic: b can be merged into any of the new root individuals, with each choice resulting in a new branch of the

derivation. Such branching can be costly in practice: all derivation paths must be fully explored in order to identify an unsatisfiable knowledge base.

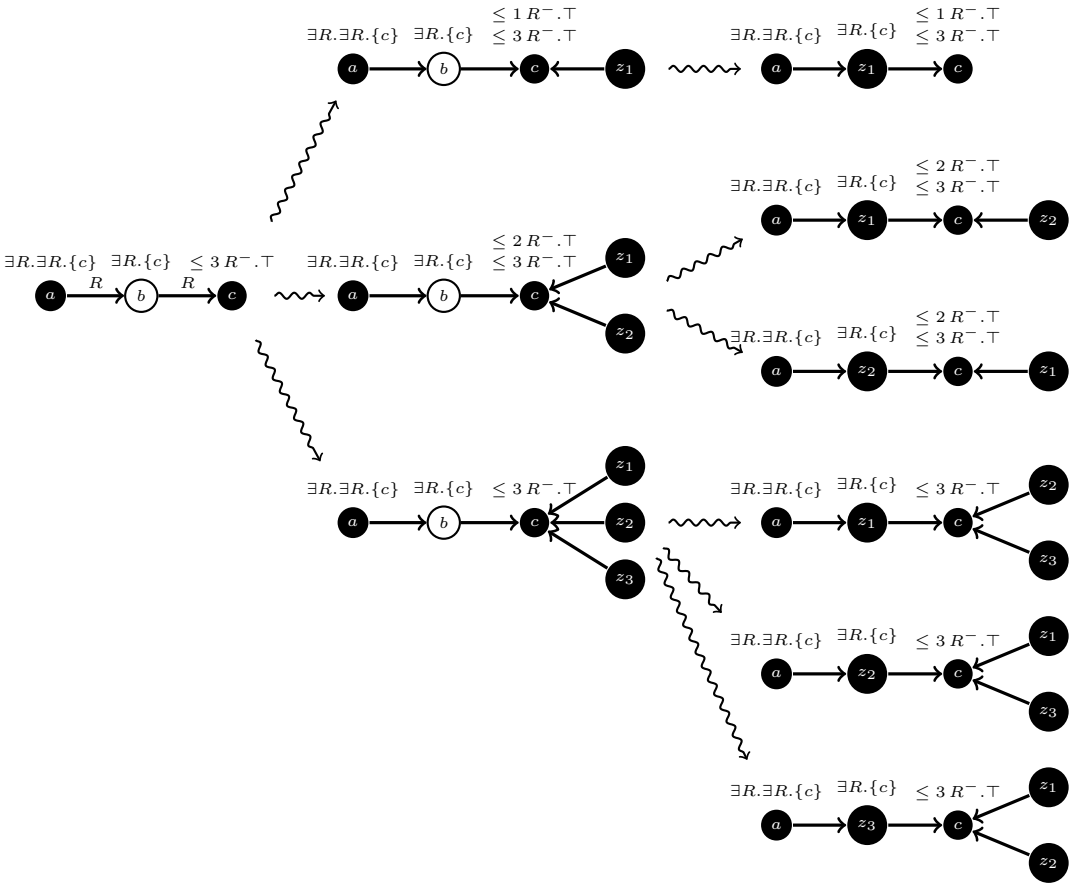


Figure 3.8: An application of the NN-rule

Although the NN-rule does ensure termination of the tableau algorithm, it is a potential source of inefficiency in knowledge bases in which large numbers appear within at-most concepts: an application of the NN-rule involving a concept $\leq n\, R.C$ guesses among n different possible sizes for the neighbor set, and subsequent applications of the $\leq$-rule must choose how to merge the new roots with blockable individuals. In the case of just a single blockable neighbor, this results in a derivation tree with n^2 branches. Furthermore, the introduction of new root individuals can result in unnecessary processing and large models.

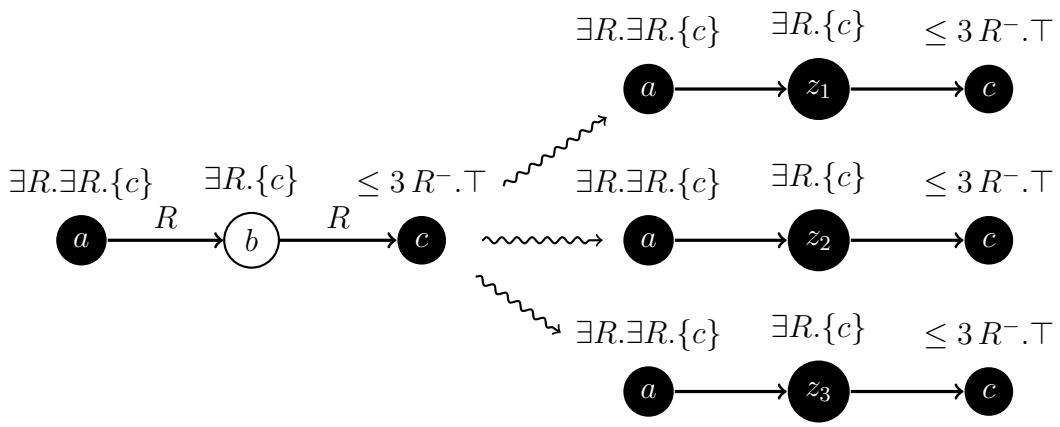


Figure 3.9: An application of the NI-rule

3.5.3 The *NI*-rule

As a replacement for the NN-rule described above, we introduce a new NI-rule, which also refines (*). Again assume that $\mathcal{A}$ contains an individual s that satisfies the precondition of (*)—that is, $\mathcal{A}$ contains assertions $R(s, a)$ and $A(s)$, where a is a root or a named individual, s is a blockable individual that is not a successor of a, and a must satisfy an at-most restriction $\leq n\, R^-.A$. In any model of $\mathcal{A}$, there can be at most n different individuals b_i that participate in assertions of the form $R(b_i, a)$ and $A(b_i)$. Hence, we associate with a a set of n fresh root individuals $\{b_1, \ldots, b_n\}$ that represent the R^--neighbors of a; unlike the root individuals introduced by the NN-rule, we do not assume that $b_i \neq b_j$. Instead of choosing some subset of these individuals to introduce and relying upon the $\leq$-rule to merge them with blockable neighbors, however, we promote blockable individuals to root individuals directly: to turn s into a root individual, we nondeterministically choose b_j from this set and merge s into b_j. In this way, the number of new root individuals that can be introduced as a result of the at-most restriction $\leq n\, R^-.A$ on a is limited to n.

An application of our NI-rule to the ABox $\mathcal{A}_7 = \{\ \exists R.\exists R.\{c\}(a),\ \leq 3\, R^-.\top(c)\ \}$ is given in Figure 3.9.

In the "root yo-yo" example from Figure 3.7, the NI-rule introduces at most

two fresh root individuals. When the *NI*-rule is applied for the third time, instead of introducing e, one of the previously introduced root individuals is reused, which ensures termination of the calculus.

The complete definition of the *NI*-rule is given in Table 5.4 on page 64.

3.5.4 Annotated Equalities

When formulating the *NI*-rule, we are faced with a technical problem: at-most restriction concepts are translated in our calculus into DL-clauses, which makes testing the condition (*) from page 26 difficult. For example, an application of the *Hyp*-rule to the third DL-clause in (3.5) (obtained from the axiom $\top \sqsubseteq \leq 2\, S^-.\top$) can produce an equality such as $c \approx b$; this equality alone does not reflect the fact that a must satisfy the at-most restriction $\leq 2\, S^-.\top$. To enable application of the *NI*-rule, we introduce *annotated equalities* in which the annotations establish an association with the at-most restriction. The third DL-clause from (3.5) is thus represented in our algorithm as follows:

$$\begin{aligned} S(y_1, x) \wedge S(y_2, x) \wedge S(y_3, x) \rightarrow \\ \quad y_1 \approx y_2 \, @^x_{\leq 2\, S^-.\top} \vee y_2 \approx y_3 \, @^x_{\leq 2\, S^-.\top} \vee y_1 \approx y_3 \, @^x_{\leq 2\, S^-.\top} \end{aligned} \tag{3.6}$$

The *Hyp*-rule then derives $c \approx b \, @^a_{\leq 2\, S^-.\top}$, which has the same meaning as $c \approx b$; however, the annotation says that, since a must satisfy the at-most restriction $\leq 2\, S^-.\top$, both b and c must also be merged with one of the (two) individuals reserved as S^--neighbors of a. The formal definition of annotated equalities is given in Definition 11 on page 55, and our final formulation of the *NI*-rule is given in Table 5.4 on page 64.

3.5.5 Nominals and Merging

The introduction of the *NI*-rule leads to another problem: repeated merging between root individuals can lead to nontermination in a "caterpillar" derivation. Consider, for example, an application of the hypertableau calculus to the following knowledge

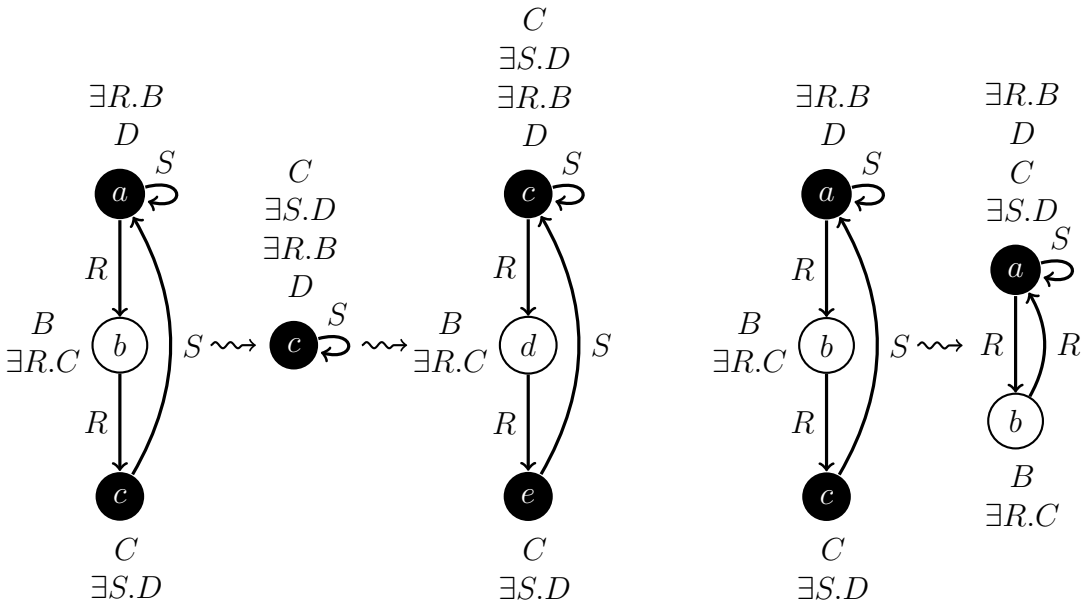
base:

$$\begin{array}{rcl} \mathcal{A}_8 & = & \{\ S(a,a), \quad \exists R.B(a)\ \} \\ \mathcal{C}_8 & = & \left\{ \begin{array}{ll} B(x) \to \exists R.C(x), & C(x) \to \exists S.D(x), \\ D(x) \to x \approx a, & S(y_1,x) \wedge S(y_2,x) \to y_1 \approx y_2 \,@^x_{\leq 1\, S^-.\top} \end{array} \right\} \end{array} \tag{3.7}$$

The ABox and the first DL-clause cause the introduction of two new blockable individuals b and c; the next two DL-clauses connect c with a by the role S; the last DL-clause produces $c \approx c\,@^x_{\leq 1\, S^-.\top}$; and an application of the *NI*-rule to this assertion causes c to become a root individual. The ABox $\mathcal{A}_8'$ resulting from these inferences is shown in the left-hand side of Figure 3.10a. Since S is inverse-functional, the individuals a and c must be merged. Because individual c is a root, it is no longer a descendant of a, so we can choose to merge a into c. The blockable individual b is then pruned (in order to avoid the problems outlined in Section 3.4), and the resulting ABox is shown in the middle part of Figure 3.10a. The existential restriction $\exists R.B$ on c, however, is not satisfied, so a similar sequence of rule applications constructs the ABox $\mathcal{A}_8''$ shown in the right-hand side of of Figure 3.10a. This ABox is isomorphic to $\mathcal{A}_8'$, so the same inferences can be repeated forever.

This problem can be intuitively explained by the following observation. The *NI*-rule introduces fresh root individuals as neighbors of an existing root individual; thus, each root individual in an ABox can be seen as a part of a "chain" showing which individual caused the introduction of which root individual. Each chain is initially anchored at a named individual: such individuals occur in the input ABox and are not introduced by the *NI*-rule. The length of a path of blockable individuals can be used to limit the length of the "chains" of root individuals. If we allow chain anchors to be removed from an ABox, then the chains remain limited in length in any given ABox. Over the course of derivation, however, one end of the chain can be extended indefinitely as the other end is shortened.

We solve this problem by allowing named individuals to be merged only into other named individuals, as specified by the postcondition of the $\approx$-rule in Table 5.4



(a) Nonterminating Variant

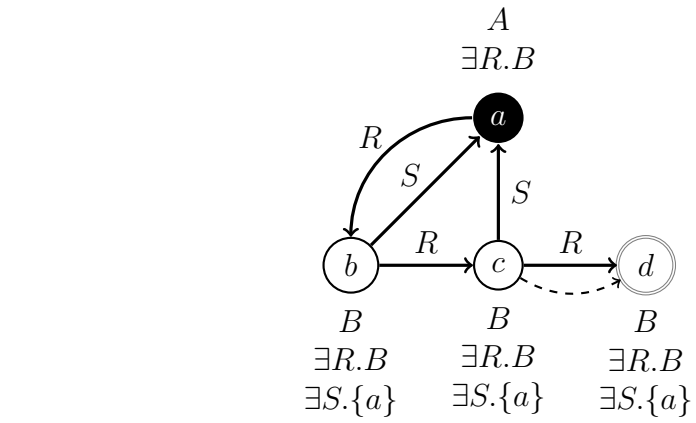
(b) Terminating Variant

Figure 3.10: The "Caterpillar" Example

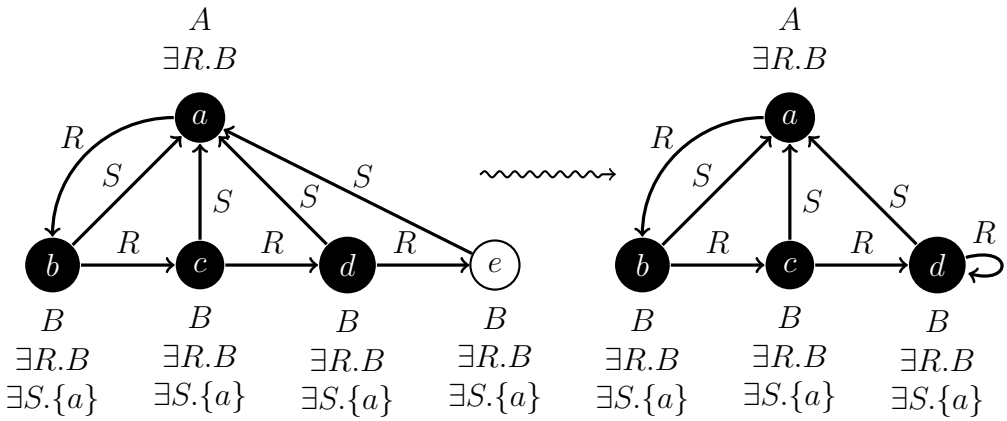
on page 64. This ensures that each chain of root individuals always remains anchored at a named individual. In our example, instead of merging a into c, we merge c into a, which results in the ABox shown in Figure 3.10b. No derivation rule is applicable to this ABox, so the algorithm terminates.

3.5.6 The *NI*-Rule and Unraveling

The *NI*-rule is required not only to ensure that ABoxes are forest shaped, but also to enable the application of blocking and unraveling. Consider, for example, the knowledge base $\mathcal{K}_9 = \{\mathcal{C}_9, \mathcal{A}_9, \emptyset\}$ shown in (3.8), in which we omit the annotations on equalities for the sake of clarity. Intuitively, the axioms of the knowledge base state that the individual a can have no R^--neighbors, and that there is an infinite chain of



(a) Premature Blocking



(b) A Correct Derivation

Figure 3.11: The *NI*-rule and Unraveling

individuals each of which is an S^--neighbor of a.

$$\begin{aligned} \mathcal{A}_9 &= \{\, A(a),\ (\exists R.B)(a),\ \} \\ \mathcal{C}_9 &= \left\{ \begin{array}{l} A(x) \wedge R(y,x) \rightarrow \bot,\quad B(x) \rightarrow (\exists R.B)(x),\quad B(x) \rightarrow (\exists S.\{a\})(x), \\ R(y_1,x) \wedge R(y_2,x) \rightarrow y_1 \approx y_2, \\ S(y_1,x) \wedge S(y_2,x) \wedge S(y_3,x) \wedge S(y_4,x) \rightarrow \\ \qquad y_1 \approx y_2 \vee y_1 \approx y_3 \vee y_1 \approx y_4 \vee y_2 \approx y_3 \vee y_2 \approx y_4 \vee y_3 \approx y_4, \end{array} \right\} \end{aligned} \tag{3.8}$$

Without the *NI*-rule, an application of our calculus to $\mathcal{A}_9$ and $\mathcal{C}_9$ might produce the ABox $\mathcal{A}_9'$ shown in Figure 3.11a. The individual d is blocked in $\mathcal{A}_9'$ by the individual c, so the derivation terminates. Note that the last DL-clause from $\mathcal{C}_9$ (which corresponds to the axiom $\top \sqsubseteq \leq 3\, S^-.\top$) is satisfied: a is the only individual in $\mathcal{A}_9'$ that has S^--neighbors and it has only two such neighbors. To construct a model from

$\mathcal{A}_9'$, we unravel the blocked parts of the ABox—that is, we construct an infinite path that extends past d by "duplicating" the fragment of the model between c and d an infinite number of times. This, however, creates additional S^--neighbors of a, which invalidates the last DL-clause from $\mathcal{C}_9$; thus, the unraveled ABox does not define a model of $\mathcal{A}_9$ and $\mathcal{C}_9$.

The *NI*-rule elegantly solves this problem. Since a must satisfy an at-most restriction of the form $\leq 3\, S^-.\top$, as soon as $S(b, a)$, $S(c, a)$, and $S(d, a)$ are derived, the *NI*-rule is applied to turn b, c, and d into root individuals. This corrects the problems with unraveling: root individuals do not become blocked, so we introduce another fresh blockable individual e. This individual is merged with another S^--neighbor of a, producing an individual with two R^--neighbors, as illustrated in Figure 3.11b. R is inverse-functional, however, so the neighbors are merged. Merging continues until b has been merged into a, causing a to become its own R-neighbor, at which point our algorithm correctly determines that the knowledge base represented by $\mathcal{A}_9$ and $\mathcal{C}_9$ is unsatisfiable.

Chapter 4

Algorithm Overview

In this chapter we present an informal overview of our hypertableau algorithm that addresses the problems due to or- and and-branching outlined in Chapter 3. We begin by describing the relationship between our calculus and resolution-based techniques.

4.1 Derivation Rules

The hyperresolution calculus [Robinson, 1965] has often been used for first-order theorem proving. It works on *clauses*—implications of the form $\bigwedge_{i=1}^{n} U_i \rightarrow \bigvee_{j=1}^{m} V_j$ where U_i and V_j are first-order atoms. The conjunction $\bigwedge_{i=1}^{n} U_i$ is called the *antecedent*, and the disjunction $\bigvee_{j=1}^{m} V_j$ is called the *consequent*; we sometimes omit $\rightarrow$ if the antecedent is empty. For $\mathbf{D_i}$ a possibly empty disjunction of literals and σ the most general unifier of $(A_1, B_1), \ldots, (A_m, B_m)$, the hyperresolution derivation rule is defined as follows (assuming that the unifier σ exists):

$$\frac{A_1 \vee \mathbf{D_1} \qquad \ldots \qquad A_m \vee \mathbf{D_m} \qquad B_1 \wedge \ldots \wedge B_m \rightarrow C_1 \vee \ldots \vee C_k}{\mathbf{D_1}\sigma \vee \ldots \vee \mathbf{D_m}\sigma \vee C_1\sigma \vee \ldots \vee C_k\sigma}$$

As is usual in resolution theorem proving, the above notation is intended to be agnostic with respect to disjunct ordering: the notation $A_i \vee \mathbf{D_i}$ does not imply that A_i is the left-most disjunct in the disjunction. To make such a hyperresolution calculus refutationally complete for first-order logic, one additionally needs a *factoring* derivation rule, which we do not discuss any further.

The hypertableau calculus [Baumgartner *et al.*, 1996] is based on the observation that, if the literals in $C_1\sigma \vee \ldots \vee C_n\sigma$ do not share variables, we can replace the clause with a nondeterministically chosen atom $C_i\sigma$ that we assume to be true. If we assume that all clauses are safe (i.e., that each variable occurring in a clause also occurs in the clause's antecedent), then clauses with empty antecendents, such as $A_i \vee \mathbf{D_i}$, and $C_1\sigma \vee \ldots \vee C_n\sigma$, are always ground, so they can always be nondeterministically split into atoms. An inference in such a hypertableau calculus is written as

$$\frac{A_1 \quad \ldots \quad A_m \qquad B_1 \wedge \ldots \wedge B_m \rightarrow C_1 \vee \ldots \vee C_k}{C_1\sigma \quad | \quad \ldots \quad | \quad C_k\sigma}$$

where σ is the most general unifier of $(A_1, B_1), \ldots, (A_m, B_m)$ and $|$ represents or-branching. On Horn clauses, each inference is deterministic,[1] and the calculus exhibits a "minimal" amount of don't-know nondeterminism on general clauses.

The hypertableau calculus can be easily applied to DLs: GCIs can be translated into first-order formulae [Borgida, 1996], which can then be converted into clauses by Skolemization, as shown in the following example.

$$A \sqsubseteq \exists R.B \quad \rightsquigarrow \quad \forall x : [A(x) \rightarrow \exists y : R(x,y) \wedge B(y)] \quad \rightsquigarrow \quad \begin{array}{l} A(x) \rightarrow B(f(x)) \\ A(x) \rightarrow R(x, f(x)) \end{array}$$

Let $\mathcal{A}$ be an ABox containing the assertions $A(a)$, $R(a,b)$, and $B(b)$. The GCI $A \sqsubseteq \exists R.B$ is clearly satisfied in $\mathcal{A}$, so there is no need to perform any inference. The clauses obtained by Skolemization, however, are not satisfied in $\mathcal{A}$, so the hypertableau calculus derives $R(a, f(a))$ and $B(f(a))$. Hence, Skolemization may make the calculus perform unnecessary inferences, which may be inefficient.

In order to avoid this, our calculus does not work with Skolemized clauses, but instead preserves the limited forms of explicit existential quantification permitted in description logics.

[1] As mentioned before, the order in which inferences are applied is nevertheless don't-care nondeterministic.

4.2 Calculus Overview

We begin by preprocessing a $\mathcal{SROIQ}$ KB $\mathcal{K}$ into a pair $\Xi(\mathcal{K}) = (\Xi_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$, where $\Xi_{\mathcal{A}}(\mathcal{K})$ is an ABox and $\Xi_{\mathcal{TR}}(\mathcal{K})$ is a set of *DL-clauses*—implications of the form $\bigwedge_{i=1}^{n} U_i \to \bigvee_{j=1}^{m} V_j$, where U_i are of the form $R(x, y)$ or $A(x)$, and V_j are of the form $R(x, y)$, $A(x)$, $\geq n\, R.C(x)$, or $x \approx y$. The preprocessing step is introduced formally in Section 5.1.

The DL-clauses in $\Xi_{\mathcal{TR}}(\mathcal{K})$ are used in the *Hyp*-rule, which is inspired by the hypertableau derivation rule and forms the basis of our calculus. For example, a GCI $\exists R.\neg A \sqsubseteq B$ is translated into a DL-clause $R(x, y) \to B(x) \vee A(y)$; then, if an ABox contains $R(a, b)$, the *Hyp*-rule derives either $B(a)$ or $A(b)$.

Our calculus handles the existential quantification implicit in assertions of the form $\geq n\, R.C(a)$ directly using the tableau-style $\geq$-rule, which simply introduces new ground facts and constants.

At-most restrictions are translated in our approach into DL-clauses containing equalities; for example, the axiom $A \sqsubseteq \leq 2\, R.B$ is translated into the DL-clause

$$A(x) \wedge R(x, y_1) \wedge B(y_1) \wedge R(x, y_2) \wedge B(y_2) \wedge R(x, y_3) \wedge B(y_3) \to y_1 \approx y_2 \vee y_1 \approx y_3 \vee y_2 \approx y_3.$$

While a concept of the form $\leq n\, R.B$ can be encoded using $O(\log n)$ bits, the corresponding DL-clause contains $O(n^2)$ literals; our translation thus incurs an exponential blowup. Neither traditional tableau algorithms nor naïve hypertableau algorithms are likely to be able to efficiently handle large numbers in number restrictions, and specialized algorithms, such as those proposed by Haarslev and Möller [2001b], Haarslev *et al.* [2001b], and Faddoul *et al.* [2008], may be required. In practice, however, few ontologies in common use employ large number restrictions, and this blowup seldom dominates the overall space or time requirements of our algorithm.

Because of the translation described in the previous paragraph, the *Hyp*-rule can derive equalities of the form $s \approx t$. These are then dealt with using the $\approx$-rule: when-

ever $s \approx t \in \mathcal{A}$ and $s \neq t$, the $\approx$-rule replaces s with t or vice versa in all assertions in $\mathcal{A}$; this is usually called *merging*.

Apart from the *Hyp*-, the $\geq$-, and the $\approx$-rules, our calculus contains the $\perp$-rule that detects obvious contradictions of the form $s \not\approx s$, and the *NI*-rule that ensures termination in the presence of nominals, number restrictions, and inverse roles. We discuss the *NI*-rule in more detail in Section 3.5.

The rules of the algorithm are formalized in Definition 13 on page 60 and Table 5.4 on page 64, and the reader may find it useful to briefly examine these definitions before continuing.

4.3 Anywhere Pairwise Blocking

We employ pairwise blocking as discussed in Section 3.3 to ensure termination of the calculus; to curb and-branching, however, we extend it to *anywhere pairwise blocking*. The key idea is to extend the set of potential blockers for s beyond the ancestors of s. In doing so, we must avoid cyclic blocks: if s is allowed to block t and t can block s, then neither s nor t is guaranteed to have all its successors constructed, which would render the calculus incorrect. Therefore, we parameterize our algorithm with a strict ordering $\prec$ on individuals that contains the ancestor relation (i.e. if a is an ancestor of b, then $a \prec b$). We allow t to block s only if, in addition to the conditions mentioned in Section 3.3, we have $t \prec s$. This version of blocking is formalized in Definition 13 on page 60. Note that, if $\prec$ coincides with the ancestor relation, then anywhere blocking becomes equivalent to ancestor blocking.

Anywhere blocking can reduce and-branching in practice. Consider again the knowledge base $\mathcal{K}_2$ from Section 3.3. After we exhaust the exponentially-many members of Π, all subsequently-created individuals will be blocked. In the best case, we can always choose B_j instead of C_j, so we create a polynomial path in the tree and then use the individuals from that path to block their siblings, as shown in Figure 3.2b.

Hence, there is a derivation for $\mathcal{K}_2$ with anywhere blocking that can be constructed in polynomial time.

4.4 Nominal Guard Concepts

Negative nominal concepts of the form $\neg\{a\}$ can be converted to ABox assertions during preprocessing, so our calculus need not handle them. Positive nominal concepts, however, are naturally translated into equalities containing constants; for example, $\top \sqsubseteq \neg A \sqcup \{a\}$ corresponds to $A(x) \to x \approx a$. Such DL-clauses are inconvenient: given an equality assertion $a \approx b$, the $\approx$-rule would need to replace all occurrences of a with b not only in the assertions, but in the DL-clauses as well; thus, the mentioned DL-clause should be replaced with $A(x) \to x \approx b$.

We avoid rewriting DL-clauses by replacing constants in the antecedents of DL-clauses with new variables, which we bind to the original constants using *nominal guard concepts*. For example, $\top \sqsubseteq \neg A \sqcup \{a\}$ is transformed into the DL-clause $A(x) \wedge O_a(z_{\{a\}}) \to x \approx z_{\{a\}}$ and the assertion $O_a(a)$; the new predicate O_a is a nominal guard concept. All constants are thus "pushed" into the assertions, so the $\approx$-rule can perform replacements only in the ABox.

This technique allows the set of DL-clauses to remain constant throughout a derivation, which simplifies analysis of our calculus. A constant clause set also aids efficient implementations of the *Hyp*-rule.

Chapter 5

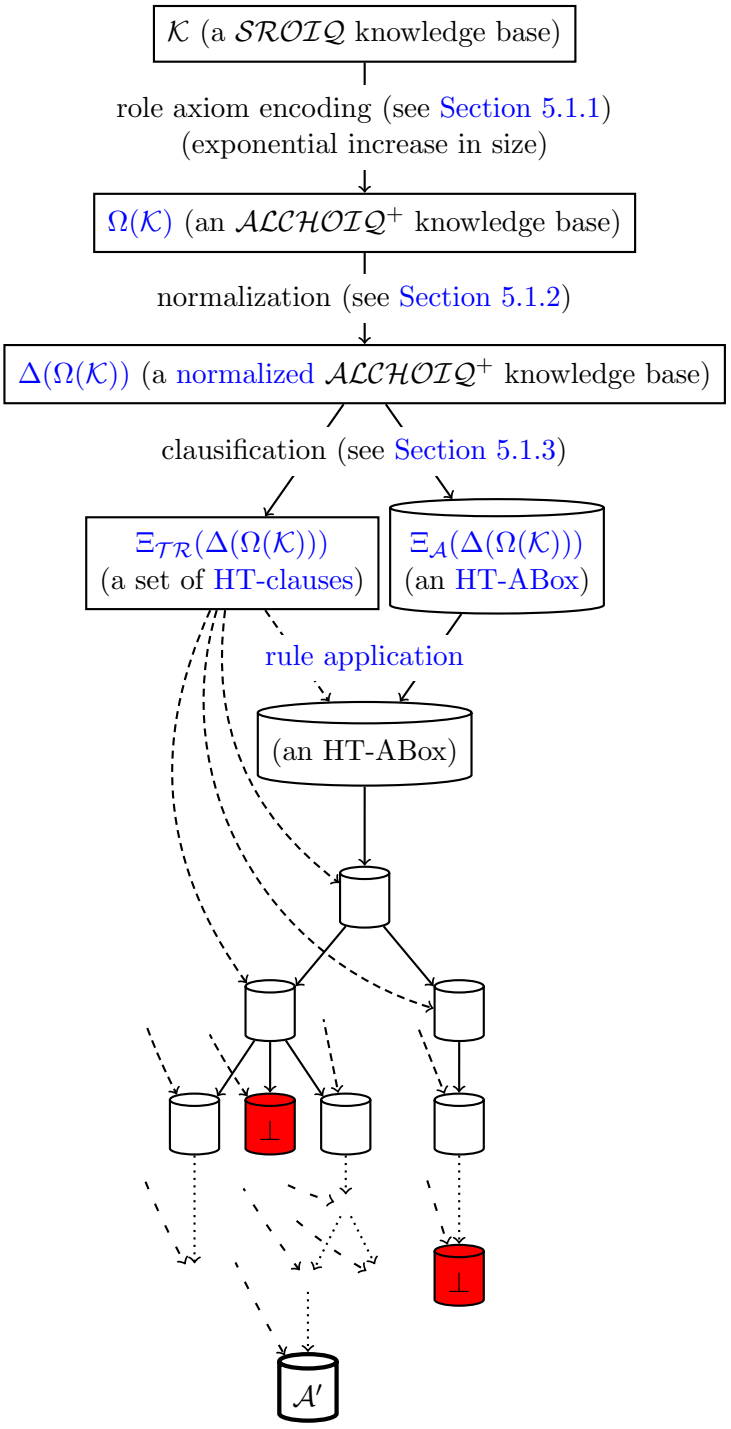
The Hypertableau Calculus for $\mathcal{SROIQ}$

We now formally present the hypertableau algorithm that can be used to test consistency of a $\mathcal{SROIQ}$ knowledge base $\mathcal{K}$. In the case that $\mathcal{K}$ is consistent, our procedure can also be to used to construct a model which shares some useful properties with models of $\mathcal{K}$. High-level reasoning services such as classification can often be implemented more efficiently using such models than is possible using black-box consistency testing alone; the procedure described in Chapter 10 for extraction of subsumption information is one example.

The various steps in our algorithm are summarized in Figure 5.1. The algorithm consists of two major phases: the *preprocessing* phase transforms a Description Logic knowledge base into an equisatisfiable first-order theory of a particular form; this phase is described in Section 5.1. The *hypertableau* phase searches for (an abstraction of) a model of the transformed knowledge base, and is described in Section 5.2.

5.1 Preprocessing

The goal of the preprocessing phase is to transform a $\mathcal{SROIQ}$ knowledge base $\mathcal{K}$ into an ABox $\Xi_{\mathcal{A}}(\mathcal{K})$ and a set of *DL-clauses* $\Xi_{\mathcal{TR}}(\mathcal{K})$ such that there is a correspondence between models of $\Xi_{\mathcal{A}}(\mathcal{K}) \cup \Xi_{\mathcal{TR}}(\mathcal{K})$ and models of $\mathcal{K}$. This correspondence is closely related to the notion of *model-conservative extensions*, as described by Lutz *et al.*



Preprocessing encodes a $\mathcal{SROIQ}$ knowledge base $\mathcal{K}$ as a set of HT-clauses and an HT-ABox which together form a *CMC-encoding* (defined in Definition 6) of $\mathcal{K}$. Such an encoding is equisatisfiable with $\mathcal{K}$, and models of the encoding coincide with models of $\mathcal{K}$ on concept and individual names occurring in $\mathcal{K}$.

The *hypertableau calculus* (described in Section 5.2) repeatedly applies derivation rules to elaborate the implications of clauses and previously-derived facts. If an ABox $\mathcal{A}'$ can be derived which does not contain $\bot$, and to which no further derivation rules can be applied, then a model for $\Xi(\Delta(\Omega(\mathcal{K})))$ can be constructed from $\mathcal{A}'$. If no such ABox exists then $\Xi(\Delta(\Omega(\mathcal{K})))$ is inconsistent. The maximum depth of such a derivation tree is doubly-exponential in the size of $\Xi(\Delta(\Omega(\mathcal{K})))$.

Figure 5.1: Overview of Hypertableau Reasoning

[2007]. We introduce the preprocessing phase by first defining its end result, and then summarizing the individual steps involved.

Definition 6 (Model-Conservative Encoding) Let $\mathcal{K}$ be a first-order theory over signature $\Sigma = (N_R, N_C, N_I)$, and let $\mathcal{K}'$ be a first-order theory over (possibly different) signature Σ'. Then $\mathcal{K}'$ is a *model-conservative encoding* of $\mathcal{K}$ if:

- for every model $\mathcal{I}$ of $\mathcal{K}$, there exists a model $\mathcal{I}'$ of $\mathcal{K}'$ such that $\mathcal{I}'$ and $\mathcal{I}$ coincide on (N_R, N_C, N_I), and
- for every model $\mathcal{I}'$ of $\mathcal{K}'$, there exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I}'$ and $\mathcal{I}$ coincide on (N_R, N_C, N_I).

$\mathcal{K}'$ is a *concept-model-conservative encoding* (abbreviated *CMC encoding*) of $\mathcal{K}$ if:

- for every model $\mathcal{I}$ of $\mathcal{K}$, there exists a model $\mathcal{I}'$ of $\mathcal{K}'$ such that $\mathcal{I}'$ and $\mathcal{I}$ coincide on $(\emptyset, N_C, N_I)$, and
- for every model $\mathcal{I}'$ of $\mathcal{K}'$, there exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I}'$ and $\mathcal{I}$ coincide on $(\emptyset, N_C, N_I)$. △

Unlike model-conservative extensions, model-conservative encodings of $\mathcal{K}$ do not necessarily include $\mathcal{K}$ as a subset. Further, CMC encodings preserve only the interpretations of concepts and individuals, not the interpretations of roles.

Definition 7 (DL-Clause) Let N_V be a set of *variables* disjoint from the set of individuals N_I. An *atom* is an expression of the form $\top(s)$, $A(s)$, $\geq n\, S.A(s)$, $R(s,t)$, or $s \approx t$, for s and t individuals or variables, A an atomic concept, R an atomic role, S a (possibly inverse) role, and n a positive integer. A *DL-clause* is an expression of the form

$$U_1 \wedge \ldots \wedge U_m \rightarrow V_1 \vee \ldots \vee V_n$$

Table 5.1: Satisfaction of DL-Clauses in an Interpretation

$\mathcal{I}, \mu \models \top(s)$	iff	$s^{\mathcal{I},\mu} \in \Delta^{\mathcal{I}}$
$\mathcal{I}, \mu \models A(s)$	iff	$s^{\mathcal{I},\mu} \in A^{\mathcal{I}}$
$\mathcal{I}, \mu \models \geq n\, S.A(s)$	iff	$s^{\mathcal{I},\mu} \in (\geq n\, S.A)^{\mathcal{I}}$
$\mathcal{I}, \mu \models R(s,t)$	iff	$\langle s^{\mathcal{I},\mu}, t^{\mathcal{I},\mu} \rangle \in R^{\mathcal{I}}$
$\mathcal{I}, \mu \models s \approx t$	iff	$s^{\mathcal{I},\mu} = t^{\mathcal{I},\mu}$
$\mathcal{I}, \mu \models \bigwedge_{i=1}^{m} U_i \rightarrow \bigvee_{j=1}^{n} V_j$	iff	$\mathcal{I}, \mu \models U_i$ for each $1 \leq i \leq m$ implies $\mathcal{I}, \mu \models V_j$ for some $1 \leq j \leq n$
$\mathcal{I} \models \bigwedge_{i=1}^{m} U_i \rightarrow \bigvee_{j=1}^{n} V_j$	iff	$\mathcal{I}, \mu \models \bigwedge_{i=1}^{m} U_i \rightarrow \bigvee_{j=1}^{n} V_j$ for all mappings μ
$\mathcal{I} \models \mathcal{C}$	iff	$\mathcal{I} \models r$ for each DL-clause $r \in \mathcal{C}$

where U_i and V_j are atoms, $m \geq 0$, and $n \geq 0$. The conjunction $U_1 \wedge \ldots \wedge U_m$ is called the *antecedent*, and the disjunction $V_1 \vee \ldots \vee V_n$ is called the *consequent*. The empty antecedent and the empty consequent of a DL-clause are written as $\top$ and $\bot$, respectively.

Let $\mathcal{I} = (\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation and $\mu : N_V \rightarrow \triangle^{\mathcal{I}}$ a mapping of variables to elements of the interpretation domain. Let $a^{\mathcal{I},\mu} = a^{\mathcal{I}}$ for an individual a and $x^{\mathcal{I},\mu} = \mu(x)$ for a variable x. Satisfaction of an atom, a DL-clause, and a set of DL-clauses $\mathcal{C}$ in $\mathcal{I}$ and μ is defined in Table 5.1. $\triangle$

Note that DL clauses contain no negation, even within atoms. Our algorithm encodes literal concepts using new atomic concept names, e.g. by replacing $\neg A$ with a new concept $Q_{\neg A}$, and enforcing the semantic connection between A and $Q_{\neg A}$ with a DL-clause of the form $A(x) \wedge Q_{\neg A}(x) \rightarrow \bot$. For the sake of brevity, we use $\neg A$ as a shorthand for the atomic concept $Q_{\neg A}$ in DL-clauses, and omit the extra DL-clauses in examples.

For ease of presentation, the preprocessing phase is broken down into several distinct steps, as shown in Figure 5.1. First, complex role inclusion axioms are encoded as TBox axioms; the procedure is described in Definition 8. Next, complex concepts occurring in the ABox and nested within TBox axioms are replaced with atomic concepts whose semantics are defined with additional axioms, and all axioms are

transformed to a simpler form; this *normalization* is described in Section 5.1.2. Finally, the normalized Description Logic knowledge base is translated into the clause format, and accompanying ABox, used directly by our reasoning algorithm; *clausification* is described in Section 5.1.3.

5.1.1 Elimination of Role Inclusion Axioms

Complex role inclusion axioms are handled in tableau algorithms using a collection of special expansion rules. For example, if R is transitive ($RR \sqsubseteq R$) and an ABox contains $\forall R.C(s)$ and $R(s,t)$, then a special $\forall_+$-rule derives $\forall R.C(t)$.

In our algorithm, however, concepts of the form $\forall R.C$ are translated into DL-clauses, so such a $\forall_+$-rule cannot easily be applied. Instead of handling complex role inclusions directly, we encode a $\mathcal{SROIQ}$ knowledge base $\mathcal{K}$ into an $\mathcal{ALCHOIQ}^+$ knowledge base $\Omega(\mathcal{K})$. This encoding eliminates all complex role inclusion axioms, but simulates their effects using additional GCIs.

Our encoding makes use of the standard translation of complex role inclusion axioms into automata, as described by Horrocks and Sattler [2004] and by Kutz *et al.* [2006]. Given a role box $\mathcal{R}$, this translation produces, for each role R, a nondeterministic finite automata $\mathcal{B}_R$ such that if a path $R_1(a_0,a_1)...R_n(a_{n-1},a_n)$ requires the presence of $R(a_0,a_n)$ in all models of $\mathcal{R}$, then $\mathcal{B}_R$ accepts the string $R_1...R_n$. For example, if the role R is transitive in $\mathcal{R}$, then $\mathcal{B}_R$ accepts the strings R, RR, RRR, etc.

For each universal $\forall R.C$ occurring in a knowledge base, we encode the automaton $\mathcal{B}_R$ by introducing new atomic concepts for each state and adding GCIs to model transitions between states, as follows:

Definition 8 Given a $\mathcal{SROIQ}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{R})$, the *concept closure* of $\mathcal{K}$ is the smallest set of concepts $({}^+\mathcal{K})$ such that

- if $C \sqsubseteq D \in \mathcal{T}$, then $\mathsf{nnf}(\neg C \sqcup D) \in ({}^+\mathcal{K})$,

- if $C(a) \in \mathcal{A}$, then $\mathsf{nnf}(C) \in ({}^{+}\mathcal{K})$,
- if $C \in ({}^{+}\mathcal{K})$ and D syntactically occurs in C, then $D \in ({}^{+}\mathcal{K})$,
- if $\leq n\, R.C \in ({}^{+}\mathcal{K})$, then $\doteq\!\neg C \in ({}^{+}\mathcal{K})$, and
- if $\forall R.C \in ({}^{+}\mathcal{K})$, $S \sqsubseteq^{*}_{\mathcal{R}} R$, and $\mathsf{Tra}(S) \in \mathcal{R}$, then $\forall S.C \in ({}^{+}\mathcal{K})$.

For each role R occurring in $\mathcal{K}$, let the translation of R wrt $\mathcal{K}$ as described by Kutz *et al.* [2006] be the nondeterministic finite automaton $\mathcal{B}_R = (P_R, N_R \cup \{\epsilon\}, \delta_R, i_R, F_R)$ with states P_R, input symbols $N_R \cup \{\epsilon\}$, nondeterministic transition function δ_R, initial state i_R, and final states F_R. For simplicity, we assume that the automaton $\mathcal{B}_R$ and $\mathcal{B}_{R'}$ for distinct roles R and R' do not share states (i.e. $P_R \cap P_{R'} = \emptyset$). We use $L(\mathcal{B}_R)$ to denote the language accepted by $\mathcal{B}_R$.

The Ω-*encoding* of $\mathcal{K}$ is the $\mathcal{ALCHOIQ}^{+}$ knowledge base $\Omega(\mathcal{K}) = (\mathcal{R}', \mathcal{T}', \mathcal{A})$ where $\mathcal{R}'$ is obtained from $\mathcal{R}$ by removing all complex role inclusion axioms and

$$\begin{aligned}\mathcal{T}' = \mathcal{T} &\cup \{\forall R.C \sqsubseteq Q^{C}_{i_R} \mid \forall R.C \in ({}^{+}\mathcal{K})\} \\ &\cup \{Q^{C}_{f} \sqsubseteq C \mid \forall R.C \in ({}^{+}\mathcal{K}) \text{ and } f \in F_R\} \\ &\cup \{Q^{C}_{p} \sqsubseteq \forall R'.Q^{C}_{p'} \mid \forall R.C \in ({}^{+}\mathcal{K}),\ R' \in N_R, \text{ and } p' \in \delta_R(p, R')\} \\ &\cup \{Q^{C}_{p} \sqsubseteq Q^{C}_{p'} \mid \forall R.C \in ({}^{+}\mathcal{K}) \text{ and } p' \in \delta_R(p, \epsilon)\}\end{aligned}$$

where Q^{C}_{p} is a fresh atomic concept for each concept C, and automata state p. $\triangle$

Similar encodings are known for various description [Tobies, 2001] and modal [Schmidt and Hustadt, 2003] logics. Note that, in order to guarantee decidability [Horrocks *et al.*, 2000a], number restrictions and local reflexivity are allowed in $\mathcal{SROIQ}$ only on simple roles—that is, on roles not occurring as consequents of complex role inclusion axioms; for similar reasons, role disjointness, irreflexivity, and asymmetry axioms are also allowed only on simple roles.

We begin by recalling three results by Kutz *et al.* [2006] that will be useful in the proofs that follow:

Lemma 1 ([Kutz *et al.*, 2006]) *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{R})$ be a $\mathcal{SROIQ}$ knowledge base. Then $\mathcal{I}$ is a model of $\mathcal{R}$ if and only if, for each (possibly inverse) role R occurring in $\mathcal{R}$, each word $\omega \in L(\mathcal{B}_R)$, and each $\langle x, Y \rangle \in \omega^{\mathcal{I}}$, it is the case that $\langle x, y \rangle \in R^{\mathcal{I}}$.*

Lemma 2 ([Kutz *et al.*, 2006]) *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{R})$ be a $\mathcal{SROIQ}$ knowledge base, and let R be a role occurring in $\mathcal{K}$. Then $R \in L(\mathcal{B}_R)$, and, if $\omega \sqsubseteq R \in \mathcal{R}$, then $\omega \in L(\mathcal{B}_R)$.*

Lemma 3 ([Kutz *et al.*, 2006]) *For knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{R})$ and role R, the size of the the automaton encoding $\mathcal{B}_R$ of R with respect to $\mathcal{K}$ is bounded exponentially in the depth*

$$d_{\mathcal{K}} = \max\{n \mid \textit{there exist } S_1 \prec ... \prec S_n, u_i, v_i \textit{ with } u_i S_{i-1} v_i \sqsubseteq S_i \in \mathcal{R}\}$$

We now show that the automaton encoding preserves the semantics of universal restrictions:

Lemma 4 *Let $\mathcal{K}$ be a $\mathcal{SROIQ}$ knowledge base and $\mathcal{I} = (\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$ a model of $\Omega(\mathcal{K})$. Further, let $R, R_1, ..., R_n$ be (possibly inverse) roles, $\mathcal{B}_R$ the automaton translation of R with respect to $\mathcal{K}$ as described by Kutz* et al. *[2006], C a concept, and x and y elements of $\triangle^{\mathcal{I}}$ such that:*

- $\forall R.C \in \mathcal{K}^+$
- $x \in (\forall R.C)^{\mathcal{I}}$,
- $\langle x, y \rangle \in R_1^{\mathcal{I}} \circ ... \circ R_n^{\mathcal{I}}$, *and*
- $R_1 ... R_n \in L(\mathcal{B}_R)$.

Then $y \in C^{\mathcal{I}}$.

Proof Assume that $R_1...R_n \in L(\mathcal{B}_R)$. Then there exists a sequence

$$p_0^0, p_0^1, ..., p_0^{m_0}, p_1^0...p_1^{m_1}, ..., p_n^{m_n}$$

of states of $\mathcal{B}_R$, with each m_i a positive integer, such that:

- $p_0^0 = i_R$
- $p_i^{j+1} \in \delta_R(p_i^j, \epsilon)$
- $p_{i+1}^0 \in \delta_R(p_i^{m_i}, R_{i+1})$
- $p_n^{m_n} \in F_R$

Let $\mathcal{I} = (\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a model of $\Omega(\mathcal{K})$, and x and y elements of $\triangle^{\mathcal{I}}$ such that $x \in (\forall R.C)^{\mathcal{I}}$ and $\langle x, y\rangle \in R_1^{\mathcal{I}} \circ ... \circ R_n^{\mathcal{I}}$. Then there exist elements $z_0, ...z_n$ of $\triangle^{\mathcal{I}}$ such that $z_0 = x$, $z_n = y$, and $\langle z_{i-1}, z_i\rangle \in R_i^{\mathcal{I}}$ for each $1 \leq i \leq n$.

Finally, assume that $z_0 \in (\forall R.C)^{\mathcal{I}}$. We show by induction on i and j that for each state p_i^j such that $0 \leq i \leq n$ and $0 \leq j \leq m_i$, it is the case that $z_i \in (Q^C_{p_i^j})^{\mathcal{I}}$. For the base case, note that since $\mathcal{I}$ is a model of $\Omega(\mathcal{K})$, it satisfies the axiom $\forall R.C \sqsubseteq Q^C_{i_R}$, thus $z_0 \in (Q^C_{i_R})^{\mathcal{I}}$. Substituting p_0^0 for i_R, we have $z_0 \in (Q^C_{p_0^0})^{\mathcal{I}}$. For induction on j, note that $\mathcal{I}$ must also satisfy $Q_p^C \sqsubseteq Q_{p'}^C$ where $p = p_i^j$ and $p' = p_i^{j+1}$, which is an axiom of $\Omega(\mathcal{K})$ since $p_i^{j+1} \in \delta_R(p_i^j, \epsilon)$; thus if $z_i \in (Q_p^C)^{\mathcal{I}}$ for $p = p_i^j$ it is the case that $z_i \in (Q_{p'}^C)^{\mathcal{I}}$ for $p' = p + i^{j+1}$. For induction on i, note that $\mathcal{I}$ must satisfy $Q_p^C \sqsubseteq \forall R_{i+1}.Q_{p'}^C$ for $p = p_i^{m_i}$ and $p' = p_{i+1}^0$, which is an axiom of $\Omega(\mathcal{K})$ since $p_{i+1}^0 \in \delta_R(p_i^{m_i}, R_{i+1})$. Because $\langle z_i, z_{i+1}\rangle \in R_{i+1}^{\mathcal{I}}$, if $z_i \in (Q_p^C)^{\mathcal{I}}$ for $p = p_i^{m_i}$, it is thus the case that $z_{i+1} \in (Q_{p'}^C)^{\mathcal{I}}$ for $p' = p_{i+1}^0$.

We thus conclude by induction that $z_n \in (Q_p^C)^{\mathcal{I}}$ for $p = p_n^{m_n}$. Since $p_n^{m_n} \in F_R$, our model $\mathcal{I}$ must satisfy $Q_p^C \sqsubseteq C$ for $p = p_n^{m_n}$, so $z_n \in C^{\mathcal{I}}$. By substitution, we conclude that $y \in C^{\mathcal{I}}$. $\square$

We next demonstrate that our translation preserves the semantics of the original knowledge base. The proof of this lemma is a natural extension of Theorem 5.2.3 from Motik [2006].

Lemma 5 *Let $\mathcal{K}$ be a $\mathcal{SROIQ}$ knowledge base. Then $\Omega(\mathcal{K})$ is a concept-model-conservative encoding of $\mathcal{K}$.*

Proof Let $\mathcal{I}$ be a model of $\mathcal{K}$. We define the interpretation $\mathcal{I}'$ by extending $\mathcal{I}$ to interpret the fresh atomic concepts Q_p^C. Let i_R and δ be the initial state and transition function, respectively, of the automaton translation $\mathcal{B}_R$ of role R with respect to $\mathcal{K}$, and let $\forall R.C$ be a concept in $\mathcal{K}^+$. Then let $\mathcal{I}'$ be the smallest extension of $\mathcal{I}$ such that

$$\begin{aligned}(Q_{p'}^C)^{\mathcal{I}'} = &\{x \mid x \in (Q_p^C)^{\mathcal{I}'} \text{ and } p' \in \delta_R(p, \epsilon)\} \cup \\ &\{y \mid \exists x : x \in (Q_p^C)^{\mathcal{I}'}, \langle x, y\rangle \in (R')^{\mathcal{I}'}, \text{ and } p' \in \delta_R(p, R')\} \cup \\ &(\forall R.C)^{\mathcal{I}} \text{ if } p' = i_R\end{aligned}$$

for each concept $Q_{p'}^C$. Clearly, $\mathcal{I}'$ and $\mathcal{I}$ coincide on $(\emptyset, N_C, N_I)$. We now show that $\mathcal{I}'$ is a model of $\Omega(\mathcal{K})$.

It is clear from the definition of $\mathcal{I}'$ that if $y \in (Q_f^C)^{\mathcal{I}'}$ for f a final state of the automata B_R, then there exist elements $x_0, ..., x_n$ of $\triangle^{\mathcal{I}'}$ and roles $R_1, ...R_n$ such that $x_0 \in (\forall R.C)^{\mathcal{I}}$ and $\langle x_{n-1}, x_n\rangle \in R_n^{\mathcal{I}'}$, and $R_1...R_n \in L(B_R)$. By Lemma 1, it is thus the case that $\langle x_0, y\rangle \in R^{\mathcal{I}}$, and since $x_0 \in (\forall R.C)^{\mathcal{I}}$ it is also the case that $y \in C^{\mathcal{I}}$. Thus $\mathcal{I}'$ satisfies all axioms of the form $Q_f^C \sqsubseteq C$ added in the construction of $\Omega(\mathcal{K})$.

Further, $\mathcal{I}'$ clearly satisfies the additional axioms of $\Omega(\mathcal{K})$ of the forms $\forall R.C \sqsubseteq Q_{i_R}^C$, $Q_p^C \sqsubseteq \forall R'.Q_{p'}^C$, and $Q_p^C \sqsubseteq Q_{p'}^C$ by definition. Finally, $\mathcal{I}'$ differs from $\mathcal{I}$ only in its interpretation of concepts not occurring in $\mathcal{K}$, so $\mathcal{I}'$ satisfies all axioms from $\mathcal{K}$. $\mathcal{I}'$ is thus a model of $\Omega(\mathcal{K})$, and $\mathcal{I}'$ and $\mathcal{I}$ coincide on $(\emptyset, N_C, N_I)$.

Let $\mathcal{I}' = (\triangle^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ be a model of $\Omega(\mathcal{K}) = (\mathcal{T}', \mathcal{A}, \mathcal{R}')$. We define the interpretation $\mathcal{I} = (\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$ as follows:

- $\triangle^{\mathcal{I}} = \triangle^{\mathcal{I}'}$
- $a^{\mathcal{I}} = a^{\mathcal{I}'}$ for every individual a
- $A^{\mathcal{I}} = A^{\mathcal{I}'}$ for every atomic concept A
- $R^{\mathcal{I}} = R^{\mathcal{I}'} \cup \{\langle x, y\rangle \mid \langle x, y\rangle \in R_1^{\mathcal{I}'} \circ ... \circ R_n^{\mathcal{I}'}$ and $R_1 ... R_n \in L(\mathcal{B}_R)\}$ for each atomic role R

Clearly, $\mathcal{I}'$ and $\mathcal{I}$ coincide on $(\emptyset, N_C, N_I)$. We now show that $\mathcal{I}$ is a model of $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{R})$.

By the definition of $\mathcal{I}$ and Lemma 2, $\mathcal{I}$ clearly satisfies all complex role inclusions in $\mathcal{R}$. Further, for each simple role S it is the case that $S^{\mathcal{I}} = S^{\mathcal{I}'}$. Since $\Omega(\mathcal{K})$ contains all simple role inclusion, role disjointness, and irreflexivity axioms of $\mathcal{R}$, and $\mathcal{I}'$ is a model of $\Omega(\mathcal{K})$, then $\mathcal{I}$ also satisfies all simple role inclusion, role disjointness, and irreflexivity axioms of $\mathcal{R}$. It is thus the case that $\mathcal{I}$ is a model of $\mathcal{R}$.

Let $\prec$ be a relation on the concepts $({}^{+}\mathcal{K})$ such that $C \prec D$ iff C or $\mathsf{nnf}(\neg C)$ occur in D. We next show by induction on $\prec$ that for every concept $C \in ({}^{+}\mathcal{K})$, it is the case that $C^{\mathcal{I}'} \subseteq C^{\mathcal{I}}$. For the base case where C is an atomic concept, a nominal, a local reflexivity axiom, or a negation of any of these, the claim follows immediately from the definition of $\mathcal{I}$. For the inductive step, we consider the different forms that $\mathsf{nnf}(C)$ can take:

- For $C = D_1 \sqcap D_2$, let α be some element of $\triangle^{\mathcal{I}'}$ such that $\alpha \in (D_1 \cap D_2)^{\mathcal{I}'}$. Then $\alpha \in D_1^{\mathcal{I}'}$ and $\alpha \in D_2^{\mathcal{I}'}$. By the inductive hypothesis, it is thus the case that $\alpha \in D_1^{\mathcal{I}}$ and $\alpha \in D_2^{\mathcal{I}}$, so $\alpha \in (D_1 \cap D_2)^{\mathcal{I}}$.
- For $C = D_1 \sqcup D_2$, let α be some element of $\triangle^{\mathcal{I}'}$ such that $\alpha \in (D_1 \cup D_2)^{\mathcal{I}'}$. Then $\alpha \in D_1^{\mathcal{I}'}$ or $\alpha \in D_2^{\mathcal{I}'}$. By the inductive hypothesis, if $\alpha \in D_1^{\mathcal{I}'}$ then $\alpha \in D_1^{\mathcal{I}}$, and if $\alpha \in D_2^{\mathcal{I}'}$ then $\alpha \in D_2^{\mathcal{I}}$. In either case, $\alpha \in (D_1 \cup D_2)^{\mathcal{I}}$.

- For $C = \exists R.D$, let α be some element of $\triangle^{\mathcal{I}'}$ such that $\alpha \in (\exists R.D)^{\mathcal{I}'}$. Then there exists some β such that $\langle \alpha, \beta \rangle \in R^{\mathcal{I}'}$ and $\beta \in D^{\mathcal{I}'}$. Then $\langle \alpha, \beta \rangle \in R^{\mathcal{I}}$ by the definition of $\mathcal{I}$, and $\beta \in D^{\mathcal{I}}$ by the inductive hypothesis, so $\alpha \in (\exists R.D)^{\mathcal{I}}$.

- For $C = \forall R.D$, let α be some element of $\triangle^{\mathcal{I}'}$ such that $\alpha \in (\forall R.D)^{\mathcal{I}'}$. Suppose there were some element β such that $\langle \alpha, \beta \rangle \in R^{\mathcal{I}}$ and $\beta \notin D^{\mathcal{I}}$. Then by the inductive hypothesis, $\beta \notin D^{\mathcal{I}'}$, and since $\alpha \in (\forall R.D)^{\mathcal{I}'}$, it must be the case that $\langle \alpha, \beta \rangle \notin R^{\mathcal{I}'}$. Then there must exist roles $R_1, ..., R_n$ such that $\langle \alpha, \beta \rangle \in R_1^{\mathcal{I}'} \circ ... \circ R_n^{\mathcal{I}'}$ and $R_1 ... R_n \in L(\mathcal{B}_R)$. But then, by Lemma 4, it is the case that $\beta \in D^{\mathcal{I}'}$, which is a contradiction. No such β exists, so $\alpha \in (\forall R.D)^{\mathcal{I}}$.

- For $C = \geq n\, R.D$, let α be some element of $\triangle^{\mathcal{I}'}$ such that $\alpha \in (\geq n\, R.D)^{\mathcal{I}'}$. Then there exist elements $\beta_1, ..., \beta_n$ such that $\langle \alpha, \beta_i \rangle \in R^{\mathcal{I}'}$ and $\beta_i \in D^{\mathcal{I}'}$ for $1 \leq i \leq n$. Due to the fact that $R^{\mathcal{I}'} \subseteq R^{\mathcal{I}}$ and the inductive hypothesis, it is thus the case that $\langle \alpha, \beta_i \rangle \in R^{\mathcal{I}}$ and $\beta_i \in D^{\mathcal{I}}$ for $1 \leq i \leq n$, so $\alpha \in (\geq n\, R.D)^{\mathcal{I}}$.

It is thus the case that $C^{\mathcal{I}'} \subseteq C^{\mathcal{I}}$ for each concept $C \in ({}^{+}\mathcal{K})$. The interpretation $\mathcal{I}'$ is a model of $\mathcal{A}$ and $R^{\mathcal{I}'} \subseteq R^{\mathcal{I}}$ for each role occurring in $\mathcal{K}$, so clearly $\mathcal{I}$ satisfies all concept and role assertions of $\mathcal{A}$. Further, $a^{\mathcal{I}} = a^{\mathcal{I}'}$ for each individual occurring in $\mathcal{K}$, so $\mathcal{I}$ also satisfies all equality and inequality assertions in $\mathcal{A}$, and thus $\mathcal{I}$ is a model of $\mathcal{A}$.

Finally, for each GCI $C \sqsubseteq D \in \mathcal{T}$, it is the case that $C \sqsubseteq D \in \mathcal{T}'$, so $C^{\mathcal{I}'} \cap \triangle^{\mathcal{I}'} \setminus D^{\mathcal{I}'} = \emptyset$ and $\triangle^{\mathcal{I}'} \subseteq (\neg C \sqcup D)^{\mathcal{I}'}$. Since $(\neg C \sqcup D)^{\mathcal{I}'} \subseteq (\neg C \sqcup D)^{\mathcal{I}}$ and $\triangle^{\mathcal{I}} = \triangle^{\mathcal{I}'}$, it is the case that $\triangle^{\mathcal{I}} \subseteq (\neg C \sqcup D)^{\mathcal{I}}$, so $\mathcal{I}$ satisfies $C \sqsubseteq D$. We thus have that $\mathcal{I}$ is a model of $\mathcal{T}$, and $\mathcal{I}$ is a model of $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{R})$. □

Finally, we describe the complexity of our transformation:

Lemma 6 *Let $\mathcal{K}$ be a $\mathcal{SROIQ}$ knowledge base. Then $\Omega(\mathcal{K})$ can be computed in time exponential in $|\mathcal{K}|$, and $|\Omega(\mathcal{K})|$ is exponential in $|\mathcal{K}|$. Further, if all complex role*

inclusion axioms of $\mathcal{K}$ *are of the form* $RR \sqsubseteq R$*, then* $\Omega(\mathcal{K})$ *can be computed in time polynomial in* $|\mathcal{K}|$*, and* $|\Omega(\mathcal{K})|$ *is polynomial in* $|\mathcal{K}|$*.*

Proof This lemma is an obvious consequence of Lemma 3, Definition 8, the observation that $d_{\mathcal{K}}$ is bounded by the number of axioms in $\mathcal{K}$, and the observation that $d_{\mathcal{K}} = 0$ if all complex role inclusion axioms in $\mathcal{K}$ are of the form $RR \sqsubseteq R$. □

Such an exponential blowup when converting $\mathcal{SROIQ}$ to $\mathcal{ALCHOIQ}^+$ is unavoidable in the worst case: $\mathcal{SROIQ}$ is N2ExpTime-hard while $\mathcal{ALCHOIQ}^+$ (a variant of $\mathcal{SHOIQ}$) is NExpTime-complete [Kazakov, 2008].

As noted in Definition 4, there is no distinction between simple and non-simple roles in $\mathcal{ALCHOIQ}^+$. Hence, in the rest of this chapter we assume that all roles are simple unless otherwise stated and, without loss of generality, we treat $\exists R.B$ as a syntactic shortcut for $\geq 1\, R.B$.

5.1.2 Normalization

Before translation into a set of DL-clauses, a knowledge base is first brought into a *normalized* form. This is done in order to make all negations explicit, and to ensure that the resulting DL-clauses are compatible with blocking.

To understand the first issue, consider the axiom $\neg A \sqsubseteq \neg(\exists R.\exists R.\exists R.B)$. Converting this axiom into DL-clauses is not straightforward because of the implicit negations; for example, the concept A is seemingly negated but, due to the negation implicit in the implication, A actually occurs positively in the axiom. Therefore, we replace this axiom with the following equivalent axiom. This makes all negations explicit, so the result can be easily translated into a DL-clause.

$$\top \sqsubseteq A \sqcup \forall R.\forall R.\forall R.\neg B \quad \rightsquigarrow \quad R(x, y_1) \wedge R(y_1, y_2) \wedge R(y_2, y_3) \wedge B(y_3) \rightarrow A(x) \tag{5.1}$$

To understand the second issue, consider the knowledge base $\mathcal{K}_{10}$, consisting of an ABox $\mathcal{A}_{10}$ and a TBox that corresponds to the set of DL-clauses $\mathcal{C}_{10}$.

$$\mathcal{A}_{10} = \{\ \neg A(a),\ B(a)\ \}$$

$$\mathcal{C}_{10} = \{\ R(x,y_1) \wedge R(y_1,y_2) \wedge R(y_2,y_3) \wedge B(y_3) \rightarrow A(x), \quad B(x) \rightarrow \exists R.B(x)\ \}$$

By applying the rules from Chapter 4, our algorithm constructs on $\mathcal{K}_{10}$ the ABox shown in Figure 5.2. According to the definition of blocking introduced in Definition 13,[1] c is now blocked by b; furthermore, no rule is applicable to the ABox, so the algorithm terminates, leading us to believe that $\mathcal{K}_{10}$ is satisfiable. The ABox, however, does not represent a model of $\mathcal{K}_{10}$: if we expand $\exists R.B(c)$ into $R(c,d)$ and $B(d)$, by the first DL-clause in $\mathcal{C}_{10}$ we can derive $A(a)$, which then contradicts $\neg A(a)$. This problem arises because the antecedent of the first DL-clause in $\mathcal{C}_{10}$ checks for a path of three R-successors, whereas the pairwise blocking condition ensures only that all paths of length two are fully constructed. Intuitively, the antecedents of each DL-clause should check for paths that "fit" into the fully-constructed model fragments. We can ensure this by renaming complex concepts into simpler ones. Thus, we transform the culprit DL-clause into the following ones, which check only for paths of length one.

$$\top \sqsubseteq A \sqcup \forall R.\neg Q_1 \qquad \rightsquigarrow \qquad R(x,y) \wedge Q_1(y) \rightarrow A(x) \tag{5.2}$$

$$\top \sqsubseteq Q_1 \sqcup \forall R.\neg Q_2 \qquad \rightsquigarrow \qquad R(x,y) \wedge Q_2(y) \rightarrow Q_1(x) \tag{5.3}$$

$$\top \sqsubseteq Q_2 \sqcup \forall R.\neg B \qquad \rightsquigarrow \qquad R(x,y) \wedge B(y) \rightarrow Q_2(x) \tag{5.4}$$

The application of these DL-clauses to the ABox shown in Figure 5.2 would additionally derive $Q_2(a)$, $Q_2(b)$, and $Q_1(a)$, so c would not be blocked. The calculus would then expand $\exists R.B(c)$ and discover a contradiction.

To formalize these ideas, we define a normalized form of DL knowledge bases.

[1] The version of blocking introduced in Definition 13 differs from the one presented in Section 3.3 in that the concept label $\mathcal{L}_{\mathcal{A}}(s)$ of an individual s consists only of atomic concepts A such that $A(s) \in \mathcal{A}$.

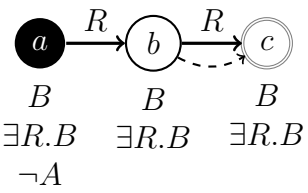


Figure 5.2: Incorrect Blocking due to Lack of Normalization

Definition 9 (Normalized Form of GCI, TBox, and ABox) A GCI is *normalized* if it is of the form $\top \sqsubseteq \bigsqcup_{i=1}^{n} C_i$, where each C_i is of the form B, $\{a\}$, $\forall R.B$, $\exists R.\mathsf{Self}$, $\neg\exists R.\mathsf{Self}$, $\geq n\, R.B$, or $\leq n\, R.B$, for B a literal concept, R a role, and n a nonnegative integer.

A TBox $\mathcal{T}$ is *normalized* if each GCI in it is normalized. An ABox $\mathcal{A}$ is *normalized* if each concept assertion in $\mathcal{A}$ contains only an atomic concept, each role assertion in $\mathcal{A}$ contains only an atomic role, and $\mathcal{A}$ contains at least one assertion. An $\mathcal{ALCHOIQ}^+$ knowledge base $\mathcal{K} = (\mathcal{R}, \mathcal{T}, \mathcal{A})$ is *normalized* if $\mathcal{T}$ and $\mathcal{A}$ are normalized. △

The following transformation can be used to normalize a knowledge base.

Definition 10 (Normalization) For an $\mathcal{ALCHOIQ}^+$ knowledge base $\mathcal{K}$, the knowledge base $\Delta(\mathcal{K})$ is computed as shown in Table 5.2. △

Normalization can be seen as a variant of the well-known structural transformation [Plaisted and Greenbaum, 1986; Nonnengart and Weidenbach, 2001]. An application of the structural transformation to (5.1) would replace each complex subconcept with a positive atomic concept, eventually producing $\top \sqsubseteq A \sqcup \forall R.Q_1$. This axiom cannot be translated into a Horn DL-clause, whereas (5.1) can; thus, the standard structural transformation can destroy Horn-ness. To prevent this, we introduce the function $\mathsf{pos}(C)$ (cf. Table 5.2) that returns false if the clausification of C does not require adding atoms into the consequent of a DL-clause. We then replace an occurrence of a concept C in a concept D with a negative literal concept $\neg Q_C$ if $\mathsf{pos}(C) = \mathsf{false}$,

Table 5.2: The Functions Used in the Normalization

$$
\begin{aligned}
\Delta(\mathcal{K}) &= \{\top(a)\} \cup \bigcup_{\alpha \in \mathcal{R} \cup \mathcal{A}} \Delta(\alpha) \cup \bigcup_{C_1 \sqsubseteq C_2 \in \mathcal{T}} \Delta(\top \sqsubseteq \mathsf{nnf}(\neg C_1 \sqcup C_2)) \\
\Delta(\top \sqsubseteq \mathbf{C} \sqcup C') &= \Delta(\top \sqsubseteq \mathbf{C} \sqcup \alpha_{C'}) \cup \bigcup_{1 \le i \le n} \Delta(\top \sqsubseteq \dot{\neg}\alpha_{C'} \sqcup C_i) \\
&\quad \text{for } C' \text{ of the form } C' = C_1 \sqcap \ldots \sqcap C_n \text{ and } n \ge 2 \\
\Delta(\top \sqsubseteq \mathbf{C} \sqcup \forall R.D) &= \Delta(\top \sqsubseteq \mathbf{C} \sqcup \forall R.\alpha_D) \cup \Delta(\top \sqsubseteq \dot{\neg}\alpha_D \sqcup D) \\
\Delta(\top \sqsubseteq \mathbf{C} \sqcup \ge n\, R.D) &= \Delta(\top \sqsubseteq \mathbf{C} \sqcup \ge n\, R.\alpha_D) \cup \Delta(\top \sqsubseteq \dot{\neg}\alpha_D \sqcup D) \\
\Delta(\top \sqsubseteq \mathbf{C} \sqcup \le n\, R.D) &= \Delta(\top \sqsubseteq \mathbf{C} \sqcup \le n\, R.\dot{\neg}\alpha_{\dot{\neg}D}) \cup \Delta(\top \sqsubseteq \dot{\neg}\alpha_{\dot{\neg}D} \sqcup \dot{\neg}D) \\
\Delta(\top \sqsubseteq \mathbf{C} \sqcup \neg\{s\}) &= \begin{cases} \bot & \text{if } \mathbf{C} \text{ is empty,} \\ \Delta(\mathbf{C}(s)) & \text{otherwise.} \end{cases} \\
\Delta(D(s)) &= \{\alpha_D(s)\} \cup \Delta(\top \sqsubseteq \dot{\neg}\alpha_D \sqcup \mathsf{nnf}(D)) \\
\Delta(R^-(s,t)) &= \{R(t,s)\} \\
\Delta(\beta) &= \{\beta\} \text{ for any other axiom } \beta
\end{aligned}
$$

$$
\alpha_C = \begin{cases} Q_C & \text{if } \mathsf{pos}(C) = \mathsf{true} \\ \neg Q_C & \text{if } \mathsf{pos}(C) = \mathsf{false} \end{cases}, \text{ where } Q_C \text{ is a fresh atomic concept unique for } C
$$

$$
\begin{aligned}
\mathsf{pos}(\top) &= \mathsf{false} & \mathsf{pos}(\bot) &= \mathsf{false} \\
\mathsf{pos}(A) &= \mathsf{true} & \mathsf{pos}(\neg A) &= \mathsf{false} \\
\mathsf{pos}(\{s\}) &= \mathsf{true} & \mathsf{pos}(\neg\{s\}) &= \mathsf{false} \\
\mathsf{pos}(\exists R.\mathsf{Self}) &= \mathsf{true} & \mathsf{pos}(\neg\exists R.\mathsf{Self}) &= \mathsf{false} \\
\mathsf{pos}(C_1 \sqcap C_2) &= \mathsf{pos}(C_1) \vee \mathsf{pos}(C_2) & \mathsf{pos}(C_1 \sqcup C_2) &= \mathsf{pos}(C_1) \vee \mathsf{pos}(C_2) \\
\mathsf{pos}(\forall R.C_1) &= \mathsf{pos}(C_1) & \mathsf{pos}(\le n\, R.C_1) &= \begin{cases} \mathsf{pos}(\dot{\neg}C_1) & \text{if } n = 0 \\ \mathsf{true} & \text{otherwise} \end{cases} \\
\mathsf{pos}(\ge n\, R.C_1) &= \mathsf{true}
\end{aligned}
$$

Note: A is an atomic concept, $C_{(i)}$ are arbitrary concepts, $\mathbf{C}$ is a possibly empty disjunction of arbitrary concepts, D is not a literal concept, and a is a fresh individual. Note that $\sqcup$ is commutative, so C' in $\mathbf{C} \sqcup C'$ is not necessarily the right-most disjunct.

and with a positive literal concept Q_C if $\mathsf{pos}(C) = \mathsf{true}$. Special care must be taken when replacing a concept D in a concept $\le n\, R.D$: since D occurs in $\le n\, R.D$ under an implicit negation, we replace D with $\dot{\neg}\alpha_{\dot{\neg}D}$ in order to preserve Horn-ness. On a Horn knowledge base $\mathcal{K}$, normalization performs the same replacements as the one presented by Hustadt *et al.* [2005], so $\Delta(\mathcal{K})$ is a Horn knowledge base as well.

Lemma 7 *The following properties hold for each $\mathcal{ALCHOIQ}^+$ knowledge base $\mathcal{K}$ and the corresponding knowledge base $\Delta(\mathcal{K})$:*

- *$\Delta(\mathcal{K})$ is a model-conservative encoding of $\mathcal{K}$;*

- $\Delta(\mathcal{K})$ *is normalized; and*
- $\Delta(\mathcal{K})$ *can be computed in time polynomial in* $|\mathcal{K}|$.

Proof (*Sketch*) Since our transformation can be seen a syntactic variant of the structural transformation, the proof that all models of $\Delta(\mathcal{K})$ coincide with those of $\mathcal{K}$ on symbols which appear in $\mathcal{K}$ is analogous to the proofs given by Plaisted and Greenbaum [1986] and Nonnengart and Weidenbach [2001], so we omit it. For the second claim, note that Δ essentially rewrites each GCI into a form $\top \sqsubseteq \bigsqcup_{i=1}^{n} C_i$ and then keeps replacing nested subconcepts of C_i until the GCI becomes normalized; it adds $\top(a)$ to the ABox so that the ABox is not empty; and it replaces all inverse role assertions with equivalent assertions on the atomic roles. Thus, $\Delta(\mathcal{K})$ is normalized. Finally, each occurrence of a concept in $\mathcal{K}$ can be replaced with a new atomic concept at most once, and all necessary syntactic transformations can be performed in polynomial time, so $\Delta(\mathcal{K})$ can be computed in polynomial time. □

5.1.3 Translation into DL-Clauses

We now introduce the notion of HT-clauses—syntactically restricted DL-clauses on which our hypertableau calculus is guaranteed to terminate. In the rest of this paper, we often use the function ar, which, given a role R and variables or constants s and t, returns an atom that is semantically equivalent to $R(s,t)$ but that contains an atomic role; that is,

$$\mathsf{ar}(R,s,t) = \begin{cases} R(s,t) & \text{if } R \text{ is an atomic role} \\ S(t,s) & \text{if } R \text{ is an inverse role and } R = S^- \end{cases} .$$

Definition 11 (HT-Clause) We assume that, for each individual a, the set of atomic concepts N_C contains a unique *nominal guard concept* which we denote as O_a; furthermore, we assume that nominal guard concepts do not occur in any input knowledge base. The intuition behind nominal guard concepts is explained in Section 4.4.

An *annotated equality* is an atom of the form $s \approx t\,@^{u}_{\leq n\,S.B}$, where s, t, and u are constants or variables, n is a nonnegative integer, S is a role, and B is a literal concept; the part $@^{u}_{\leq n\,S.B}$ of the atom is called the *annotation*. This atom is semantically equivalent to $s \approx t$. As explained in Section 3.5, annotations are only used to ensure termination of the hypertableau phase.

An *HT-clause* is a DL-clause r of the following form, for $m \geq 0$ and $n \geq 0$:

$$U_1 \wedge \ldots \wedge U_m \rightarrow V_1 \vee \ldots \vee V_n \tag{5.5}$$

Furthermore, it must be possible to separate the variables into a *center variable* x, a set of *branch variables* y_i, and a set of *nominal variables* z_j such that the following properties hold, for A an atomic concept, B an atomic concept which is not a nominal guard concept, O_a a nominal guard concept, R an atomic role, and S a role.

1. Each atom in the antecedent of r is of the form $A(x)$, $R(x,x)$, $R(x,y_i)$, $R(y_i,x)$, $A(y_i)$, or $A(z_j)$.

2. Each atom in the consequent of r is of the form $B(x)$, $\geq h\,S.B(x)$, $B(y_i)$, $R(x,x)$, $R(x,y_i)$, $R(y_i,x)$, $R(x,z_j)$, $R(z_j,x)$, $x \approx z_j$, or $y_i \approx y_j\,@^{x}_{\leq h\,S.B}$.

3. Each y_i occurs in the antecedent of r in an atom of the form $R(x,y_i)$ or $R(y_i,x)$.

4. Each z_j occurs in the antecedent of r in an atom of the form $O_a(z_j)$.

5. Each equality $y_i \approx y_j\,@^{x}_{\leq h\,S.A}$ in the consequent of r occurs in a subclause of r of the form (5.6) where $y^1, \ldots, y^{h+1}$ are branch variables such that no y^k with $1 \leq k \leq h+1$ occurs elsewhere in r.

$$\ldots \bigwedge_{k=1}^{h+1} [\mathsf{ar}(S,x,y^k) \wedge A(y^k)] \ldots \rightarrow \ldots \bigvee_{1 \leq k < \ell \leq h+1} y^k \approx y^\ell\,@^{x}_{\leq h\,S.A} \ldots \tag{5.6}$$

6. Each equality $y_i \approx y_j \, @^x_{\leq h\, S.\neg A}$ in the consequent of r occurs in a subclause of r of the form (5.7) where $y^1, \ldots, y^{h+1}$ are branch variables such that no y^k with $1 \leq k \leq h+1$ occurs elsewhere in r.

$$\ldots \bigwedge_{k=1}^{h+1} \mathsf{ar}(S, x, y^k) \ldots \rightarrow \ldots \bigvee_{k=1}^{h+1} A(y^k) \vee \bigvee_{1 \leq k < \ell \leq h+1} y^k \approx y^\ell \, @^x_{\leq h\, S.\neg A} \ldots \tag{5.7}$$

△

HT-clauses are more general than what is strictly needed to capture $\mathcal{ALCHOIQ}^+$ knowledge bases. For example, HT-clauses of the form $R(x,y) \wedge A(y) \rightarrow S(x,y)$ express a form of *relativized* role inclusions, and *safe* role expressions can be captured by HT-clauses of the form $R(x,y) \wedge S(y,x) \rightarrow U(x,y) \vee T(y,x)$ [Tobies, 2001].

We now show how to transform a normalized $\mathcal{ALCHOIQ}^+$ knowledge base into a set of HT-clauses.

Definition 12 (Clausification) The *clausification* of a normalized $\mathcal{ALCHOIQ}^+$ knowledge base $\mathcal{K} = (\mathcal{R}, \mathcal{T}, \mathcal{A})$ is the pair $\Xi(\mathcal{K}) = (\Xi_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$ in which $\Xi_{\mathcal{TR}}(\mathcal{K})$ is a set of DL-clauses and $\Xi_{\mathcal{A}}(\mathcal{K})$ is an ABox, both obtained as shown in Table 5.3, where each $Q_{\neg A}$ is a fresh atomic concept not occurring in $\mathcal{K}$. △

Lemma 8 *Let $\mathcal{K} = (\mathcal{R}, \mathcal{T}, \mathcal{A})$ be a normalized $\mathcal{ALCHIQ}$ knowledge base. Then the clausification $\Xi(\mathcal{K}) = (\Xi_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$ is a model-conservative encoding of $\mathcal{K}$, and $\Xi_{\mathcal{TR}}(\mathcal{K})$ contains only HT-clauses.*

Proof We begin by showing that $\Xi_{\mathcal{TR}}(KB)$ contains only HT-clauses. By inspecting Table 5.3, it is easy to see that $\Xi_{\mathcal{R}}(\mathcal{R})$ contains only clauses with simple role atoms containing a center variable and at most one branch variable, which also occurs in the antecedent of the clause; every clause r in $\Xi_{\mathcal{R}}(\mathcal{R})$ is clearly an HT-clause.

For $\Xi_{\mathcal{T}}(\mathcal{T})$, the lhs and rhs functions are used to construct the antecedent and consequent of each clause, respectively. We consider each of the properties of DL-clauses given by Definition 11.

Table 5.3: Translation of a Normalized Knowledge Base to HT-Clauses

$$
\begin{array}{rcl}
\Xi_{\mathcal{T}}(\mathcal{T}) & = & \{\bigwedge_{i=1}^{n} \mathsf{lhs}(C_i) \to \bigvee_{i=1}^{n} \mathsf{rhs}(C_i) \mid \text{ for each } \top \sqsubseteq \bigsqcup_{i=1}^{n} C_i \text{ in } \mathcal{T}\} \\
\Xi_{\mathcal{R}}(\mathcal{R}) & = & \{\mathsf{ar}(R,x,y) \to \mathsf{ar}(S,x,y) \mid \text{for each } R \sqsubseteq S \text{ in } \mathcal{R}\} \cup \\
 & & \{\mathsf{ar}(S_1,x,y) \wedge \mathsf{ar}(S_2,x,y) \to \bot \mid \text{for each } \mathsf{Dis}(S_1,S_2) \in \mathcal{R}\} \cup \\
 & & \{\top \to \mathsf{ar}(R,x,x) \mid \text{for each } \mathsf{Ref}(R) \in \mathcal{R}\} \cup \\
 & & \{\mathsf{ar}(S,x,x) \to \bot \mid \text{for each } \mathsf{Irr}(S) \in \mathcal{R}\} \cup \\
 & & \{\mathsf{ar}(R,x,y) \to \mathsf{ar}(R,y,x) \mid \text{for each } \mathsf{Sym}(R) \in \mathcal{R}\} \cup \\
 & & \{\mathsf{ar}(S,x,y) \wedge \mathsf{ar}(S,y,x) \to \bot \mid \text{for each } \mathsf{Asy}(S) \in \mathcal{R}\} \\
\Xi_{\mathcal{A}}(\mathcal{K}) & = & \{R(a,b) \mid R(a,b) \in \mathcal{A}\} \cup \\
 & & \{A(a) \mid \text{for each atomic concept } A \text{ such that } A(a) \in \mathcal{A}\} \cup \\
 & & \{Q_{\neg A}(a) \mid \text{for each atomic concept } A \text{ such that } \neg A(a) \in \mathcal{A}\} \cup \\
 & & \{O_a(a) \mid \text{for each } \{a\} \text{ occurring in } \mathcal{K}\} \\
\Xi_{\mathcal{TR}}(\mathcal{K}) & = & \Xi_{\mathcal{T}}(\mathcal{T}) \cup \Xi_{\mathcal{R}}(\mathcal{R}) \cup \{A(x) \wedge Q_{\neg A}(x) \to \bot \mid \\
 & & \qquad \text{for each } Q_{\neg A} \text{ occurring in } \Xi_{\mathcal{T}}(\mathcal{T}) \text{ or } \Xi_{\mathcal{A}}(\mathcal{K})\}
\end{array}
$$

Note: Whenever $\mathsf{lhs}(C_i)$ or $\mathsf{rhs}(C_i)$ is undefined, it is omitted in the HT-clause.

C	$\mathsf{lhs}(C)$	$\mathsf{rhs}(C)$
A		$A(x)$
$\neg A$	$A(x)$	
$\{a\}$	$O_a(z_C)$	$x \approx z_C$
$\geq n\, R.A$		$\geq n\, R.A(x)$
$\geq n\, R.\neg A$		$\geq n\, R.Q_{\neg A}(x)$
$\exists R.\mathsf{Self}$		$\mathsf{ar}(R,x,x)$
$\neg\exists R.\mathsf{Self}$	$\mathsf{ar}(R,x,x)$	
$\forall R.A$	$\mathsf{ar}(R,x,y_C)$	$A(y_C)$
$\forall R.\neg A$	$\mathsf{ar}(R,x,y_C) \wedge A(y_C)$	
$\leq n\, R.A$	$\bigwedge_{i=1}^{n+1} [\mathsf{ar}(R,x,y_C^i) \wedge A(y_C^i)]$	$\bigvee_{1 \leq i < j \leq n+1} y_C^i \approx y_C^j \,@^x_{\leq n\, R.A}$
$\leq n\, R.\neg A$	$\bigwedge_{i=1}^{n+1} \mathsf{ar}(R,x,y_C^i)$	$\bigvee_{i=1}^{n+1} A(y_C^i) \vee \bigvee_{1 \leq i < j \leq n+1} y_C^i \approx y_C^j \,@^x_{\leq n\, R.\neg A}$

Note: Each $y_C^{(i)}$ and z_C is a fresh variable unique for C (and i).

- The lhs and rhs functions produce only atoms conforming to conditions 1 and 2 of Definition 11.

- In the three cases in which $\mathsf{rhs}(C)$ produces a branch variable y, the function $\mathsf{lhs}(C)$ also produces a role atom $\mathsf{ar}(R, x, y)$, so each clause conforms to condition 3 of Definition 11.

- A nominal variable z is introduced only by applying rhs to a concept of the form $\{a\}$. In this case, $\mathsf{lhs}(\{a\}) = O_a(z)$, so each clause conforms to condition 4 of Definition 11.

- An equality $y_i \approx y_j \, @^x_{\leq h\, S.A}$ is introduced only by applying rhs to a concept of the form $\leq n\, R.A$. In this case, $\mathsf{lhs}(\leq n\, R.A)$ and $\mathsf{rhs}(\leq n\, R.A)$ form a subclause of form (5.6), so each clause conforms to condition 5 of Definition 11.

- An equality $y_i \approx y_j \, @^x_{\leq h\, S.\neg A}$ is introduced only by applying rhs to a concept of the form $\leq n\, R.\neg A$. In this case, $\mathsf{lhs}(\leq n\, R.\neg A)$ and $\mathsf{rhs}(\leq n\, R.\neg A)$ form a subclause of form (5.7), so each clause conforms to condition 6 of Definition 11.

We next show that $\Xi(\mathcal{K})$ is a model-conservative encoding of $\mathcal{K}$.

The following equivalences between DL concepts and first-order formulae are well known [Borgida, 1996]:

$$\begin{aligned}
\forall R.B(x) &\equiv \forall y : \neg R(x, y) \vee B(y) \\
\leq n\, R.B(x) &\equiv \forall y_1, \ldots, y_{n+1} : \bigwedge_{1 \leq i \leq n+1} [R(x, y_i) \wedge B(y_i)] \rightarrow \bigvee_{1 \leq i < j \leq n+1} y_i \approx y_j \\
\{a\}(x) &\equiv x \approx a
\end{aligned}$$

Let $\Xi'_{\mathcal{TR}}(\mathcal{K})$ be the set of HT-clauses defined just like $\Xi_{\mathcal{TR}}(\mathcal{K})$, but with the difference that $\mathsf{lhs}(\{a\}) = \top$ and $\mathsf{rhs}(\{a\}) = x \approx a$. Then, $(\Xi'_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$ is obtained from $\mathcal{K}$

by replacing concepts of the form $\forall R.B$, $\leq n\, R.B$ and $\{a\}$ with the equivalent first-order formulae, making the semantics of role axioms explicit, and introducing the atomic concept names $Q_{\neg A}$ in place of the literals $\neg A$. The clausified knowledge base $(\Xi'_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$ is clearly a model-conservative encoding of $\mathcal{K}$. We now show that $(\Xi_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$ is a model-conservative encoding of $(\Xi'_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$.

($\Rightarrow$) Each model $\mathcal{I}'$ of $(\Xi'_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$ is extended to a model $\mathcal{I}$ of $(\Xi_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$ by setting $O_a^{\mathcal{I}} = \{a^{\mathcal{I}'}\}$ for each nominal guard concept O_a.

($\Leftarrow$) Each model $\mathcal{I}$ of $\Xi(\mathcal{K})$ is a model of $(\Xi'_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$: for each $\gamma \in \Xi'_{\mathcal{TR}}(\mathcal{K})$, we have $\delta \in \Xi_{\mathcal{TR}}(\mathcal{K})$ and $O_{a_k}(a_k) \in \Xi_{\mathcal{A}}(\mathcal{K})$, where γ and δ are of the form shown below.

$$\gamma = \bigwedge U_i \rightarrow \bigvee V_j \vee \bigvee_{k=1}^{n} x_k \approx a_k$$

$$\delta = \bigwedge U_i \wedge \bigwedge_{k=1}^{n} O_{a_k}(z_{\{a_k\}}) \rightarrow \bigvee V_j \vee \bigvee_{k=1}^{n} x_k \approx z_{\{a_k\}}$$

Now if the disjunction $\bigvee_{k=1}^{n} x_k \approx a_k$ in some γ were not true in $\mathcal{I}$ for some values of $x_1, \ldots, x_n$, then clearly δ would not be true in $\mathcal{I}$ for the same values of $x_1, \ldots, x_n$. $\square$

5.2 The Hypertableau Calculus for HT-Clauses

We now present the hypertableau calculus for deciding the satisfiability of an ABox $\mathcal{A}$ and a set of HT-clauses $\mathcal{C}$. As explained in Chapter 4, our algorithm uses several types of individuals. Each individual is either root or blockable as summarized next; when we refer simply to an individual, we mean either a root or a blockable one.

- *Root* individuals are those that either occur in the input ABox, or are introduced by the *NI*-rule. Their important characteristic is that they can be connected in arbitrary, and not just tree-like, ways.
  - Root individuals that occur in the input ABox are called *named* individuals.

– Root individuals that are introduced by the *NI*-rule are identified by finite strings of the form $a.\gamma_1.\ldots.\gamma_n$ where a is a named individual, each γ_ℓ is of the form $\langle R.B.i\rangle$, and $n \geq 0$. Root individuals introduced by applying the *NI*-rule to an assertion $s \approx t\,@^{u}_{\leq n\,R.B}$ are all of the form $u.\langle R.B.i\rangle$ with $1 \leq i \leq n$.

- *Blockable* individuals are introduced by the $\geq$-rule, and make up the tree-like parts of a model. The set of blockable individuals is disjoint from the set of root individuals. Blockable individuals are identified by finite strings of the form $s.i_1.i_2.\ldots.i_n$ where s is a root individual, each i_ℓ is an integer, and $n \geq 1$. This string representation naturally induces the parent–child relationship between individuals; for example, $s.2$ is the second child of the individual s, which can be either blockable or root.

We now introduce our algorithm.

Definition 13 (Hypertableau Algorithm)

Individuals. Given a set of *named* individuals N_I, the set of *root individuals* N_O is the smallest set such that $N_I \subseteq N_O$ and, if $x \in N_O$, then $x.\langle R, B, i\rangle \in N_O$ for each role R, literal concept B, and positive integer i. The set of *all individuals* N_A is the smallest set such that $N_O \subseteq N_A$ and, if $x \in N_A$, then $x.i \in N_A$ for each positive integer i. The individuals in $N_A \setminus N_O$ are *blockable individuals*. A blockable individual $x.i$ is a *successor* of x, and x is a *predecessor* of $x.i$. *Descendant* and *ancestor* are the transitive closures of successor and predecessor, respectively.

ABoxes. The hypertableau algorithm operates on ABoxes that are obtained by extending the standard definition from Section 2.1 as follows.

- In addition to assertions from Section 2.1, an ABox can contain annotated equality assertions (as given by Definition 11) and a special assertion $\bot$ that is

false in all interpretations. Furthermore, assertions can refer to the individuals from N_A and not only from N_I.

- Each (in)equality $s \approx t$ ($s \not\approx t$) also stands for the symmetric (in)equality $t \approx s$ ($t \not\approx s$). The same is true for annotated equalities.

- An ABox $\mathcal{A}$ can contain *renamings* of the form $a \mapsto b$ where a and b are root individuals. Let $\mapsto^*$ be the reflexive-transitive closure of $\mapsto$ in $\mathcal{A}$. An individual b is the *canonical name* of a root individual a in $\mathcal{A}$, written $b = \|a\|_{\mathcal{A}}$, if b is the only individual such that both $a \mapsto^* b$ and there exists no individual $c \neq b$ such that $b \mapsto^* c$ (i.e. b is the tail of every maximal $\mapsto$ path originating at a). As we show in Lemma 9, the derivation rules of our calculus ensure that $\mapsto$ is a functional and acyclic relation, so a unique individual b satisfying this definition exists for every a. In order to make this definition complete, however, we arbitrarily define a to be its own canonical name in the case that no individual b meeting the above definition exists.

An *input ABox* is an ABox containing only named individuals, no annotated equalities, and no renamings, and in which all concepts and all roles are atomic.

Satisfaction of such ABoxes in an interpretation is obtained by a straightforward generalization of the definitions in Section 2.1: all individuals are interpreted as elements of the interpretation domain $\triangle^{\mathcal{I}}$, and $\mathcal{I} \models a \mapsto b$ iff $a^{\mathcal{I}} = b^{\mathcal{I}}$.

Pairwise Anywhere Blocking. The *labels of an individual* s and *of an individual pair* $\langle s, t \rangle$ in an ABox $\mathcal{A}$ are defined as follows:

$$\mathcal{L}_{\mathcal{A}}(s) = \{\, A \mid A(s) \in \mathcal{A} \text{ and } A \text{ is an atomic concept} \,\}$$
$$\mathcal{L}_{\mathcal{A}}(s,t) = \{\, R \mid R(s,t) \in \mathcal{A} \,\}$$

Let $\prec$ be a strict ordering (i.e., a transitive and irreflexive relation) on N_A containing the ancestor relation—that is, if s' is an ancestor of s, then $s' \prec s$. By induction on $\prec$, we assign to each individual s in $\mathcal{A}$ a status as follows:

- a blockable individual s is *directly blocked by* a blockable individual t if and only if the following conditions are satisfied, for s' and t' the predecessors of s and t, respectively:
  - s' and t' are blockable individuals,
  - t is not blocked,
  - $t \prec s$,
  - $\mathcal{L}_{\mathcal{A}}(s) = \mathcal{L}_{\mathcal{A}}(t)$ and $\mathcal{L}_{\mathcal{A}}(s') = \mathcal{L}_{\mathcal{A}}(t')$, and
  - $\mathcal{L}_{\mathcal{A}}(s, s') = \mathcal{L}_{\mathcal{A}}(t, t')$ and $\mathcal{L}_{\mathcal{A}}(s', s) = \mathcal{L}_{\mathcal{A}}(t', t)$;
- s is *indirectly blocked* iff it has a predecessor that is blocked; and
- s is *blocked* iff it is either directly or indirectly blocked.

Pruning. The ABox $\mathsf{prune}_{\mathcal{A}}(s)$ is obtained from $\mathcal{A}$ by removing all assertions containing a descendant of s.

Merging. The ABox $\mathsf{merge}_{\mathcal{A}}(s \to t)$ is obtained from $\mathsf{prune}_{\mathcal{A}}(s)$ by replacing the individual s with the individual t in all assertions and their annotations (but not in renamings) and, if both s and t are root individuals, adding the renaming $s \mapsto t$.

Derivation Rules. Table 5.4 specifies *derivation rules* that, given an ABox $\mathcal{A}$ and a set of HT-clauses $\mathcal{C}$, derive one or more ABoxes $\mathcal{A}_1, \ldots, \mathcal{A}_n$. In the *Hyp*-rule, σ is a mapping from the set of variables N_V to the individuals occurring in the assertions of $\mathcal{A}$, and $\sigma(U)$ is the result of replacing each variable x in the atom U with $\sigma(x)$.

Rule Precedence. The $\approx$-rule can be applied to a (possibly annotated) equality $s \approx t$ in an ABox $\mathcal{A}$ only if $\mathcal{A}$ does not contain an equality $s \approx t \, @^{u}_{\leq n\, R.B}$ to which the *NI*-rule is applicable (with the same s and t).

Clash. An ABox $\mathcal{A}$ contains a *clash* iff $\bot \in \mathcal{A}$; otherwise, $\mathcal{A}$ is *clash-free*.

Derivation. For a set of HT-clauses $\mathcal{C}$ and an input ABox $\mathcal{A}$, a *derivation* is a pair (T, λ) where T is a finitely branching tree and λ is a function that labels the nodes of T with ABoxes such that the following properties hold for each node t of T:

- $\lambda(t) = \mathcal{A}$ if t is the root of T;
- t is a leaf of T if $\bot \in \lambda(t)$ or no derivation rule is applicable to $\lambda(t)$ and $\mathcal{C}$;
- in all other cases, t has children $t_1, \ldots, t_n$ such that $\lambda(t_1), \ldots, \lambda(t_n)$ are exactly the results of applying one (arbitrarily chosen, but respecting the rule precedence) applicable rule to $\lambda(t)$ and $\mathcal{C}$. △

We stress several important aspects of Definition 13.

As described in Section 3.5.3 and Section 3.5.4, the *NI*-rule promotes blockable individuals into root individuals, and annotated equalities carry information about how individuals should be promoted. If the preconditions of the *NI*-rule are satisfied for an annotated equality $s \approx t\,@^{u}_{\leq n\,R.B}$, then the rule must be applied even if $s = t$; hence, such an equality plays a role in a derivation even though it is a logical tautology. Furthermore, even though the *NI*-rule is not applied to $s \approx t\,@^{u}_{\leq n\,R.B}$ if u is a blockable individual, the equality cannot be eagerly simplified into $s \approx t$ because u can subsequently be merged into a root individual so the annotation might become important. Finally, if $\mathcal{C}$ has been obtained by clausification of a DL knowledge base that does not use nominals, inverse roles, and number restrictions, then the precondition of the *NI*-rule will never be satisfied, so we need not keep track of annotations at all.

Renamings are used to keep track of root individuals that are merged into other root individuals, which is necessary to make the *NI*-rule sound. For example, if a root individual $a.\langle R, B, 2\rangle$ is merged into a named individual b, then the *NI*-rule must use b instead of $a.\langle R, B, 2\rangle$ in all future inferences.

Table 5.4: Derivation Rules of the Hypertableau Calculus

Hyp-rule	If 1. $r \in \mathcal{C}$, where $r = U_1 \wedge \ldots \wedge U_m \rightarrow V_1 \vee \ldots \vee V_n$, and 2. a mapping σ from the variables in r to the individuals of $\mathcal{A}$ exists such that 2.1 there is no variable x in r such that $\sigma(x)$ is indirectly blocked, 2.2 $\sigma(U_i) \in \mathcal{A}$ for each $1 \leq i \leq m$, and 2.3 $\sigma(V_j) \notin \mathcal{A}$ for each $1 \leq j \leq n$, then $\mathcal{A}_1 := \mathcal{A} \cup \{\bot\}$ if $n = 0$; $\mathcal{A}_j := \mathcal{A} \cup \{\sigma(V_j)\}$ for $1 \leq j \leq n$ otherwise.
$\geq$-rule	If 1. $\geq n\, R.B(s) \in \mathcal{A}$, 2. s is not blocked in $\mathcal{A}$, and 3. $\mathcal{A}$ does not contain individuals $u_1, \ldots, u_n$ such that 3.1 $\{\mathsf{ar}(R, s, u_i), B(u_i) \mid 1 \leq i \leq n\} \cup \{u_i \not\approx u_j \mid 1 \leq i < j \leq n\} \subseteq \mathcal{A}$, and 3.2 u_i is not indirectly blocked in $\mathcal{A}$ for each $1 \leq i \leq n$ then $\mathcal{A}_1 := \mathcal{A} \cup \{\mathsf{ar}(R, s, t_i),\ B(t_i) \mid 1 \leq i \leq n\} \cup \{t_i \not\approx t_j \mid 1 \leq i < j \leq n\}$ where $t_1, \ldots, t_n$ are fresh distinct (blockable) successors of s.
$\approx$-rule	If 1. $s \approx t \in \mathcal{A}$ (the equality can possibly be annotated), 2. $s \neq t$, and 3. neither s nor t is indirectly blocked then $\mathcal{A}_1 := \mathsf{merge}_{\mathcal{A}}(s \rightarrow t)$ if t is a named individual, or t is a root individual and s is not a named individual, or s is a descendant of t; $\mathcal{A}_1 := \mathsf{merge}_{\mathcal{A}}(t \rightarrow s)$ otherwise.
$\bot$-rule	If $s \not\approx s \in \mathcal{A}$ where s is not indirectly blocked then $\mathcal{A}_1 := \mathcal{A} \cup \{\bot\}$.
NI-rule	If 1. $s \approx t\, @^u_{\leq n\, R.B} \in \mathcal{A}$ (the symmetry of $\approx$ applies as usual), 2. u is a root individual, 3. s is a blockable individual that is not a successor of u, 4. t is a blockable individual, and 5. neither s nor t is indirectly blocked then $\mathcal{A}_i := \mathsf{merge}_{\mathcal{A}}(s \rightarrow \|u.\langle R, B, i\rangle\|_{\mathcal{A}})$ for each $1 \leq i \leq n$.

While we defer the formal impact of blocking on model construction to Definition 15, the intuition is that assertions containing at least one successor of a blocked individual are not used to construct a model from an ABox labeling a leaf in a derivation. The $\geq$-rule is thus inapplicable to all blocked individuals; this is required to ensure termination. Since assertions containing at least one indirectly blocked individual are not relevant to model construction, derivation rules (other than the $\bot$-rule[2]) are

[2]While most implementations will apply the $\bot$-rule with high precedence, which would prevent its application to indirectly blocked individuals, our calculus does allow clashes to result from assertions of the form $s \not\approx s$ where s is indirectly blocked. In such cases further rule applications in other parts

applicable only to individuals that are either directly blocked or not blocked, as this is sufficient for completeness. Since all rules are sound, however, one may choose to disregard this restriction if that makes implementation easier. We also note that it is not uncommon for individuals blocked at node t in a derivation tree to become unblocked in a descendant of t; the distinction between indirectly blocked individuals and pruning is thus significant. In fact, we define the ordering $\prec$ used to compute blocking only with respect to a particular ABox, which allows for the possibility of different orderings at different nodes in the derivation tree.

We now prove some properties of our calculus that will be useful in showing how a derivation tree can be explored to determine the consistency of a knowledge base, and to construct a model of the knowledge base if one exists. We begin by introducing the notion of HT-ABoxes, which formalizes the idea of forest-shaped ABoxes introduced in Section 3.3.

Definition 14 (HT-ABoxes) An ABox $\mathcal{A}$ is an *HT-ABox* if it satisfies the following conditions, for R an atomic role, S a role, B an atomic concept which is not a nominal guard concept, O_a a nominal guard concept, $s, t, u \in N_A$, $a \in N_O$, $b \in N_I$, and i, j integers.

1. Each role assertion in $\mathcal{A}$ is of the form $R(a, s)$, $R(s, a)$, $R(s, s.i)$, $R(s.i, s)$, or $R(s, s)$.

2. Each equality in $\mathcal{A}$ is either of the form $s \approx t\, @^a_{\leq n\, R.B}$ with s a blockable individual that is not a successor of a and t a blockable individual, or it is a possibly annotated equality of the form $s.i \approx s.j$, $s.i \approx s$, $s.i.j \approx s$, $s \approx s$, or $s \approx a$. (The symmetry of $\approx$ applies in all these cases as usual.)

3. Each concept assertion in $\mathcal{A}$ is of the form $B(s)$, $\geq n\, S.B(s)$, or $O_a(b)$.

of the ABox would always eventually result in a clash, so applying the $\bot$-rule even to indirectly blocked individuals is a (minor) optimization.

4. If $\mathcal{A}$ contains $s \approx t \, @^{u}_{\leq n \, R.B}$, then $\mathcal{A}$ also contains $\mathsf{ar}(R, u, s)$ and $\mathsf{ar}(R, u, t)$.

5. If $\mathcal{A}$ contains a blockable individual $s.i$ in some assertion, then $\mathcal{A}$ must contain an assertion of the form $R(s, s.i)$ or $R(s.i, s)$.

6. $\mathcal{A}$ contains at least one assertion.

7. The relation $\mapsto$ in $\mathcal{A}$ is acyclic, $\mathcal{A}$ contains at most one renaming $a \mapsto b$ for an individual a, and, if $\mathcal{A}$ contains $a \mapsto b$, then a does not occur in any assertion in $\mathcal{A}$. △

Clearly, each input ABox is an HT-ABox. We now prove that, given an HT-ABox, our calculus produces only HT-ABoxes.

Lemma 9 (HT-Preservation) *For $\mathcal{C}$ a set of HT-clauses and $\mathcal{A}$ an HT-ABox, each ABox $\mathcal{A}'$ obtained by applying a derivation rule to $\mathcal{C}$ and $\mathcal{A}$ is an HT-ABox.*

Proof Let $\mathcal{C}$, $\mathcal{A}$, and $\mathcal{A}'$ be as stated in the lemma. We now analyze each derivation rule from Table 5.4 and show that $\mathcal{A}'$ satisfies the remaining conditions of HT-ABoxes.

(*Hyp*-rule) Consider an application of the *Hyp*-rule to an HT-clause r of type (5.5) with a mapping σ, deriving an assertion $\sigma(V)$.

Assume that V is of the form $y_i \approx y_j \, @^{x}_{\leq k \, R.B}$, so $\sigma(V)$ is of the form $s \approx t \, @^{u}_{\leq k \, R.B}$. By Definition 11, the antecedent of r then contains atoms of the form $\mathsf{ar}(R, x, y_i)$ and $\mathsf{ar}(R, x, y_j)$ so, by the precondition of the *Hyp*-rule, $\mathcal{A}$ contains assertions $\mathsf{ar}(R, u, s)$ and $\mathsf{ar}(R, u, t)$. If u is a root individual and either s or t is a blockable individual that is not a successor of u, then $\sigma(V)$ clearly satisfies Property (2) of HT-ABoxes. Otherwise, since $\mathcal{A}$ satisfies Property (1) of HT-ABoxes, we have the possibilities shown in Table 5.5, for v a blockable individual, and a and b root individuals. For brevity, we omit the symmetric combinations where the roles of $\mathsf{ar}(R, u, s)$ and $\mathsf{ar}(R, u, t)$ are exchanged. Clearly, $\sigma(V)$ satisfies Property (2) of HT-ABoxes. Finally, $\sigma(V)$ obviously satisfies Property (4) of HT-ABoxes.

Table 5.5: Cases in an Application of the *Hyp*-Rule to Role Assertions

$\mathsf{ar}(R, u, s)$	$\mathsf{ar}(R, u, t)$	$s \approx t \, @^{u}_{\leq k \, R.B}$
$\mathsf{ar}(R, v, a)$	$\mathsf{ar}(R, v, b)$	$a \approx b \, @^{v}_{\leq k \, R.B}$
$\mathsf{ar}(R, v, a)$	$\mathsf{ar}(R, v, v.n)$	$a \approx v.n \, @^{v}_{\leq k \, R.B}$
$\mathsf{ar}(R, v, a)$	$\mathsf{ar}(R, v, v)$	$a \approx v \, @^{v}_{\leq k \, R.B}$
$\mathsf{ar}(R, v.n, a)$	$\mathsf{ar}(R, v.n, v)$	$a \approx v \, @^{v.n}_{\leq k \, R.B}$
$\mathsf{ar}(R, v, v.m)$	$\mathsf{ar}(R, v, v.n)$	$v.m \approx v.n \, @^{v}_{\leq k \, R.B}$
$\mathsf{ar}(R, v, v.m)$	$\mathsf{ar}(R, v, v)$	$v.m \approx v \, @^{v}_{\leq k \, R.B}$
$\mathsf{ar}(R, v.n, v.n.m)$	$\mathsf{ar}(R, v.n, v)$	$v.n.m \approx v \, @^{v.n}_{\leq k \, R.B}$
$\mathsf{ar}(R, v, v)$	$\mathsf{ar}(R, v, v)$	$v \approx v \, @^{v}_{\leq k \, R.B}$
$\mathsf{ar}(R, v.n, v.n)$	$\mathsf{ar}(R, v.n, v)$	$v.n \approx v \, @^{v.n}_{\leq k \, R.B}$
$\mathsf{ar}(R, v.n, v)$	$\mathsf{ar}(R, v.n, v)$	$v \approx v \, @^{v.n}_{\leq k \, R.B}$

Assume that V is of the form $x \approx z_j$, so $\sigma(V)$ is of the form $s \approx t$. By Definition 11, the antecedent of r then contains an atom $O_a(z_j)$, so either $O_a(s) \in \mathcal{A}$ or $O_a(t) \in \mathcal{A}$. By Property (3) of HT-ABoxes, either s or t is a named individual, so $\sigma(V)$ satisfies Property (2) of HT-ABoxes.

Assume that V is of the form $R(x, x)$. Then, $\sigma(V)$ is of the form $R(s, s)$, and it satisfies Property (1) of HT-ABoxes.

Assume that V is of the form $R(x, y_i)$ or $R(y_i, x)$, so $\sigma(V)$ is of the form $R(s, t)$. By Definition 11, the antecedent of r then contains an atom of the form $S(x, y_i)$ or $S(y_i, x)$, and either $S(s, t) \in \mathcal{A}$ or $S(t, s) \in \mathcal{A}$; these assertions satisfy Property (1) of HT-ABoxes, so $R(s, t)$ satisfies it as well.

Assume that V is of the form $R(x, z_j)$ or $R(z_j, x)$, so $\sigma(V)$ is of the form $R(s, t)$. By Definition 11, the antecedent of r then contains an atom of the form $O_a(z_j)$ for O_a a nominal guard concept, and either $O_a(s) \in \mathcal{A}$ or $O_a(t) \in \mathcal{A}$; by Property (3) of HT-ABoxes, either s or t is a named individual, so $R(s, t)$ satisfies Property (1) of HT-ABoxes.

Assume that V is of the form $B(x)$, $\geq n \, S.B(x)$, or $B(y_i)$, so $\sigma(V)$ is of the form $B(s)$ or $\geq n \, S.B(s)$. By Definition 11, B is a literal but not a nominal guard concept,

so $\sigma(V)$ satisfies Property (3) of HT-ABoxes.

($\geq$-rule) Consider an application of the $\geq$-rule to an assertion $\geq n\, R.B(s)$. By Property (3) of HT-ABoxes, B is not a nominal guard concept, so all assertions $B(t_i)$ introduced by the rule satisfy Property (3) of HT-ABoxes. Furthermore, all t_i introduced by the rule are fresh blockable successors of s, and all role assertions introduced by the rule are of the form $R(s, t_i)$ or $R(t_i, s)$, so they satisfy Properties (1) and (5) of HT-ABoxes. The inequalities introduced by the rule trivially satisfy the properties of HT-ABoxes.

($\approx$-rule) Consider an application of the $\approx$-rule to a possibly annotated equality $s \approx t$, where s is merged into t (the annotation of the equality plays no role here). By the conditions on the $\mapsto$ relation of $\mathcal{A}$, the ABox $\mathcal{A}$ contains no renaming for s or t, so the renaming $s \mapsto t$ is the only renaming for s in $\mathcal{A}'$, and adding this renaming to $\mathcal{A}$ does not introduce a cycle in $\mapsto$. Merging replaces all occurrences of s in $\mathcal{A}$, so no assertion of $\mathcal{A}'$ contains s. Hence, the $\mapsto$ relation in $\mathcal{A}'$ satisfies Property (7) of HT-ABoxes.

The *NI*-rule is not applicable to $s \approx t$ by the rule precedence, so, by the preconditions of the *NI*-rule and Property (2) of HT-ABoxes, $s \approx t$ can be of the form $v \approx a$, $v.i \approx v.j$, $v.i \approx v$, or $v.i.j \approx v$ for $a \in N_O$ and $v \in N_A$; we denote this property with (*). Since pruning and replacements are applied to all assertions of $\mathcal{A}$ uniformly, $\mathcal{A}'$ clearly satisfies Property (4) of HT-ABoxes. Furthermore, pruning removes all successors of s, so $\mathcal{A}'$ satisfies Property (5) of HT-ABoxes. We next consider the types of assertions of $\mathcal{A}$ that change when s is merged into t.

Consider a role assertion $R(s, u) \in \mathcal{A}$ that is changed into $R(t, u) \in \mathcal{A}'$. If either t or u is a root individual, then $R(t, u)$ clearly satisfies Property (1) of HT-ABoxes, so assume that t and u are both blockable individuals. Then, u is not a successor of s, since the $\approx$-rule prunes all assertions that contain a descendant of the merged individual. But then, by (*) and since $R(s, u)$ satisfies Property (1) of HT-ABoxes,

Table 5.6: Cases in an Application of the $\approx$-Rule to Role Assertions

$R(s,u)$	$s \approx t$	$R(t,u)$
$R(v.i,v)$	$v.i \approx v.j$	$R(v.j,v)$
$R(v.i,v)$	$v.i \approx v$	$R(v,v)$
$R(t.j.i,t.j)$	$t.j.i \approx t$	$R(t,t.j)$
$R(v.i,v.i)$	$v.i \approx v.j$	$R(v.j,v.j)$
$R(v.i,v.i)$	$v.i \approx v$	$R(v,v)$
$R(t.j.i,t.j.i)$	$t.j.i \approx t$	$R(t,t)$

we have the possibilities shown in Table 5.6. The cases when $R(u,s) \in \mathcal{A}$ is changed into $R(u,t) \in \mathcal{A}'$ by merging are analogous.

We now consider the form of equalities that can be derived from other equalities via merging. Since pruning and replacements are applied to all assertions of $\mathcal{A}$ uniformly, $\mathcal{A}'$ clearly satisfies Property (4) of HT-ABoxes. An equality $u \approx v\,@^{s}_{\leq n\,R.C}$ can be changed into $u \approx v\,@^{t}_{\leq n\,R.C}$, but the resulting equality always satisfies Property (2) of HT-ABoxes. Furthermore, for a a root individual, $s \approx u\,@^{a}_{\leq n\,R.C}$ can be changed into $t \approx u\,@^{a}_{\leq n\,R.C}$, and $s \approx a$ can be changed into $t \approx a$; however, in both cases, the resulting equality satisfies Property (2) of HT-ABoxes. For the remaining cases, assume that a possibly annotated equality $s \approx u$ is changed into a possibly annotated equality $t \approx u$. If s is a root individual, then t is a root individual as well (the $\approx$-rule never merges a root individual into a blockable one), so $t \approx u$ satisfies Property (2) of HT-ABoxes. Assume that s is a blockable individual. Since the $\approx$-rule prunes all assertions that contain a descendant of the merged individual, u is not a successor of s. By (*), Property (2) of HT-ABoxes, and the fact that the NI-rule is not applicable to $\mathcal{A}$, we have the possibilities shown in Table 5.7. In all cases, the resulting assertion satisfies Property (2) of HT-ABoxes. Furthermore, replacing s with t in $s \approx t \in \mathcal{A}$ results in $t \approx t \in \mathcal{A}'$, so $\mathcal{A}'$ satisfies Property (6) of HT-ABoxes.

Consider an assertion $C(s) \in \mathcal{A}$ that is changed into $C(t) \in \mathcal{A}'$. The only nontrivial case is when C is a nominal guard concept O_a. By Property (3) of HT-ABoxes, s is

Table 5.7: Cases in an Application of the $\approx$-Rule to Equalities

$s \approx u$	$s \approx t$	$t \approx u$
$v.i \approx v.k$	$v.i \approx v.j$	$v.j \approx v.k$
$v.i \approx v$	$v.i \approx v.j$	$v.j \approx v$
$u.k.i \approx u$	$u.k.i \approx u.k.j$	$u.k.j \approx u$
$v.i \approx v.k$	$v.i \approx v$	$v \approx v.k$
$v.i \approx v$	$v.i \approx v$	$v \approx v$
$u.k.i \approx u$	$u.k.i \approx u.k$	$u.k \approx u$
$t.j.i \approx t.j.k$	$t.j.i \approx t$	$t \approx t.j.k$
$t.j.i \approx t.j$	$t.j.i \approx t$	$t \approx t.j$
$t.j.i \approx t$	$t.j.i \approx t$	$t \approx t$

then a named individual. The $\approx$-rule replaces named individuals only with other named individuals, so t is a named individual as well. Thus, $C(t)$ satisfies Property (3) of HT-ABoxes.

(NI-rule) Consider an application of the NI-rule to an equality $s \approx t\,@^{u}_{\leq n\,R.B}$ that merges s into a root individual $\|u.\langle R, B, i\rangle\|_{\mathcal{A}}$. The individual s is blockable, so no renaming is added to $\mathcal{A}$ and the $\mapsto$ relation in $\mathcal{A}'$ satisfies Property (7) of HT-ABoxes. Since s is replaced by a root individual in role and equality assertions, all resulting assertions satisfy Properties (1) and (2) of HT-ABoxes. Since s is not a named individual, no assertion involving a nominal guard concept is affected by merging, so $\mathcal{A}'$ satisfies Property (3). Since pruning and replacements are applied to all assertions of $\mathcal{A}$ uniformly, $\mathcal{A}'$ clearly satisfies Property (4) of HT-ABoxes. Pruning removes all successors of s, so $\mathcal{A}'$ satisfies Property (5) of HT-ABoxes. Finally, $\mathcal{A}'$ is clearly not empty, so it satisfies Property (6). □

We next prove soundness and completeness of our calculus. We use these notions as is customary in resolution-based theorem proving: a calculus is sound if its derivation rules preserve satisfiability of a theory, and it is complete if, whenever the calculus terminates without detecting a contradiction, the theory is indeed satisfiable.

Lemma 10 (Soundness) *Let $\mathcal{C}$ be a set of HT-clauses and $\mathcal{A}$ an input ABox such that $\mathcal{I}$ is a model of $(\mathcal{C}, \mathcal{A})$. Then, each derivation for $\mathcal{C}$ and $\mathcal{A}$ contains a branch such that for each node t on the branch there exists a model $\mathcal{I}'$ of $(\mathcal{C}, \lambda(t))$ which coincides with $\mathcal{I}$ on (N_R, N_C, N_N), where N_N is the set of named individuals.*

Proof We say that a model $\mathcal{I}$ of an ABox $\mathcal{A}_0$ is *NI-compatible* with $\mathcal{A}_0$ if the following conditions are satisfied:

- For each root individual a occurring in $\mathcal{A}_0$, each concept $\leq n\, R.B$, and each $\alpha \in \triangle^{\mathcal{I}}$ such that $a^{\mathcal{I}} \in (\leq n\, R.B)^{\mathcal{I}}$, $\langle a^{\mathcal{I}}, \alpha \rangle \in R^{\mathcal{I}}$, and $\alpha \in B^{\mathcal{I}}$, we have that $\alpha = (a.\langle R, B, i \rangle)^{\mathcal{I}}$ for some $1 \leq i \leq n$.
- If $s \approx t\, @^{u}_{\leq n\, R.B} \in \mathcal{A}_0$, then we have $\langle u^{\mathcal{I}}, s^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$, $\langle u^{\mathcal{I}}, t^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$, $s^{\mathcal{I}} \in B^{\mathcal{I}}$, $t^{\mathcal{I}} \in B^{\mathcal{I}}$, and $u^{\mathcal{I}} \in (\leq n\, R.B)^{\mathcal{I}}$.

Intuitively, the first condition ensures that each root individual $a.\langle R, B, i \rangle$ is interpreted as an appropriate "neighbor" of $a^{\mathcal{I}}$, and the second condition ensures that u, s, and t are interpreted in $\mathcal{I}$ in accordance with the annotation.

To prove this lemma, we first show the following property (*): if there exists a model $\mathcal{I}$ of $(\mathcal{C}, \mathcal{A}_0)$ that is *NI*-compatible with $\mathcal{A}_0$ and $\mathcal{A}_1, \ldots, \mathcal{A}_n$ are ABoxes obtained by applying a derivation rule to $\mathcal{C}$ and $\mathcal{A}_0$, then there exists a model $\mathcal{I}'$ of $(\mathcal{C}, \mathcal{A}_i)$ that coincides with $\mathcal{I}$ on (N_R, N_C, N_N) and is *NI*-compatible with $\mathcal{A}_i$, for some $1 \leq i \leq n$. Let $\mathcal{I}$ be a model of $(\mathcal{C}, \mathcal{A}_0)$ that is *NI*-compatible with $\mathcal{A}_0$, and consider all possible derivation rules that can derive $\mathcal{A}_1, \ldots, \mathcal{A}_n$ from $\mathcal{A}_0$ and $\mathcal{C}$.

(*Hyp*-rule) Consider an application of the *Hyp*-rule to an HT-clause r of the form (5.5). Since $\sigma(U_i) \in \mathcal{A}_0$, we have $\mathcal{I} \models \sigma(U_i)$ for all $1 \leq i \leq m$. But then, $\mathcal{I} \models \sigma(V_j)$ for some $1 \leq j \leq n$. Since $\mathcal{A}_j := \mathcal{A}_0 \cup \{\sigma(V_j)\}$, we have $\mathcal{I} \models (\mathcal{C}, \mathcal{A}_j)$.

If $\mathcal{I} \models \sigma(V_j)$ for some atom V_j not of the form $\psi = y_k \approx y_\ell\, @^{x}_{\leq h\, R.B}$, then $\mathcal{I}$ is clearly *NI*-compatible with $\mathcal{A}_j$. Assume then that $\mathcal{I} \models \sigma(V_j)$ for V_j of the form ψ. Considerlearly $\langle \sigma(x)^{\mathcal{I}}, \sigma(y_k)^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$, $\langle \sigma(x)^{\mathcal{I}}, \sigma(y_\ell)^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$, $\sigma(y_k)^{\mathcal{I}} \in B^{\mathcal{I}}$, and $\sigma(y_\ell)^{\mathcal{I}} \in B^{\mathcal{I}}$ by

properties 5 and 6 of HT-clauses and the fact that $\mathcal{I} \models \sigma(U_i)$ for each $1 \leq i \leq m$ and $\mathcal{I} \not\models \sigma(V_k)$ for each $k \neq j$ such that $1 \leq k \leq n$.

Assume that $\mathcal{I}$ is not *NI*-compatible with $\mathcal{A}_j$ for each $1 \leq j \leq n$. From the above, $\mathcal{I} \not\models \sigma(V_j)$ for each V_j not of the form ψ, and $\sigma(x)^{\mathcal{I}} \notin (\leq h\, R.B)^{\mathcal{I}}$ for each V_j of the form ψ. Let $\mu : N_V \to \triangle^{\mathcal{I}}$ be a variable mapping such that $\mu(x) = \sigma(x)^{\mathcal{I}}$ and $\mu(y_k) = \sigma(y_k)^{\mathcal{I}}$ for each branch variable y_k not occurring in an atom of the form ψ; furthermore, for each set of branch variables $y_1, \ldots, y_{h+1}$ occurring in an atom of the form ψ, we set $\mu(y_1), \ldots, \mu(y_{h+1})$ to arbitrarily chosen domain elements that verify $\sigma(x)^{\mathcal{I}} \notin (\leq h\, R.B)^{\mathcal{I}}$. Clearly, $\mathcal{I}, \mu \not\models V_j$ for each V_j not occurring in a subset (5.6) or (5.7) of r; furthermore, by the definition of μ, we have that $\mathcal{I}, \mu \not\models V_j$ for each V_j occurring in a subset of (5.6) or (5.7) of r. But then, we conclude $\mathcal{I}, \mu \not\models (\mathcal{C}, \mathcal{A}_0)$, which is a contradiction.

($\geq$-rule) Since $\geq n\, R.B(s) \in \mathcal{A}_0$, we have $\mathcal{I} \models \geq n\, R.B(s)$, which implies that domain elements $\alpha_1, \ldots, \alpha_n \in \triangle^{\mathcal{I}}$ exist where $\langle s^{\mathcal{I}}, \alpha_i \rangle \in R^{\mathcal{I}}$ and $\alpha_i \in B^{\mathcal{I}}$ for $1 \leq i \leq n$, and $\alpha_i \neq \alpha_j$ for $1 \leq i < j \leq n$. Let $\mathcal{I}'$ be an interpretation obtained from $\mathcal{I}$ by setting $t_i^{\mathcal{I}'} = \alpha_i$. Clearly, $\mathcal{I}' \models \mathsf{ar}(R, s, t_i)$, $\mathcal{I}' \models B(t_i)$, and $\mathcal{I}' \models t_i \not\approx t_j$ for $i \neq j$, so $\mathcal{I}' \models (\mathcal{C}, \mathcal{A}_1)$. The individuals t_i are not root individuals, so $\mathcal{I}'$ is *NI*-compatible with $\mathcal{A}_1$. Further, $\mathcal{I}$ and $\mathcal{I}'$ differ only with respect to interpretation of each t_i, which are not named individuals, so $\mathcal{I}'$ coincides with $\mathcal{I}$ on (N_R, N_C, N_N).

($\approx$-rule) Assume that the $\approx$-rule is applied to the assertion $s \approx t \in \mathcal{A}_0$ and s is merged into t. Since $\mathcal{I} \models s \approx t$, we have $s^{\mathcal{I}} = t^{\mathcal{I}}$. Pruning removes assertions, so $\mathcal{I}$ is a model of the pruned ABox by monotonicity. Merging simply replaces an individual with a synonym, so $\mathcal{I} \models (\mathcal{C}, \mathcal{A}_1)$. Furthermore, by Property (7) of HT-ABoxes, $\mathcal{A}$ does not contain renamings for s and t, so $\|s\|_{\mathcal{A}_1} = t$; hence, $\mathcal{I}$ is *NI*-compatible with $\mathcal{A}_1$.

($\bot$-rule) This rule is never applicable if $(\mathcal{C}, \mathcal{A}_0)$ is satisfiable.

(*NI*-rule) Assume that the *NI*-rule is applied to some $s \approx t\, @^{u}_{\leq n\, R.B} \in \mathcal{A}_0$ and s is merged into a root individual. Since $\mathcal{I}$ is *NI*-compatible with $\mathcal{A}_0$, we have

$u^{\mathcal{I}} \in (\leq n\, R.B)^{\mathcal{I}}$, $\langle u^{\mathcal{I}}, s^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$, $s^{\mathcal{I}} \in B^{\mathcal{I}}$, and $s^{\mathcal{I}} = (u.\langle R, B, i\rangle)^{\mathcal{I}}$ for some $1 \leq i \leq n$. Let $v_i = \|u.\langle R, B, i\rangle\|_{\mathcal{A}_0}$; since $\mathcal{I}$ is NI-compatible, we have $(u.\langle R, B, i\rangle)^{\mathcal{I}} = v_i^{\mathcal{I}}$. Thus, the NI-rule replaces s by its synonym v_i, so $\mathcal{I} \models (\mathcal{C}, \mathcal{A}_i)$ just like in the case of the $\approx$-rule. If v_i does not occur in $\mathcal{A}_0$, the interpretation $\mathcal{I}$ may not be NI-compatible with $\mathcal{A}_i$ because it does not interpret $v_i.\langle S, C, \ell\rangle$ correctly. We then extend $\mathcal{I}$ to $\mathcal{I}'$ as follows. For each m, S, and C such that $v_i^{\mathcal{I}} \in (\leq m\, S.C)^{\mathcal{I}}$, let $\alpha_1, \ldots, \alpha_k$ be the elements of $\triangle^{\mathcal{I}}$ such that $\langle v_i^{\mathcal{I}}, \alpha_j\rangle \in S^{\mathcal{I}}$ and $\alpha_j \in C^{\mathcal{I}}$; clearly, $k \leq m$. We then set $(v_i.\langle S, C, \ell\rangle)^{\mathcal{I}'} = \alpha_\ell$ for $1 \leq \ell \leq k$. Since none of $v_i.\langle S, C, \ell\rangle$ occurs in $\mathcal{A}_i$, we have $\mathcal{I}' \models (\mathcal{C}, \mathcal{A}_i)$, so $\mathcal{I}'$ is NI-compatible with $\mathcal{A}_j$. Since v_i does not occur in $\mathcal{A}_0$, it is not a named individual, so $\mathcal{I}'$ coincides with $\mathcal{I}$ on (N_R, N_C, N_N).

This completes the proof of (*). To prove the main claim of this lemma, let $\mathcal{A}$ be an input ABox. Similarly as for the NI-rule in the proof of (*), we can extend $\mathcal{I}$ to a model $\mathcal{I}'$ of $(\mathcal{C}, \mathcal{A})$ which coincides with $\mathcal{I}$ on (N_R, N_C, N_N). Since $\mathcal{A}$ does not contain annotated equalities, $\mathcal{I}'$ is NI-compatible with $\mathcal{A}$. The claim of this lemma then follows by a straightforward inductive application of (*). □

We now present the procedure for constructing an interpretation from a clash-free HT-abox $\mathcal{A}$. Since our logic does not have the finite model property, we obtain this model by *unraveling* $\mathcal{A}$ as intuitively explained in Section 3.3. As usual, elements of the unraveled model are *paths* [Horrocks and Sattler, 2001; Horrocks and Sattler, 2007], with each path representing one copy of an individual occurring in $\mathcal{A}'$.

Definition 15 (Unravelling) Let $\mathcal{A}$ be a clash-free HT-ABox. Given an individual s that is directly blocked in $\mathcal{A}$, the *blocker* of s is the smallest individual t w.r.t. $\prec$ such that s is directly blocked by t.

A *path* is finite sequence of pairs of individuals $p = [\frac{s_0}{s'_0}, \ldots, \frac{s_n}{s'_n}]$. Let $\mathsf{tail}(p) = s_n$ and $\mathsf{tail}'(p) = s'_n$. Furthermore, let $q = [p \mid \frac{s_{n+1}}{s'_{n+1}}]$ be the path $[\frac{s_0}{s'_0}, \ldots, \frac{s_n}{s'_n}, \frac{s_{n+1}}{s'_{n+1}}]$; we say that q is a *successor* of p, and p is a *predecessor* of q. Given an HT-ABox $\mathcal{A}$, the set of all paths $\mathcal{P}(\mathcal{A})$ on $\mathcal{A}$ is defined inductively as follows:

- $[\frac{a}{a}] \in \mathcal{P}(\mathcal{A})$ for each root individual a occurring in $\mathcal{A}$;
- $[p \mid \frac{s'}{s'}] \in \mathcal{P}(\mathcal{A})$ if $p \in \mathcal{P}(\mathcal{A})$, s' is a successor of $\mathsf{tail}(p)$, s' occurs in $\mathcal{A}$, and s' is not blocked in $\mathcal{A}$; and
- $[p \mid \frac{s}{s'}] \in \mathcal{P}(\mathcal{A})$ if $p \in \mathcal{P}(\mathcal{A})$, s' is a successor of $\mathsf{tail}(p)$, s' occurs in $\mathcal{A}$, s' is directly blocked in $\mathcal{A}$, and s is the blocker of s' in $\mathcal{A}$.

Intuitively, $[p \mid \frac{s}{s'}]$ represents a copy of the nonblocked individual s standing in place of the individual s', which is either s or is blocked by s. If an individual s' is blocked by one of its ancestors s in $\mathcal{A}$, then $\mathcal{P}(\mathcal{A})$ will contain an infinite number of paths of the form $[..., \frac{s}{s}, \frac{t_1}{u_1}, ..., \frac{t_n}{u_n}, \frac{s}{s'}, \frac{t_1}{u_1}, ..., \frac{t_n}{u_n}, \frac{s}{s'}, \frac{t_1}{u_1}, ..., \frac{t_n}{u_n}, \frac{s}{s'}, ...]$.

Given HT-ABox $\mathcal{A}$, the *unraveling* $\Gamma(\mathcal{A})$ of $\mathcal{A}$ is the interpretation $(\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$ defined as follows:

$$
\begin{aligned}
\triangle^{\mathcal{I}} &= \mathcal{P}(\mathcal{A}) \\
a^{\mathcal{I}} &= [\tfrac{a}{a}] \text{ for each root individual } a \text{ that occurs in an assertion in } \mathcal{A} \\
a^{\mathcal{I}} &= b^{\mathcal{I}} \text{ if } a \neq b \text{ and } \|a\|_{\mathcal{A}} = b \\
A^{\mathcal{I}} &= \{p \mid A(\mathsf{tail}(p)) \in \mathcal{A}\} \\
R^{\mathcal{I}} &= \{\langle [\tfrac{a}{a}], p\rangle \mid a \text{ is a root individual and } R(a, \mathsf{tail}(p)) \in \mathcal{A}\} \cup \\
&\quad \{\langle p, [\tfrac{a}{a}]\rangle \mid a \text{ is a root individual and } R(\mathsf{tail}(p), a) \in \mathcal{A}\} \cup \\
&\quad \{\langle p, [p \mid \tfrac{s}{s'}]\rangle \mid R(\mathsf{tail}(p), s') \in \mathcal{A}\} \cup \\
&\quad \{\langle [p \mid \tfrac{s}{s'}], p\rangle \mid R(s', \mathsf{tail}(p)) \in \mathcal{A}\} \cup \\
&\quad \{\langle p, p\rangle \mid R(\mathsf{tail}(p), \mathsf{tail}(p)) \in \mathcal{A}\}
\end{aligned}
$$

$\triangle$

Next we show that the above unravelling models when applied to ABoxes generated by our calculus.

Lemma 11 (Completeness) *Let (T, λ) be a derivation for a set of HT-clauses $\mathcal{C}$ and an input ABox $\mathcal{A}$ such $\lambda(t) = \mathcal{A}'$ is clash-free for some leaf t of T. Then $\Gamma(\mathcal{A}')$ is a model of $(\mathcal{C}, \mathcal{A})$.*

Proof Let $\Gamma(\mathcal{A}') = (\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be the unraveling of $\mathcal{A}'$ as given by Definition 15. $\mathcal{A}'$ is an HT-ABox, so $\triangle^{\mathcal{I}}$ is not empty. We now show that, for each path p_s of the form $[\frac{s}{s'}]$ or $[q_s \mid \frac{s}{s'}]$ and each individual w, the following claims hold (*):

- *$R(s,s) \in \mathcal{A}'$ (resp. $A(s) \in \mathcal{A}'$) iff $\langle p_s, p_s \rangle \in R^{\mathcal{I}}$ (resp. $p_s \in A^{\mathcal{I}}$):* Immediate by the definition of $\Gamma(\mathcal{A}')$.

- *If $B(w) \in \mathcal{A}'$ and $\mathcal{L}_{\mathcal{A}'}(w) = \mathcal{L}_{\mathcal{A}'}(s')$ for B a literal concept, then $p_s \in B^{\mathcal{I}}$:* The proof is immediate if B is atomic. If $B = \neg A$, since the $\bot$-rule is not applicable to $\mathcal{A}'$, we have $A(w) \notin \mathcal{A}'$; but then, we have $A(s') \notin \mathcal{A}'$ and $A(s) \notin \mathcal{A}'$, which by the previous case implies $p_s \notin A^{\mathcal{I}}$.

- *If $\geq n\, R.B(s) \in \mathcal{A}'$, then $p_s \in (\geq n\, R.B)^{\mathcal{I}}$:* By the definition of paths, s is not blocked; since the $\geq$-rule is not applicable to $\geq n\, R.B(s)$, individuals $u_1, \ldots, u_n$ exist such that $\mathsf{ar}(R, s, u_i) \in \mathcal{A}'$ and $B(u_i) \in \mathcal{A}'$ for $1 \leq i \leq n$, and $u_i \not\approx u_j \in \mathcal{A}'$ for $1 \leq i < j \leq n$. Each assertion $\mathsf{ar}(R, s, u_i)$ satisfies Property (1) of HT-ABoxes, so each u_i can be of one of the following forms.

  - $u_i = s$. Let $p_{u_i} = p_s$. But then, (*), $\mathsf{ar}(R, s, u_i) \in \mathcal{A}'$, and $B(u_i) \in \mathcal{A}'$ imply $\langle p_s, p_{u_i} \rangle \in R^{\mathcal{I}}$ and $p_{u_i} \in B^{\mathcal{I}}$.

  - u_i is a successor of s. If u_i is directly blocked by the blocker v_i, then let $p_{u_i} = [p_s \mid \frac{v_i}{u_i}]$; otherwise, u_i is not blocked because s is not blocked, and let $p_{u_i} = [p_s \mid \frac{u_i}{u_i}]$. Either way, we have $\mathsf{ar}(R, \mathsf{tail}(p_s), u_i) \in \mathcal{A}'$, which, by the definition of $\Gamma(\mathcal{A}')$, implies $\langle p_s, p_{u_i} \rangle \in R^{\mathcal{I}}$. Furthermore, we have $\mathcal{L}_{\mathcal{A}'}(u_i) = \mathcal{L}_{\mathcal{A}'}(\mathsf{tail}(p_{u_i}))$ and $B(u_i) \in \mathcal{A}'$, so $p_{u_i} \in B^{\mathcal{I}}$.

– u_i is a blockable predecessor of s. Since s is blockable, we have $p_s = [q_s \mid \frac{s}{s'}]$; hence, let $p_{u_i} = q_s$. If s' is not blocked, then $s = s'$ and $\mathsf{tail}(p_{u_i}) = u_i$, so we have $\mathsf{ar}(R, s', \mathsf{tail}(p_{u_i})) \in \mathcal{A}'$. If s' is blocked by the blocker s, then by the definition of pairwise blocking $\mathcal{L}_{\mathcal{A}'}(\mathsf{tail}(p_{u_i}), s') = \mathcal{L}_{\mathcal{A}'}(u_i, s)$ and $\mathcal{L}_{\mathcal{A}'}(s', \mathsf{tail}(p_{u_i})) = \mathcal{L}_{\mathcal{A}'}(s, u_i)$, so we again have $\mathsf{ar}(R, s', \mathsf{tail}(p_{u_i})) \in \mathcal{A}'$. Either way, we have $\langle p_s, p_{u_i} \rangle \in R^{\mathcal{I}}$ by the definition of $\Gamma(\mathcal{A}')$. Furthermore, $B(u_i) \in \mathcal{A}'$ and $\mathcal{L}_{\mathcal{A}'}(u_i) = \mathcal{L}_{\mathcal{A}'}(\mathsf{tail}(p_{u_i}))$ imply $p_{u_i} \in B^{\mathcal{I}}$.

– u_i and s do not satisfy any of the previous three conditions. If s is a blockable individual, then u_i is a root individual, so let $p_{u_i} = [\frac{u_i}{u_i}]$. If s is a root individual, then u_i is not indirectly blocked in $\mathcal{A}'$ by Condition 3.2 of the $\geq$-rule; but then, none of the ancestors of u_i are blocked in $\mathcal{A}'$, so we can choose some $p_{u_i} \in \triangle^{\mathcal{I}}$ of the form $p_{u_i} = [p \mid \frac{u_i}{u_i}]$. Either way, we have $\mathsf{ar}(R, s, u_i) \in \mathcal{A}'$ and $B(u_i) \in \mathcal{A}'$, which imply $\langle p_s, p_{u_i} \rangle \in R^{\mathcal{I}}$ and $p_{u_i} \in B^{\mathcal{I}}$.

Consider now each $1 \leq i < j \leq n$. If $\mathsf{tail}'(p_{u_i}) \not\approx \mathsf{tail}'(p_{u_j}) \in \mathcal{A}'$, since $\bot \notin \mathcal{A}'$ and the $\bot$-rule is not applicable, we have $\mathsf{tail}'(p_{u_i}) \neq \mathsf{tail}'(p_{u_j})$, so $p_{u_i} \neq p_{u_j}$. Furthermore, if $\mathsf{tail}'(p_{u_i}) \not\approx \mathsf{tail}'(p_{u_j}) \notin \mathcal{A}'$, this is because $\mathsf{tail}'(p_{u_i}) \neq u_i$, which is possible only if s' is directly blocked by the blocker s and $u_i = s$ or u_i is a blockable predecessor of s. Note, however, that s can have at most one blockable predecessor, and that there can be at most one u_i such that $u_i = s$. Therefore, we have $u_i \neq u_j$, which implies $p_{u_i} \neq p_{u_j}$, and we conclude $p_s \in (\geq n\, R.B)^{\mathcal{I}}$.

For an assertion $\alpha' \in \mathcal{A}'$ of the form $a \approx b$ and $a \not\approx b$ with a and b named individuals, it is straightforward to see that $\Gamma(\mathcal{A}') \models \alpha'$. Furthermore, if α' is of the form $R(a, b)$ or $B(a)$, or $\geq n\, R.B(a)$ with a a named individual, (*) implies $\Gamma(\mathcal{A}') \models \alpha'$. Consider now each $\alpha \in \mathcal{A}$. By induction on the application of the derivation rules, it is straightforward to show that, if $\alpha \notin \mathcal{A}'$, then $\mathcal{A}'$ contains renamings that, when

applied to α, produce an assertion $\alpha' \in \mathcal{A}'$. But then, since $\Gamma(\mathcal{A}') \models \alpha'$, we have $\Gamma(\mathcal{A}') \models \alpha$ by the definition of $\Gamma(\mathcal{A}')$.

It remains to be shown that $\Gamma(\mathcal{A}') \models \mathcal{C}$. Consider each HT-clause $r \in \mathcal{C}$ containing atoms of the form $A_i(x)$, $U_k(x, x)$, $\mathsf{ar}(R_i, x, y_i)$, $B_i(y_i)$, and $C_j(z_j)$ in the antecedent. Furthermore, consider a variable mapping μ such that the antecedent of r is true in $\Gamma(\mathcal{A}')$ and μ—that is, $p_x \in A_i^{\mathcal{I}}$, $\langle p_x, p_x \rangle \in U_k^{\mathcal{I}}$, $\langle p_x, p_{y_i} \rangle \in R_i^{\mathcal{I}}$, $p_{y_i} \in B_i^{\mathcal{I}}$, and $p_{z_j} \in C_j^{\mathcal{I}}$ for $p_x = \mu(x)$, $p_{y_i} = \mu(y_i)$, and $p_{z_j} = \mu(z_j)$. Let $s = \mathsf{tail}(p_x)$, $s' = \mathsf{tail}'(p_x)$, and $t_i' = \mathsf{tail}'(p_{y_i})$. By the definition of $\Gamma(\mathcal{A}')$ and the fact that $\mathcal{L}_{\mathcal{A}'}(t_i') = \mathcal{L}_{\mathcal{A}'}(t_i)$, we have $A_i(s) \in \mathcal{A}'$, $U_k(s, s) \in \mathcal{A}'$, $B_i(t_i') \in \mathcal{A}'$, and $\mathsf{ar}(R_i, s, t_i') \in \mathcal{A}'$. Depending on the relationship between s and t_i', we define t_i as follows.

- t_i' is a successor of s. Let $t_i = t_i'$. Clearly, $B_i(t_i) \in \mathcal{A}'$ and $\mathsf{ar}(R_i, s, t_i) \in \mathcal{A}'$.
- t_i' is a predecessor of s. We have the following cases.
  - s directly blocks s'. Let t_i be the predecessor of s; such t_i exists since s is blockable. By the definition of pairwise blocking, then $\mathcal{L}_{\mathcal{A}'}(s', t_i') = \mathcal{L}_{\mathcal{A}'}(s, t_i)$ and $\mathcal{L}_{\mathcal{A}'}(t_i', s') = \mathcal{L}_{\mathcal{A}'}(t_i, s)$, and $\mathcal{L}_{\mathcal{A}'}(t_i') = \mathcal{L}_{\mathcal{A}'}(t_i)$; therefore, we conclude that $B_i(t_i) \in \mathcal{A}'$ and $\mathsf{ar}(R_i, s, t_i) \in \mathcal{A}'$.
  - s' is not blocked. Then $s = s'$; furthermore, t_i' is not blocked (as s' would be indirectly blocked otherwise). Let $t_i = t_i'$. Clearly, it is the case that $B_i(t_i) \in \mathcal{A}'$ and $\mathsf{ar}(R_i, s, t_i) \in \mathcal{A}'$.
- t_i' is neither the successor nor the predecessor of s. By Property (2) of HT-ABoxes, then $t_i' = s$ or t_i' is a root individual a, so $p_{y_i} = p_x$ or $p_{y_i} = [\frac{a}{a}]$, respectively. Let $t_i = \mathsf{tail}(p_{y_i})$. By $\langle p_x, p_{y_i} \rangle \in R_i^{\mathcal{I}}$ and the definition of $\Gamma(\mathcal{A}')$, we conclude that $B_i(t_i) \in \mathcal{A}'$ and $\mathsf{ar}(R_i, s, t_i) \in \mathcal{A}'$.

By Definition 11, the antecedent of r contains an atom of the form $O_a(z_j)$ for each nominal variable z_j. Thus, by the definition of I and Property (3) of HT-ABoxes, we have p_{z_j} is of the form $[\frac{u_j}{u_j}]$ for u_j a named individual; furthermore, $C_j(u_j) \in \mathcal{A}'$.

Let σ be a mapping such that $\sigma(x) = s$, $\sigma(y_i) = t_i$, and $\sigma(z_j) = u_j$. Clearly, neither s nor t_i are indirectly blocked, and $\sigma(U_j) \in \mathcal{A}'$ for each atom U_j in the antecedent of r. The *Hyp*-rule is not applicable to r, $\mathcal{A}'$, and σ, so r contains an atom V_i in the consequent such that $\sigma(V_i) \in \mathcal{A}'$. Depending on the type of V_i, we have the following possibilities.

- V_i is of the form $y_i \approx y_j \, @^x_{\leq k\, S.B}$; thus, we have $t_i \approx t_j \, @^s_{\leq k\, S.B} \in \mathcal{A}'$. Since the $\approx$-rule is not applicable to $\mathcal{A}'$, we have $t_i = t_j$. By Definition 11, r contains a subclause of the form (5.6) or (5.7), so the antecedent of r contains atoms $\mathsf{ar}(S, x, y_i)$ and $\mathsf{ar}(S, x, y_j)$; therefore, $\langle p_x, p_{y_i} \rangle \in S^{\mathcal{I}}$ and $\langle p_x, p_{y_j} \rangle \in S^{\mathcal{I}}$. The *NI*-rule is not applicable to $t_i \approx t_j \, @^s_{\leq k\, S.B}$ so, by the preconditions of the *NI*-rule, if s is a root individual, then t_i is either a root individual or a successor of s. Thus, by Property (1) of HT-ABoxes, we have these possibilities: t_i is a root individual, $t_i = s$, t_i is a predecessor of s, or t_i is a successor of s. If t_i is a root individual or if $t_i = s$, then $t_i = t_j$ implies $p_{y_i} = p_{y_j}$ by the definition of paths. Both p_{y_i} and p_{y_j} can be successors of p_x, but again, $t_i = t_j$ implies $p_{y_i} = p_{y_j}$. Both p_{y_i} and p_{y_j} can be predecessors of p_x; but then $p_{y_i} = p_{y_j}$ since p_x can have at most one predecessor. Assume that p_{y_i} is a predecessor, but p_{y_j} is a successor of p_x; since t_i is not blocked, it must be that $t_i \neq t_j$, which is a contradiction. Thus, $\Gamma(\mathcal{A}'), \mu \models r$.

- V_i is of the form $x \approx z_j$; thus, we have $s \approx u_j \in \mathcal{A}$. Since the $\approx$-rule is not applicable to $\mathcal{A}'$, we have $s = u_j$. Since u_j is a named individual, it cannot block other individuals, so $s' = s$, which implies $p_x = p_{z_j}$. Thus, $\Gamma(\mathcal{A}'), \mu \models r$.

- V_i is of the form $T_i(x, x)$; thus, we have $T_i(s, s) \in \mathcal{A}'$. By (*), we then have $\langle p_x, p_x \rangle \in R_i^{\mathcal{I}}$. Thus, $\Gamma(\mathcal{A}'), \mu \models r$.

- V_i is of the form $D_i(x)$ for D_i a literal concept or of the form $\geq n\, T.B$; thus, we have $D_i(s) \in \mathcal{A}'$. By (*), we then have $p_x \in D_i^{\mathcal{I}}$. Thus, $\Gamma(\mathcal{A}'), \mu \models r$.

- V_i is of the form $E_i(y_i)$ for E_i a literal concept; thus, we have $E_i(t_i) \in \mathcal{A}'$. By the definition of blocking, we have $\mathcal{L}_{\mathcal{A}'}(t_i) = \mathcal{L}_{\mathcal{A}'}(t_i')$; by (*), we then have $p_{y_i} \in E_i^{\mathcal{I}}$. Thus, $\Gamma(\mathcal{A}'), \mu \models r$.

- V_i is of the form $\mathsf{ar}(S_i, x, y_i)$, so $\mathsf{ar}(S_i, s, t_i) \in \mathcal{A}'$. By the definition of blocking, we have $\mathsf{ar}(S_i, s', t_i') \in \mathcal{A}'$. Finally, by the definition of $\Gamma(\mathcal{A}')$, we have $\langle p_x, p_{y_i} \rangle \in S_i^{\mathcal{I}}$. Thus, $\Gamma(\mathcal{A}'), \mu \models r$.

- V_i is of the form $\mathsf{ar}(S_j, x, z_j)$, so $\mathsf{ar}(S_j, s, u_j) \in \mathcal{A}'$. Since u_j is a named individual, by the definition of $\Gamma(\mathcal{A}')$ we have $\langle p_x, p_{z_j} \rangle \in S_j^{\mathcal{I}}$. Thus, $\Gamma(\mathcal{A}'), \mu \models r$.

It is thus the case that $\Gamma(\mathcal{A}') \models \mathcal{C}$. □

We next prove termination of the hypertableau calculus.

Lemma 12 (Termination) *For a set of HT-clauses $\mathcal{C}$ and an input ABox $\mathcal{A}$, let $|\mathcal{C}, \mathcal{A}|$ be the sum of the number of assertions in $\mathcal{A}$, the number of concepts and roles in $\mathcal{C}$, and of $\lceil \log n \rceil$ for each integer n occurring in $\mathcal{C}$ or $\mathcal{A}$ in an atom of either the form $\geq n\, R.B$ or the form $y_i \approx y_j \, @^x_{\leq n\, R.B}$. Both the length of every path on each derivation for $\mathcal{C}$ and $\mathcal{A}$ and the total number of individuals introduced on each such path is at most doubly exponential in $|\mathcal{C}, \mathcal{A}|$.*

Proof We begin by showing that the number of new individuals introduced on each derivation path is at most doubly exponential in $|\mathcal{C}, \mathcal{A}|$.

A *path of length n* between individuals s and t in an ABox $\mathcal{A}'$ is a sequence of individuals $u_0, u_1, \ldots, u_n$ such that $u_0 = s$, $u_n = t$, and, for each $0 \leq i \leq n-1$, either $R_i(u_i, u_{i+1}) \in \mathcal{A}'$ or $R_i(u_{i+1}, u_i) \in \mathcal{A}'$ for $R_0, \ldots R_{n-1}$ atomic roles.

A *root path* for a root individual t in an HT-ABox $\mathcal{A}'$ is a path between t and a named individual s such that all intermediate individuals u_i, $1 \leq i \leq n-1$, are root individuals. The *level* $\mathsf{lev}(t)$ of t is the length of the shortest root path for t. Thus, $\mathsf{lev}(t) = 0$ if t is a named individual.

The *depth* $\mathsf{dep}(t)$ of an individual t is the number of ancestors of t. Thus, $\mathsf{dep}(t) = 0$ if t is a root individual. Due to Property (5) of HT-ABoxes, if an individual t occurs in an HT-ABox $\mathcal{A}'$, then $\mathcal{A}'$ contains a path of length $\mathsf{dep}(t)$ between a root individual s and t such that the individuals u_i, $0 \leq i \leq n-1$, are all ancestors of t; since each individual has at most one predecessor, these u_i are also the only ancestors of t.

We now show that the maximum level of a root individual and the maximum depth of every individual are both at most exponential in $|\mathcal{C}, \mathcal{A}|$.

An application of a derivation rule never increases the level of an individual. This is because a named individual is never pruned and can be merged only into another named individual,[3] and a root individual can be merged only into another root individual. Such rule applications can only make a root path shorter, and not longer.

Let m be the number of atomic concepts and n the number of atomic roles that occur in $\mathcal{A}$ and $\mathcal{C}$, let $\wp = 2^{2m+2n} + 1$, and let $\mathcal{A}'$ be an ABox labeling a node of a derivation for $\mathcal{A}$ and $\mathcal{C}$. We next show that (1) $\mathsf{dep}(t) \leq \wp$ for each individual t occurring in $\mathcal{A}'$, and (2) if t is a root individual, then $\mathsf{lev}(t) \leq \wp$.

(Claim 1) For a pair of individuals s and t occurring in $\mathcal{A}'$, there are 2^m different possible labels $\mathcal{L}_{\mathcal{A}'}(s)$ and 2^n different possible labels $\mathcal{L}_{\mathcal{A}'}(s,t)$. Thus, if $\mathcal{A}'$ contains at least $\wp = 2^m \cdot 2^m \cdot 2^n \cdot 2^n + 1$ predecessor-successor pairs of blockable individuals, then $\mathcal{A}'$ must contain two pairs $\langle s, s.i\rangle$ and $\langle t, t.j\rangle$ such that the following conditions are satisfied:

$$\begin{array}{rclcrcl} \mathcal{L}_{\mathcal{A}'}(s.i) & = & \mathcal{L}_{\mathcal{A}'}(t.j) & \qquad & \mathcal{L}_{\mathcal{A}'}(s) & = & \mathcal{L}_{\mathcal{A}'}(t) \\ \mathcal{L}_{\mathcal{A}'}(s, s.i) & = & \mathcal{L}_{\mathcal{A}'}(t, t.j) & & \mathcal{L}_{\mathcal{A}'}(s.i, s) & = & \mathcal{L}_{\mathcal{A}'}(t.j, t) \end{array}$$

Since $\prec$ contains the ancestor relation, a path in $\mathcal{A}'$ containing $\wp$ blockable individuals must include at least one blocked individual, so a blockable individual of depth $\wp$ must

[3] If a derivation rule were to replace a named individual with an individual that is not named, the levels of other root individuals could increase; see Figure 3.7 and the related discussion in Section 3.5.1.

be blocked. The $\geq$-rule is applied only to individuals that are not blocked, so the rule cannot introduce an individual u such that $\mathsf{dep}(u) > \wp$.

(Claim 2) We show that the following stronger claim (*) holds for each root individual s occurring in an assertion in $\mathcal{A}'$ (the symmetry of $\approx$ applies as usual):

1. $\mathsf{lev}(s) \leq \wp$;
2. if $R(s,t) \in \mathcal{A}'$ or $R(t,s) \in \mathcal{A}'$ or $t \approx u\,@^{s}_{\leq n\,R.B} \in \mathcal{A}'$ with t a blockable nonsuccessor of s, then $\mathsf{lev}(s) + \mathsf{dep}(t) \leq \wp$; and
3. if $s \approx t \in \mathcal{A}'$ with t a blockable nonsuccessor of s (where the equality can be annotated), then $\mathsf{lev}(s) + \mathsf{dep}(t) \leq \wp + 1$.

This claim is clearly true for the input ABox $\mathcal{A}$ labeling the root of a derivation, which contains only named individuals. We now assume that (*) holds for some ABox $\mathcal{A}'$ and consider all possible derivation rules that can be applied to $\mathcal{A}'$.

- Assume that the *Hyp*-rule derives an assertion $R(s,t)$ or $R(t,s)$, where s is a root individual and t is a blockable nonsuccessor of s. Let $R(x,y)$ or $R(y,x)$ be the atom from the consequent of an HT-clause r that is instantiated by the derivation rule. We have the following two possibilities for the antecedent of r.
  - The antecedent of r contains an atom of the form $S(x,y)$ or $S(y,x)$ that is matched to an assertion of the form $S(s,t)$ or $S(t,s)$ in $\mathcal{A}'$. Since $\mathcal{A}'$ satisfies (*), the resulting ABox satisfies (*) as well.
  - The antecedent of r contains an atom of the form $O_a(x)$ or $O_a(y)$ that is matched to an assertion of the form $O_a(s)$ in $\mathcal{A}'$ (since t is blockable, $\mathcal{A}'$ cannot contain $O_a(t)$ by Property 3 of HT-ABoxes). Then $\mathsf{dep}(t) \leq \wp$ and $\mathsf{lev}(s) = 0$, so the resulting ABox satisfies (*) as well.

- Assume that the *Hyp*-rule derives an assertion $t \approx u \,@^{s}_{\leq n\, R.B}$, where s is a root individual and t is a blockable nonsuccessor of s. By Definition 11, the antecedent of the HT-clause then contains atoms of the form $\mathsf{ar}(R, x, y_i)$ and $\mathsf{ar}(R, x, y_j)$ that are matched to assertions $\mathsf{ar}(R, s, t)$ and $\mathsf{ar}(R, s, u)$ in $\mathcal{A}'$. Since $\mathcal{A}'$ satisfies (*), we have $\mathsf{lev}(s) + \mathsf{dep}(t) \leq \wp$, so the resulting equality satisfies Item 2 of (*). To show that $t \approx u \,@^{s}_{\leq n\, R.B}$ satisfies Item 3 of (*), assume that u is a root individual and t is a nonsuccessor of u. Since $\mathcal{A}'$ contains $\mathsf{ar}(R, s, u)$, we have that $\mathsf{lev}(u) \leq \mathsf{lev}(s) + 1$; but then, $\mathsf{lev}(u) + \mathsf{dep}(t) \leq \wp + 1$, as required.

- If the *Hyp*-rule derives an assertion $s \approx t$, where s is a root individual and t is a blockable nonsuccessor of s, the only remaining possibility is that the consequent of the HT-clause then contains the equality $x \approx z_j$. By Definition 11, the antecedent then contains $O_a(z_j)$ that is matched to an assertion $O_a(s)$ in $\mathcal{A}'$, where s is a named individual. Then $\mathsf{dep}(t) \leq \wp$ and $\mathsf{lev}(s) = 0$, so the resulting ABox satisfies (*).

- Assume that the $\geq$-rule introduces an assertion of the form $R(s, t)$ or $R(t, s)$ where t is a fresh individual. Individual t is always a successor of s, so the resulting ABox trivially satisfies (*).

- Assume that the $\approx$-rule is applied to an assertion of the form $u \approx s$ and that u is merged into s. By the definition of merging, we have that $\mathsf{dep}(u) \geq \mathsf{dep}(s)$ and u is pruned. If s is a blockable individual, then u is blockable as well, and the resulting ABox satisfies (*) because u is replaced with an individual of equal or smaller depth. Therefore, we assume that s is a root individual and consider the types of assertions that can be added to $\mathcal{A}'$ as a result of merging.

  - If $R(u, u)$ is changed into $R(s, s)$, the resulting ABox clearly satisfies (*).

– Assume that $R(u,t)$ where t is a root individual is changed into $R(s,t)$. This inference can make root paths to s and t only shorter and not longer, so the levels of s and t can only decrease rather than increase. Thus, the resulting ABox satisfies Item 1 of (*).

– Assume that $R(u,t)$, where t is a predecessor of u, is changed into $R(s,t)$; the only nontrivial case is when t is a blockable nonsuccessor of s. Since t is a predecessor of u, we have $\mathsf{dep}(t)+1=\mathsf{dep}(u)$; since $\mathcal{A}'$ satisfies (*), we have $\mathsf{lev}(s)+\mathsf{dep}(u)\leq\wp+1$; but then, $\mathsf{lev}(s)+\mathsf{dep}(t)\leq\wp$ as required.

– The cases when $R(t,u)$ is changed into $R(t,s)$ are analogous.

– Assume that a possibly annotated equality $v\approx u$ is changed into $v\approx s$. The only nontrivial case is when v is a blockable nonsuccessor of s. If u is a root individual, then the level of s after merging is bounded by $\min(\mathsf{lev}(s),\mathsf{lev}(u))$ before merging, so (*) is preserved. If u and v are both blockable individuals, then by Property (2) of HT-ABoxes, either u is an ancestor of v, or u and v are siblings, or v is an ancestor of u. If u is an ancestor of v, then pruning u removes $v\approx u$ from $\mathcal{A}'$. If v is a sibling or an ancestor of u, then u must be a nonsuccessor of s, so $\mathsf{lev}(s)+\mathsf{dep}(u)\leq\wp+1$; but then, $\mathsf{dep}(v)\leq\mathsf{dep}(u)$, so $\mathsf{lev}(s)+\mathsf{dep}(v)\leq\wp+1$ and (*) is preserved.

– Assume that $v\approx v'\,@^{u}_{\leq n\,R.B}$ is changed into $v\approx v'\,@^{s}_{\leq n\,R.B}$ or $v\approx s\,@^{s}_{\leq n\,R.B}$. The only nontrivial case is when v is a blockable nonsuccessor of s. Since u is pruned before merging, by Properties (2) and (4) or HT-ABoxes v must be a predecessor of u, so $\mathsf{dep}(v)+1=\mathsf{dep}(u)$. Furthermore, by the same properties u must be a blockable nonsuccessor of s, so $\mathsf{lev}(s)+\mathsf{dep}(u)\leq\wp+1$. But then, $\mathsf{lev}(s)+\mathsf{dep}(v)\leq\wp$, as required.

- An application of the $\bot$-rule trivially preserves (*).

- Assume that the *NI*-rule is applied to an assertion $s \approx t\,@^{u}_{\leq n\,R.B}$ replacing s with a root individual $v = \|u.\langle R, B, i\rangle\|_{\mathcal{A}'}$. If v already occurs in an assertion in $\mathcal{A}'$, then v satisfies Item 1 of (*). If, however, v is fresh, by Property (4) of HT-ABoxes v will be connected to u by a role assertion, so $\mathsf{lev}(v) \leq \mathsf{lev}(u) + 1$. Furthermore, since s is a blockable nonsuccessor of u, we have $\mathsf{lev}(u) + \mathsf{dep}(s) \leq \wp$. Finally, since s is blockable, $\mathsf{dep}(s) \geq 1$, so $\mathsf{lev}(u) \leq \wp - 1$. As a consequence, we conclude that $\mathsf{lev}(v) \leq \wp$, which proves Item 1 of (*). The proof that the assertions introduced through merging satisfy (*) is analogous to the case for the $\approx$-rule.

We now complete the proof that the total number of individuals introduced by derivation rules is at most doubly exponential in $|\mathcal{C}, \mathcal{A}|$.

All named individuals are of level 0 and are never introduced by the derivation rules. An application of the *NI*-rule to a root individual u of level ℓ can introduce at most n root individuals of level $\ell+1$ for each concept $\leq n\,R.B$ that occurs in $\mathcal{C}$. Thus, for each named individual, the derivation rules can create a tree of root individuals. The maximum depth of the tree is $\wp$, which is exponential in $|\mathcal{C}, \mathcal{A}|$. Furthermore, the maximum branching factor b is equal to the sum of all numbers occurring in $\mathcal{C}$ in atoms of the form $y_i \approx y_j\,@^{x}_{\leq n\,R.B}$ (i.e. one application of the $\geq$-rule for each such concept). Clearly, b is exponential in $|\mathcal{C}, \mathcal{A}|$, so each such tree is doubly exponential in $|\mathcal{C}, \mathcal{A}|$. (If numbers were coded in unary, then the branching factor would be polynomial, but each such tree would still be doubly exponential in $|\mathcal{C}, \mathcal{A}|$.)

Similarly, each root individual can become the root of a tree of blockable individuals of depth $\wp$. Each blockable individual is introduced by applying the $\geq$-rule to its predecessor. Furthermore, the $\geq$-rule can be applied to an individual s at most once for each concept of the form $\geq n\,R.B$. Thus, the branching factor is exponential assuming binary coding of numbers, and each such tree is at most doubly exponential in $|\mathcal{C}, \mathcal{A}|$.

Thus, the total number of individuals appearing in each path of a derivation is at most doubly exponential in $|\mathcal{C}, \mathcal{A}|$.

To find a bound on the length of each path of a derviation, we first observe that once all assertions involving an individual have been removed due to merging or pruning, no assertion involving that individual is ever reintroduced on the same derivation path. For root individuals, this is a consequence of the fact that if a root individual s is removed from an ABox $\mathcal{A}'$ due to merging, then a renaming is added to $\mathcal{A}'$ that ensures $\|s\|_{\mathcal{A}'} \neq s$. Once a renaming is added to $\mathcal{A}'$, all ABoxes occurring below $\mathcal{A}'$ in a derivation will contain this renaming as well, so no subsequent application of the *NI*-rule can reintroduce s. For other individuals, we simply note that the supply of blockable individuals is infinite, so we can assume that each fresh individual introduced by the $\geq$-rule is unique.

Each application of a derivation rule either merges one individual into another or introduces at least one assertion not previously present in $\mathcal{A}'$. Since merging removes an individual and there are at most doubly-exponentially many individuals, there can be at most doubly-exponentially many rule applications resulting in a merge. Further, each assertion involves at most two individuals, the number of distinct assertions about a pair of individuals is polynomial in $|\mathcal{C}, \mathcal{A}|$, and assertions are only removed as a result of pruning, so there can be at most doubly-exponentially many rule applications resulting in fresh assertions. □

We now state the main theorem of this chapter.

Theorem 13 *The satisfiability of a $\mathcal{SROIQ}$ knowledge base $\mathcal{K}$ can be decided by computing $\mathcal{K}' = \Delta(\Omega(\mathcal{K}))$ and then checking whether some derivation for $\Xi(\mathcal{K}')$ contains a leaf node labeled with a clash-free ABox. Such an algorithm can be implemented such that it runs in nondeterministic triple exponential time with respect to $|\mathcal{K}|$.*

Proof The first part of the theorem follows immediately from Lemmas 5, 7, 10, and 11. For the second part, preprocessing produces at worst an exponential increase in size and number of symbols from $|\mathcal{K}|$ to $|\Xi(\mathcal{K}')|$. By Lemma 12, the total number of individuals in each ABox is at most doubly exponential in $|\Xi(\mathcal{K}')|$, thus clearly each rule application runs in nondeterministic double exponential time in $|\Xi(\mathcal{K}')|$, and each derivation path contains a number of steps which is at mostly doubly exponential in $|\Xi(\mathcal{K}')|$. The existence of a leaf derivation node labeled with a clash-free ABox can thus be checked by nondeterministically applying the hypertableau derivation rules to construct an ABox, and this procedure runs in nondeterministic triple exponential time with respect to $|\mathcal{K}|$. □

Chapter 6

Optimizations

While the encodings and calculus described in Chapter 5 have been shown to provide a novel theoretical framework that can be used to search for models of $\mathcal{SROIQ}$ knowledge bases, our primary goal is to develop techniques that lend themselves to efficient implementation. In this chapter we we discuss the possibilities of optimizing the blocking condition to single and subset blocking; furthermore, we argue that modifying the algorithm to make it optimal w.r.t. worst-case complexity might be difficult.

6.1 Caching Blocking Labels

Let $\mathcal{T}$ and $\mathcal{R}$ be a $\mathcal{SHIQ}^+$ TBox and RBox, respectively, and let $\mathcal{C} = \Xi_{\mathcal{TR}}(\mathcal{T} \cup \mathcal{R})$; since $\mathcal{T}$ does not contain nominals, no assertions involving nominal guard concepts are needed. Furthermore, assume that the classification of $\mathcal{T} \cup \mathcal{R}$ involves n calls to the hypertableau algorithm for $(\{A_i(a_i), \neg B_i(a_i)\}, \mathcal{C})$. Then, if a derivation for $(\{A_i(a_i), \neg B_i(a_i)\}, \mathcal{C})$ contains a leaf node labeled with a clash-free ABox $\mathcal{A}_i$, we can use the nonblocked individuals from $\mathcal{A}_i$ as blockers in all subsequent satisfiability checks of $(\{A_j(a_j), \neg B_j(a_j)\}, \mathcal{C})$ for $j > i$.

This is a simple consequence of the following fact. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be two models of $\mathcal{T} \cup \mathcal{R}$ such that $\triangle^{\mathcal{I}_1} \cap \triangle^{\mathcal{I}_2} = \emptyset$; furthermore, let $\mathcal{I}$ be defined as $\triangle^{\mathcal{I}} = \triangle^{\mathcal{I}_1} \cup \triangle^{\mathcal{I}_2}$, $A^{\mathcal{I}} = A^{\mathcal{I}_1} \cup A^{\mathcal{I}_2}$, and $R^{\mathcal{I}} = R^{\mathcal{I}_1} \cup R^{\mathcal{I}_2}$, for each atomic concept A and each atomic role

R. Then, by a simple induction on the structure of axioms in $\mathcal{T} \cup \mathcal{R}$, it is trivial to show that $\mathcal{I} \models \mathcal{T} \cup \mathcal{R}$. This property does not hold in the presence of nominals, which can impose a bound on the number of elements in the interpretation of a concept; the bound could be satisfied in $\mathcal{I}_1$ and $\mathcal{I}_2$ individually, but violated in $\mathcal{I}$.

Our optimization is correct because, instead of $(\{A_i(a_i), \neg B_i(a_i)\}, \mathcal{C})$, we can check the satisfiability of $(\mathcal{A}_i \cup \{A_i(a_i), \neg B_i(a_i)\}, \mathcal{C})$, and in doing so we can use the individuals from $\mathcal{A}_i$ as potential blockers due to anywhere blocking. This optimization can be seen as a very simple form of model caching [Horrocks, 2007], and it has been key to obtaining the results that we present in Chapter 12. For example, on GALEN only one subsumption test is costly because it computes a substantial part of a model of the TBox; all subsequent subsumption tests reuse large parts of that model.

In practice, we do not need to keep the entire ABox $\mathcal{A}_i$ around; rather, for each nonblocked blockable individual t with a predecessor t', we simply need to retain the sets $\mathcal{L}_{\mathcal{A}_i}(t)$, $\mathcal{L}_{\mathcal{A}_i}(t')$, $\mathcal{L}_{\mathcal{A}_i}(t, t')$, and $\mathcal{L}_{\mathcal{A}_i}(t', t)$.

6.2 Single Blocking

For DLs such as $\mathcal{SHOQ}^+$ that do not provide for inverse roles, pairwise blocking can be weakened to atomic single blocking, defined as follows.

Definition 16 (Atomic Single Blocking) *Atomic single blocking* is obtained from pairwise blocking (see Definition 13) by changing the notion of direct blocking: a blockable individual s is directly blocked by a blockable individual t if and only if t is not blocked, $t \prec s$, and $\mathcal{L}_{\mathcal{A}}(s) = \mathcal{L}_{\mathcal{A}}(t)$ for $\mathcal{L}_{\mathcal{A}}(s)$ as in Definition 13.[1] △

In some cases, this simpler blocking condition can make the hypertableau algorithm construct smaller ABoxes, which can lead to increased efficiency. We next formalize the notion of HT-clauses to which atomic single blocking is applicable.

[1] The name "atomic" reflects the fact that $\mathcal{L}_{\mathcal{A}}(s)$ contains only atomic concepts.

Definition 17 (Simple HT-Clause) An HT-clause r is *simple* if it satisfies the following restrictions, for x a center variable, y_i a branch variable, z_j a nominal variable, B a literal concept, and R an atomic role:

- Each atom in the antecedent of r is of the form $A(x)$, $R(x,x)$, $R(x,y_i)$, $A(y_i)$, or $A(z_j)$.
- Each atom in the consequent of r is of the form $B(x)$, $\geq h\, R.B(x)$, $B(y_i)$, $R(x,x)$, $R(x,y_i)$, $R(x,z_j)$, $x \approx z_j$, or $y_i \approx y_j$. △

It is straightforward to see that, if $\mathcal{K}$ is a $\mathcal{SHOQ}^+$ knowledge base, then $\Xi_{\mathcal{TR}}(\mathcal{K})$ contains only simple HT-clauses. The completeness of the hypertableau algorithm with atomic single blocking on simple HT-clauses is straightforward to show.

Lemma 14 *Let $\mathcal{C}$ be a set of simple HT-clauses, and $\mathcal{A}$ an input ABox. If a derivation with atomic single blocking for $\mathcal{C}$ and $\mathcal{A}$ exists in which a leaf node is labeled with a clash-free ABox $\mathcal{A}'$, then $(\mathcal{C},\mathcal{A})$ is satisfiable.*

Proof By slightly modifying the proof of Lemma 9, it is possible to show the following property (*): each atom in $\mathcal{A}'$ involving an atomic role is of the form $R(s,a)$, $R(s,s)$, or $R(s,s.i)$, for a a named individual and s any individual.

Let $\mathcal{I}$ be a model constructed in the same way as in Lemma 11, but by using single blocking. Due to (*), whenever $\langle p_1,p_2\rangle \in R^{\mathcal{I}}$, then p_2 is either of the form $[\frac{a}{a}]$ for a a named individual, it is a successor of p_1, or $p_2 = p_1$. The proof that $\mathcal{I}$ is a model of $(\mathcal{C},\mathcal{A})$ is a straightforward consequence of the following observations about the proof of Lemma 11:

- In the proof that $\geq n\, R.B(s) \in \mathcal{A}'$ implies $p_s \in (\geq n\, R.B)^{\mathcal{I}}$, individual u_i can never be a blockable predecessor of s. Thus, labels $\mathcal{L}_{\mathcal{A}'}(s,u_i)$, $\mathcal{L}_{\mathcal{A}'}(u_i,s)$, and $\mathcal{L}_{\mathcal{A}'}(u_i)$ are never relevant.

- In the proof that $\mathcal{I} \models \mathcal{C}$, it is not possible that s' is blocked and t'_i is a predecessor of s'. Thus, labels $\mathcal{L}_{\mathcal{A}'}(s, t_i)$, $\mathcal{L}_{\mathcal{A}'}(t_i, s)$, and $\mathcal{L}_{\mathcal{A}'}(t_i)$ are never relevant.

The proof that $\mathcal{I}$ is a model of $(\mathcal{A}, \mathcal{C})$ thus requires only $\mathcal{L}_{\mathcal{A}'}(s) = \mathcal{L}_{\mathcal{A}'}(t)$ to hold when s is blocked by the blocker t; hence, $\mathcal{I}$ is a model of $(\mathcal{A}, \mathcal{C})$ even if atomic single blocking is used. □

The following variant of single blocking can also be applied to DLs with inverse roles but no number restrictions, such as $\mathcal{SHOI}$.

Definition 18 (Full Single Blocking) *Full single blocking* is obtained from atomic single blocking (see Definition 16) by changing the definition of $\mathcal{L}_{\mathcal{A}}(s)$ as follows:

$$\mathcal{L}_{\mathcal{A}}(s) = \{\, C \mid C(s) \in \mathcal{A} \text{ where } C \text{ is of the form } A \text{ or } \geq 1\, R.B \text{ with } A \text{ an atomic and } B \text{ a literal concept} \,\}$$

△

For t to directly block s in $\mathcal{A}$ under atomic single blocking, it suffices if s and t occur in the same atomic concepts in $\mathcal{A}$. Intuitively, this is because the model construction from Lemma 11 "copies" all nonatomic concepts from t to s; hence, assertions of the form $C(s)$ where C is not atomic are not relevant. In contrast, in full single blocking, s and t must occur in $\mathcal{A}$ in *exactly* the same concepts (apart from negated atomic concepts). Intuitively, given a clash-free ABox $\mathcal{A}'$ to which no derivation rule is applicable, a model for $(\mathcal{A}, \mathcal{C})$ is constructed from $\mathcal{A}'$ by replacing s with t; for the result to be a model, the two individuals must occur in exactly the same concepts.

Full single blocking must be applied with care in the hypertableau setting. Consider the following knowledge base $\mathcal{K}_{11}$, consisting of an ABox $\mathcal{A}_{11}$ and a set of HT-clauses $\mathcal{C}_{11}$.

$$\mathcal{A}_{11} = \{\, \exists T.C(a) \,\}$$

$$\mathcal{C}_{11} = \{C(x) \rightarrow \exists R.D(x),\ D(x) \rightarrow \exists S^{-}.C(x),\ R(x, y_1) \wedge S(x, y_2) \rightarrow \bot\}$$

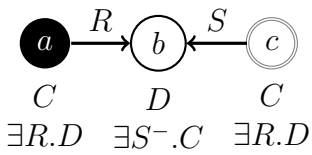


Figure 6.1: Problems with Single Blocking

On $\mathcal{K}_{11}$, the hypertableau algorithm with full single blocking produces the ABox shown in Figure 6.1. The individual d is blocked by b, so the algorithm terminates; an expansion of $\exists R.D(d)$, however, would reveal that $\mathcal{K}_{11}$ is unsatisfiable. The problem arises because the HT-clause $R(x, y_1) \wedge S(x, y_2) \rightarrow \bot$ contains two role atoms, which allows the HT-clause to examine both the successor and the predecessor of x. Full single blocking, however, does not ensure that both predecessors and successors of x have been fully built. We can correct this problem by requiring the normalized GCIs to contain at most one $\forall R.C$ concept. For example, if we replace our HT-clause with $R(x, y_1) \rightarrow Q(x)$ and $Q(x) \wedge S(x, y_2) \rightarrow \bot$, then the first HT-clause would additionally derive $Q(b)$, so d would not be blocked by b.

We can apply full single blocking to the DL $\mathcal{SHOI}$ provided that each HT-clause contains at most one role atom in the antecedent. We can always ensure this by suitably renaming complex concepts with atomic ones.

Lemma 15 *Let $\mathcal{A}$ be an ABox and $\mathcal{C}$ a set of HT-clauses such that, for each $r \in \mathcal{C}$, (*i*) r contains no atoms of the form $R(x, x)$, (*ii*) the antecedent of r contains at most one role atom, and (*iii*) all at-least restriction concepts are of the form $\geq 1\, S.B$ for S a role and B a literal concept. If a derivation with full single blocking for $\mathcal{C}$ and $\mathcal{A}$ exists in which a leaf node is labeled with a clash-free ABox $\mathcal{A}'$, then $(\mathcal{C}, \mathcal{A})$ is satisfiable.*

Proof Let $\mathcal{A}''$ be obtained from $\mathcal{A}'$ by removing each assertion containing an indirectly blocked individual. Since no derivation rule is applicable to indirectly blocked

individuals, no derivation rule is applicable to $\mathcal{A}''$ and $\mathcal{C}$. For an individual s occurring in $\mathcal{A}''$, let $[s]_{\mathcal{A}''} = s$ if s is not blocked in $\mathcal{A}''$, and let $[s]_{\mathcal{A}''} = s'$ if s is blocked in $\mathcal{A}''$ by the blocker s'.

Note the following useful property (*): if $\neg A(s) \in \mathcal{A}''$, then $A(s) \notin \mathcal{A}''$ since the $\bot$-rule is not applicable to $\mathcal{A}''$; but then, $A([s]_{\mathcal{A}''}) \notin \mathcal{A}''$ as well.

We now construct an interpretation $\mathcal{I}$ from $\mathcal{A}''$ as follows.

$$\begin{aligned}
\triangle^{\mathcal{I}} =& \{s \mid s \text{ occurs in } \mathcal{A}'' \text{ and it is not blocked in } \mathcal{A}''\} \\
s^{\mathcal{I}} =& [s]_{\mathcal{A}''} \text{ for each individual } s \text{ occurring in } \mathcal{A}'' \\
A^{\mathcal{I}} =& \{[s]_{\mathcal{A}''} \mid A(s) \in \mathcal{A}''\} \\
R^{\mathcal{I}} =& \{\langle [s]_{\mathcal{A}''}, [t]_{\mathcal{A}''} \rangle \mid R(s,t) \in \mathcal{A}''\}
\end{aligned}$$

It is straightforward to see that $\mathcal{I} \models \mathcal{A}''$. Consider now each HT-clause $r \in \mathcal{C}$ that contains in the antecedent one atom of the form $R(x, y)$, as well as atoms of the form $A_i(x)$, $B_i(y)$, $C_i(z_i)$. Let σ be a mapping from the variables of r to the individuals in $\mathcal{A}''$ such that $\mathcal{I} \models \sigma(U_i)$ for each atom U_i from the antecedent of r. By the definition of $\mathcal{I}$, individuals s and t then exist such that $R(s,t) \in \mathcal{A}''$, $\sigma(x) = [s]_{\mathcal{A}''}$, and $\sigma(y) = [t]_{\mathcal{A}''}$. By the definition of full single blocking, then $A_i(s) \in \mathcal{A}''$ and $B_i(t) \in \mathcal{A}''$ as well. Furthermore, since each z_i occurs in a nominal guard concept, $\sigma(z_i)$ is a named individual. Let σ' be such that $\sigma'(x) = s$, $\sigma'(y) = t$, and $\sigma'(z_i) = \sigma(z_i)$. Since the *Hyp*-rule is not applicable to $\mathcal{C}$ and $\mathcal{A}''$ for σ', we have $\sigma'(V_j) \in \mathcal{A}''$ for some atom V_j from the consequent of r. Consider now the possible forms that V_j can have.

- If $V_j = S(x, y)$, then $\mathcal{I} \models S(\sigma(x), \sigma(y))$ by the definition of $\mathcal{I}$. The case $V_j = S(y, x)$ is analogous.
- If $V_j = A(x)$ for A an atomic concept, then $A([s]_{\mathcal{A}''}) \in \mathcal{A}''$ by the definition of full single blocking; but then, $\mathcal{I} \models A(\sigma(x))$ by the definition of $\mathcal{I}$. The case when $V_j = A(y)$ is analogous.

- If $V_j = \neg A(x)$, then $A([s]_{\mathcal{A}''}) \notin \mathcal{A}''$ by (*); but then, by the definition of $\mathcal{I}$ we have $\mathcal{I} \models \neg A(\sigma(x))$. The case when $V_j = \neg A(y)$ is analogous.

- If $V_j = D(x)$ for $D = \geq 1\, R.B$, then $D([s]_{\mathcal{A}''}) \in \mathcal{A}''$ by the definition of full single blocking. Since the $\geq$-rule is not applicable to $[s]_{\mathcal{A}''}$, an individual t exists such that $\mathsf{ar}(R, s, t) \in \mathcal{A}''$ and if B is atomic, then $B(t) \in \mathcal{A}''$, and if $B = \neg A$, then $A(t) \notin \mathcal{A}''$. By the definition of full single blocking, if B is atomic, then $B([t]_{\mathcal{A}''}) \in \mathcal{A}''$, and if $B = \neg A$, then $A([t]_{\mathcal{A}''}) \notin \mathcal{A}''$. By the definition of $\mathcal{I}$, we have $\langle [s]_{\mathcal{A}''}, [t]_{\mathcal{A}''} \rangle \in R^{\mathcal{I}}$, and $[t]_{\mathcal{A}''} \in B^{\mathcal{I}}$; therefore, $\mathcal{I} \models D(\sigma(x))$. The case when $V_j = D(y)$ is analogous.

- If $V_j = x \approx z_i$, then $\sigma'(x) \approx \sigma'(z_i) \in \mathcal{A}''$; since the $\approx$-rule is not applicable to $\mathcal{A}''$, we have $\sigma'(x) = \sigma'(z_i)$. But then, since named individuals cannot block other individuals, we have $\sigma(x) = \sigma'(x)$; hence, $\mathcal{I} \models \sigma(x) \approx \sigma(z_i)$.

Thus, in all cases we have $\mathcal{I} \models \sigma(V_j)$. The case when r does not contain a role atom $R(x, y)$ in the antecedent is analogous, so $\mathcal{I} \models (\mathcal{A}, \mathcal{C})$. □

6.3 Subset Blocking

In tableau algorithms for DLs without inverse roles, full single blocking condition from Definition 18 can be further weakened to *full subset blocking* [Baader *et al.*, 1996].

Definition 19 (Full Subset Blocking) *Full subset blocking* is obtained from full single blocking (see Definition 18) by changing the notion of direct blocking: a blockable individual s is directly blocked by an individual t if and only if t is not blocked, $t \prec s$, and $\mathcal{L}_{\mathcal{A}}(s) \subseteq \mathcal{L}_{\mathcal{A}}(t)$. △

Full subset blocking is problematic in the hypertableau setting. Consider the knowledge base $\mathcal{K}_{12}$ that consists of an ABox $\mathcal{A}_{12}$ and a TBox corresponding to

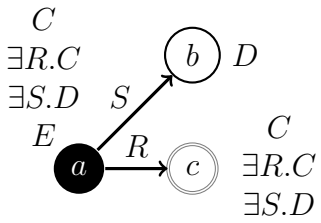


Figure 6.2: Problems with Full Subset Blocking

the HT-clauses $\mathcal{C}_{12}$.

$$\begin{aligned} \mathcal{A}_{12} &= \{\, \exists T.C(a) \,\} \\ \mathcal{C}_{12} &= \left\{ \begin{array}{ll} C(x) \rightarrow \exists R.C(x), & C(x) \rightarrow \exists S.D(x), \\ S(x,y) \wedge D(y) \rightarrow E(x), & R(x,y) \wedge E(y) \rightarrow \bot \end{array} \right\} \end{aligned}$$

On $\mathcal{K}_{12}$, our algorithm can produce the ABox shown in Figure 6.2, in which d is blocked by b. If, however, we expand $\exists S.D(d)$ into $S(d,e)$ and $D(e)$, we can derive $E(d)$; together with $R(b,d)$ and the HT-clause $R(x,y) \wedge E(y) \rightarrow \bot$, we get a contradiction.

The problem arises because, in the hypertableau setting, the syntactic distinction between atomic and inverse roles is lost: an atom $R^-(x,y)$ is transformed (by the function ar) into the semantically equivalent atom $R(y,x)$. In $\mathcal{K}_{12}$, the HT-clause $S(x,y) \wedge D(y) \rightarrow E(x)$ can be seen as including an implicit inverse role, because it examines a successor of x in the antecedent in order to derive new information about x in the consequent, thus mimicking the behavior of tableau algorithms with the semantically equivalent GCI $D \sqsubseteq \forall S^-.E$.

The semantically equivalent but inverse-free GCI $\exists S.D \sqsubseteq E$ would, in our hypertableau algorithm, be transformed into exactly the same HT-clause. In the tableau setting, however, this GCI would be treated very differently: it would result in the $\sqsubseteq$-rule deriving $(E \sqcup \forall S.\neg D)(s)$ for all individuals s. A similar effect could be achieved in the hypertableau setting by translating $\exists S.D \sqsubseteq E$ into two HT-clauses: $\top \rightarrow E(x) \vee Q(x)$ and $Q(x) \wedge S(x,y) \wedge D(y) \rightarrow \bot$. This introduces nondeterminism, but solves the problem with full subset blocking by deriving either $E(c)$ or $Q(c)$, the first of which leads

to an immediate contradiction, and the second of which delays blocking.

In general, it is easy to see that full subset blocking could be used in the hypertableau setting by modifying the preprocessing phase so as to ensure that HT-clauses do not include such implicit inverses. It is not clear, however, if this would be very useful: it would result in (possibly) smaller ABoxes, but at the cost of (possibly) larger derivation trees.

6.4 The Number of Blockable Individuals

Buchheit *et al.* [1993] presented a tableau algorithm for the DL $\mathcal{ALCNR}$ which, due to anywhere blocking, runs in nondeterministic exponential time instead of nondeterministic doubly-exponential time, and Donini *et al.* [1998] presented a similar result for the basic DL $\mathcal{ALC}$. It is interesting to compare these algorithms to ours to see whether anywhere blocking can improve the worst-case complexity of our algorithm when $\mathcal{K}$ is a $\mathcal{SHIQ}^+$ knowledge base. In such a case, no HT-clause in $\Xi(\mathcal{K})$ contains a nominal guard concept, which prevents the derivation of assertions satisfying the preconditions of the *NI*-rule; hence, no new root individuals are introduced in a derivation, which eliminates a significant source of complexity.

The following example shows that, unfortunately, anywhere blocking does not improve the worst-case complexity; in fact, we identify a tension between and- and or-branching. In the example, we use the well-known encoding of binary numbers by concepts $B_0, B_1, \ldots, B_{k-1}$: we assign to each individual s in an ABox $\mathcal{A}$ a binary number $\ell_{\mathcal{A}}(s) = b_{k-1} \ldots b_1 b_0$ such that $b_i = 1$ if and only if $B_i(s) \in \mathcal{A}$. Using k concepts, we can thus encode 2^k different binary numbers. Furthermore, for any atomic role R we can ensure that whenever an individual t is an R-successor of s in $\mathcal{A}$, then $\ell_{\mathcal{A}}(t) = (\ell_{\mathcal{A}}(s) + 1) \bmod 2^k$ using the well-known *R-successor counting formula*

[Tobies, 2000]:

$$\{\ B_i \sqcap \bigsqcap_{j=0}^{i-1} B_j \ \sqsubseteq \forall R.\neg B_i, \qquad \neg B_i \sqcap \bigsqcap_{j=0}^{i-1} B_j \ \sqsubseteq \forall R.B_i \qquad \mid 0 \leq i < k\ \} \cup$$

$$\{\ B_i \sqcap \bigsqcup_{j=0}^{i-1} \neg B_j \sqsubseteq \forall R.B_i, \qquad \neg B_i \sqcap \bigsqcup_{j=0}^{i-1} \neg B_j \sqsubseteq \forall R.\neg B_i \qquad \mid 0 \leq i < k\ \}$$

Intuitively, this formula requires that if all the bits $b_0, ..., b_{i-1}$ are set to 1 in $\ell_{\mathcal{A}}(s)$, then b_i is flipped in $\ell_{\mathcal{A}}(t)$; if there exists some bit in $b_0, ..., b_{i-1}$ set to 0, then b_i remains the same in $\ell_{\mathcal{A}}(t)$.

Let $\mathcal{K}_{13}$ be the following knowledge base. For the sake of brevity, we omit the HT-clauses corresponding to the axioms in $\mathcal{K}_{13}$.

$$C(a) \tag{6.1}$$

$$C \sqsubseteq \exists L.C \sqcap \exists R.C \tag{6.2}$$

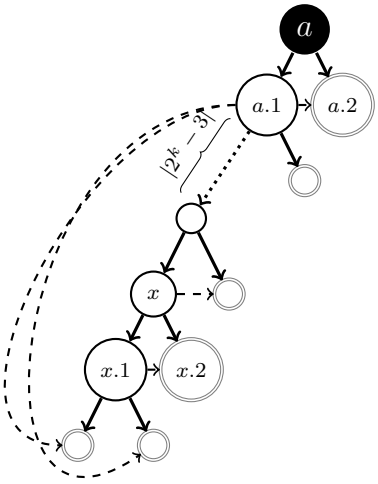
$$(\textit{The } R\textit{-successor formula for } B_0, \ldots, B_{k-1}) \tag{6.3}$$

$$(\textit{The } L\textit{-successor formula for } B_0, \ldots, B_{k-1}) \tag{6.4}$$

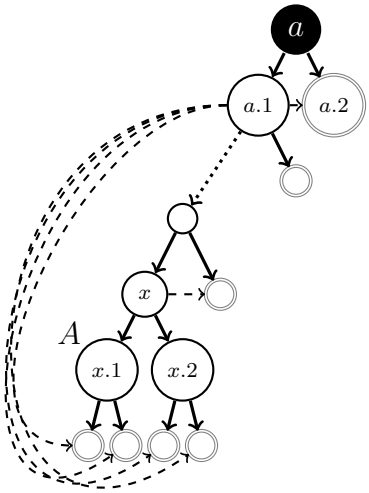
$$B_0 \sqcap \ldots \sqcap B_{k-1} \sqsubseteq A \tag{6.5}$$

$$\exists L.A \sqcap \exists R.A \sqsubseteq A \tag{6.6}$$

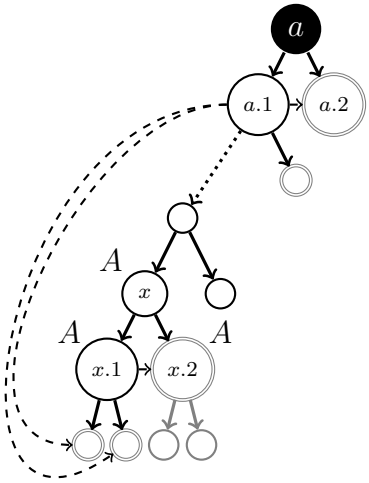
Figure 6.3 schematically presents a derivation on $\mathcal{K}_{13}$ in which a doubly exponential number of blockable individuals is introduced. For simplicity of presentation, we use single anywhere blocking. Due to (6.1)–(6.4), our algorithm can create individuals $a.1$, $a.2$, $a.1.1$, $a.1.2$, $a.1.1.1$, $a.1.1.2$, and so on, such that $s.1$ is an L-successor of s, and $s.2$ is an R-successor of s. After creating the individuals of the form $a.1^{2^k-1}.1$ and $a.1^{2^k-1}.2$ where 1^{2^k-1} is a string of $2^k - 1$ ones, each individual $x.1$ blocks $x.2$ (cf. Figure 6.3a). But then, due to (6.5), $a.1^{2^k-1}.1$ and $a.1^{2^k-1}.2$ become instances of A. By (6.6), $a.1^{2^k-1}$ is made an instance of A as well, so it does not block its sibling $a.1^{2^k-2}.2$ any more; hence, $a.1^{2^k-2}.2$ is now expanded to exponential depth (cf. Figure 6.3b). By repeating this process, the algorithm derives that $a.1^{2^k-2}$ is an



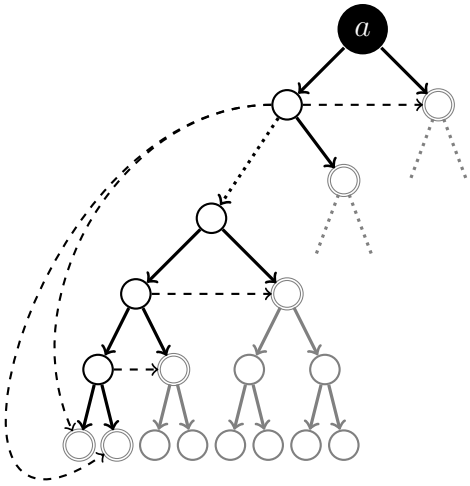
(a) An exponential path is constructed with each blockable individual blocking its sibling. No individual contains A in its label.



(b) Adding A to the label of $x.1$ unblocks $x.2$.



(c) Adding A to the label of $x.2$ makes $x.2$ blocked, and forces the addition of A to the label of x. This unblocks the sibling of x so another subtree is created.



(d) Derivation terminates with an exponential number of unblocked individuals, but a doubly-exponential number of indirectly blocked individuals.

Figure 6.3: Creation of an Exponentially Deep Binary Tree of Blockable Individuals

instance of A, but then it does not block its sibling $a.1^{2^k-3}.2$ (cf. Figure 6.3d). Eventually, the algorithm constructs a binary tree of exponential depth, thus creating a doubly-exponential number of blockable nodes in total (cf. Figure 6.3d).

Buchheit *et al.* [1993] and Donini *et al.* [1998] obtained the nondeterministic exponential behavior by applying the $\sqcap$-, $\sqcup$-, $\forall$-, and $\sqsubseteq$-rules exhaustively before applying the $\exists$-rule. Such a strategy ensures that the label of an individual s is fully constructed before introducing a successor of s, which prevents individuals from being indirectly blocked. On $\mathcal{K}_{11}$, this means that the GCI (6.6) is applied to each individual s *before* introducing its successors. Thus, before the existentials on s are expanded, the assertion $(\forall L.\neg A \sqcup \forall R.\neg A \sqcup A)(s)$ is introduced and one disjunct is chosen nondeterministically. The choices $(\forall L.\neg A)(s)$ and $(\forall R.\neg A)(s)$ will lead to a clash, so the algorithm eventually derives $A(s)$, before it expands the existentials on s and introduces $s.1$ and $s.2$. Thus, while generating at most exponential models, this algorithm incurs a massive amount of nondeterminism.

Nondeterministic exponential behavior can be guaranteed in the hypertableau algorithm by nondeterministically fixing the label of each individual before applying the $\leq$-rule to it. This technique is similar to the one used by Tobies [2001] in order to obtain a PSpace decision procedure for concept satisfiability in a DL with inverse roles but without GCIs. The performance results in Chapter 12, however, seem to suggest that this might not be beneficial in practice. Still, it might be worth exploring whether nondeterministically adding concepts to labels of individuals can be used as an optimization that would detect "early blocks" and thus prevent the construction of large models.

6.5 The Number of Root Individuals

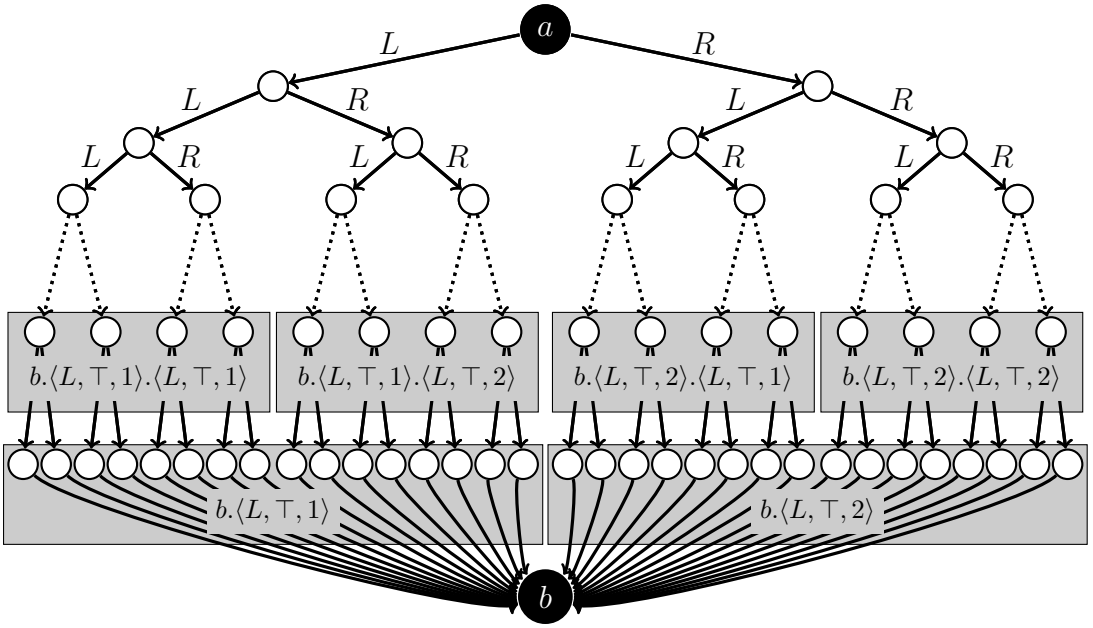
$\mathcal{SHOIQ}$ is NExpTime-complete [Tobies, 2000], and it is straightforward to extend this result to $\mathcal{ALCHOIQ}^+$; the encoding of complex role axioms presents another

exponential increase in complexity, so $\mathcal{SROIQ}$ is N2ExpTime-complete [Kazakov, 2008]. Thus, one might wonder whether the nondeterministic triple-exponential complexity result in Theorem 13 can be sharpened to obtain a worst-case optimal decision procedure. This, unfortunately, is not the case: we present an example on which our algorithm generates a number of root individuals with is doubly-exponential in the size of an $\mathcal{ALCHOIQ}^+$ knowledge base (and thus triply-exponential overall). We construct $\mathcal{K}_{14}$ by extending $\mathcal{K}_{13}$ (axioms (6.1)–(6.6)) with the following two axioms:

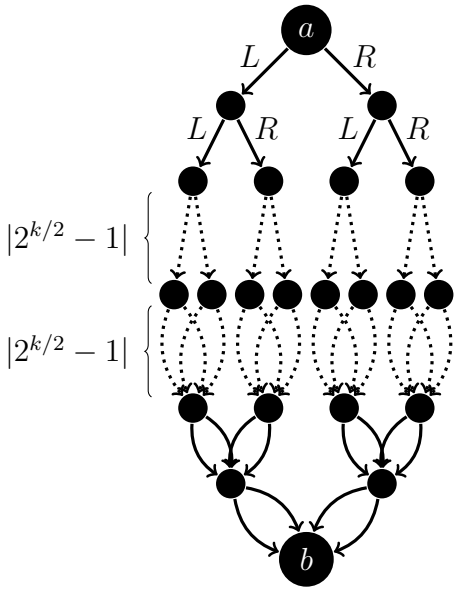
$$B_0 \sqcap \ldots \sqcap B_{k-1} \sqsubseteq \{b\} \tag{6.7}$$

$$A \sqsubseteq \leq 2\, L^-.\top \sqcap \leq 2\, R^-.\top \tag{6.8}$$

As shown in Section 6.4, the axioms of $\mathcal{K}_{13}$ can cause our algorithm to construct a binary tree of blockable individuals with exponential depth. Axiom (6.7) of $\mathcal{K}_{14}$, however, merges the leaves of this tree into the single named individual b, and axiom (6.8) ensures that the NI-rule is applied to each of the remaining blockable individuals, beginning with the neighbors of b. If, at each application of the NI-rule, we always merge blockable individuals into root individuals as shown in Figure 6.4a, then our algorithm constructs the ABox shown in Figure 6.4b, which contains two binary trees of root individuals of depth $2^{k/2}$. Unlike the case with $\mathcal{K}_{13}$, fully constructing individual labels does not avoid double-exponential behavior, since the promotion of blockable individuals to root individuals prevents blocking.



(a) A root introduction strategy for the NI-rule



(b) The resulting tree, containing a doubly-exponential number of root individuals

Chapter 7

Related Work

In this chapter we describe the relationship between our work and several other well-known reasoning techniques.

7.1 Hypertableau vs. Absorption

Absorption has been extensively used in tableau calculi to address the problems with or-branching outlined in Section 3.2 [Horrocks, 2007]. The basic absorption algorithm tries to rewrite GCIs into the form $A \sqsubseteq C$ where A is an atomic concept. After such preprocessing, instead of deriving $\neg A \sqcup C$ for each individual in an ABox, $C(s)$ is derived only if the ABox contains $A(s)$; thus, the nondeterminism introduced by the absorbed GCIs is localized. This basic technique has been refined and extended in several ways. *Negative absorption* rewrites GCIs into the form $\neg A \sqsubseteq C$ where A is an atomic concept; then, $C(s)$ is derived only if an ABox contains $\neg A(s)$ [Horrocks, 2007]. *Role absorption* rewrites GCIs into the form $\exists R.\top \sqsubseteq C$; then, $C(s)$ is derived only if an ABox contains $R(s,t)$ [Tsarkov and Horrocks, 2004]. *Binary absorption* rewrites GCIs into the form $A_1 \sqcap A_2 \sqsubseteq C$; then, $C(s)$ is derived only if an ABox contains both $A_1(s)$ and $A_2(s)$ [Hudek and Weddell, 2006].

These techniques have proven indispensable in practice; however, our analysis shows potential for further improvement. For example, the axiom $\exists R.A \sqsubseteq A$ from (3.1) cannot be absorbed directly, and applying role absorption to (3.1) produces the

axiom $\exists R.\top \sqsubseteq A \sqcup \forall R.\neg A$ containing a disjunction in the consequent. Binary absorption is not directly applicable to (3.1) since the axiom does not contain two concepts on the left-hand side of $\sqsubseteq$, but the algorithm by Hudek and Weddell [2006] additionally transforms (3.1) into an absorbable axiom $A \sqsubseteq \forall R^{-}.A$. Consider, however, the following axiom:

$$\top \sqsubseteq \forall R.\neg C \sqcup \forall S.D \tag{7.1}$$

The binary absorption algorithm can process the two disjuncts in (7.1) in two ways. If $\forall R.\neg C$ is processed before $\forall S.D$, then (7.1) is transformed into the axioms shown in (7.2), both of which can be applied deterministically in a tableau algorithm. If, however, $\forall S.D$ is processed before $\forall R.\neg C$, then (7.1) is transformed into the axioms shown in (7.3). The first axiom is absorbable, but the second is not, so a tableau algorithm will be nondeterministic.

$$C \sqsubseteq \forall R^{-}.Q_1 \qquad\qquad Q_1 \sqsubseteq \forall S.D \tag{7.2}$$

$$Q_2 \sqsubseteq \forall R.\neg C \qquad\qquad \top \sqsubseteq D \sqcup \forall S^{-}.Q_2 \tag{7.3}$$

Heuristics are used in practice to find a "good" absorption (see, e.g., [Wu and Haarslev, 2008]), but there are no guarantees that the result will incur the "least" amount of nondeterminism; this is so even on Horn knowledge bases, for which reasoning without any nondeterminism is possible in principle [Hustadt *et al.*, 2005]. In contrast, our algorithm is guaranteed to preprocesses a Horn knowledge base into Horn DL-clauses that will always result in deterministic derivations. For example, (7.1) is transformed into a Horn DL-clause (7.4).

$$R(x, y_1) \wedge C(y_1) \wedge S(x, y_2) \rightarrow D(y_2) \tag{7.4}$$

Even in the case of inherently nondeterministic knowledge bases, absorption can

be further optimized. Consider axiom (7.5), which is translated into DL-clause (7.6):

$$\top \sqsubseteq A \sqcup \forall R.B \sqcup \forall S.C \tag{7.5}$$

$$R(x, y_1) \wedge S(x, y_2) \rightarrow A(x) \vee B(y_1) \vee C(y_2) \tag{7.6}$$

The binary absorption algorithm transforms (7.5) into the following axioms:

$$Q_1 \sqcap Q_2 \sqsubseteq A \tag{7.7}$$

$$\top \sqsubseteq B \sqcup \forall R^-.Q_1 \tag{7.8}$$

$$\top \sqsubseteq C \sqcup \forall S^-.Q_2 \tag{7.9}$$

Axiom (7.7) is absorbable; however, (7.8) and (7.9) are not, so their application introduces a nondeterministic choice point for each individual occurring in an ABox. This problem can be ameliorated by using role absorption and transforming (7.8) and (7.9) into (7.10) and (7.11):

$$\exists R^-.\top \sqsubseteq B \sqcup \forall R^-.Q_1 \tag{7.10}$$

$$\exists S^-.\top \sqsubseteq C \sqcup \forall S^-.Q_2 \tag{7.11}$$

Now (7.10) can be used to derive $(B \sqcup \forall R^-.Q_1)(b)$ from $R(a, b)$, and (7.11) can be used to derive $(C \sqcup \forall S^-.Q_2)(d)$ from $S(c, d)$; however, these two disjunctions are derived even if $a \neq c$. In contrast, the DL-clause (7.6) derives a disjunction only if $a = c$; thus, literals $R(x, y_1)$ and $S(x, y_2)$ in (7.6) act as "guards." The presence of variables in the antecedent (the shared variable x in this example) makes the guards more selective than if each guard were applied in isolation. Furthermore, if $a = c$, we derive a disjunction $A(a) \vee B(b) \vee C(d)$, which involves three different individuals (a, b, and d in this case); in contrast, consequences of tableau algorithms typically involve just one individual. Thus, through the usage of variables, DL-clauses can be more global in their effect than tableau rules.

To the best of our knowledge, no known absorption technique can localize the effects of axioms with number restrictions, such as (7.12).

$$\geq 2\, R.B \sqsubseteq A \tag{7.12}$$

In order to ensure that only instances of B are counted, tableau algorithms need to include a *choose*-rule that, for each assertion $R(a, b)$, nondeterministically derives $B(b)$ or $\neg B(b)$. In the hypertableau setting, however, (7.12) is translated into the following DL-clause:

$$R(x, y_1) \wedge R(x, y_2) \wedge B(y_1) \wedge B(y_2) \rightarrow A(x) \vee y_1 \approx y_2 \tag{7.13}$$

No *choose*-rule is needed, as the DL-clause is simply applied to assertions of the form $R(a, b)$, $B(b)$, $R(a, c)$, and $B(c)$; furthermore, the conclusion is a tautology whenever $b = c$. The presence of "guard" atoms in the antecedent of (7.13) thus significantly reduces the nondeterminism introduced by such number restrictions. Furthermore, on Horn knowledge bases with number restrictions (which includes the common case of functional roles), our calculus exhibits no nondeterminism; in contrast, tableau calculi still need the *choose*-rule, which introduces nondeterminism even if all GCIs have been fully absorbed.

The hypertableau calculus as presented in this paper does not generalize negative absorption directly; for example, the negatively absorbed axiom (7.14) is translated into a DL-clause (7.15) which is then applied to all individuals in an ABox.

$$\neg A \sqsubseteq B \tag{7.14}$$

$$\rightarrow A(x) \vee B(x) \tag{7.15}$$

Negative absorption can, however, easily be applied in our setting: to negatively absorb an atomic concept A, we simply replace in the input ABox and the DL-clauses all occurrences of A with $\neg A'$ where A' is a fresh concept, and then move the literals

involving A' to the appropriate side of DL-clauses. In our example, (7.15) would be thus converted into (7.16), which can then be applied deterministically.

$$A'(x) \rightarrow B(x) \tag{7.16}$$

Note that this will transform a DL-clause $A(x) \rightarrow B(x)$ into $\rightarrow A'(x) \vee B(a)$; however, a similar situation arises in tableau calculi, where applying negative absorption to $\neg A \sqsubseteq B$ means that $A \sqsubseteq B$ cannot be absorbed.

To summarize, unlike various absorption techniques that are guided primarily by heuristics, the hypertableau calculus provides a framework that captures all variants of absorption we are aware of, guarantees deterministic behavior whenever the input knowledge base is Horn, eliminates the need for the nondeterministic *choose*-rule, and allows for a more powerful use of "guard" atoms to further localize any remaining nondeterminism. Furthermore, in Section 5.1.3 we show that the our calculus provides a proof-theoretic framework for DLs that can uniformly handle certain useful extensions of $\mathcal{SROIQ}$.

7.2 Relationship with Caching

Various *caching* optimizations can be used to reduce the sizes of the models constructed during knowledge base classification [Ding and Haarslev, 2006; Horrocks, 2007]. Most such optimizations operate by storing information about the roots of some trees of blockable individuals when those trees have been fully constructed; when a fresh individual is introduced, the cache is searched for a tree with a similar root, and if such a tree is found then derivation rules (in particular the $\geq$-rule) are not applied to the fresh individual. In the proposed approaches, caching is used in parallel with blocking—that is, caching alone does not guarantee termination of the calculus, and caching must be carefully integrated with blocking in order not to affect soundness and/or completeness. This integration is particularly problematic in the presence of inverse roles.

In contrast, anywhere blocking alone is sufficient to guarantee termination of the calculus, and like caching it can avoid the construction of identical subtrees in different parts of a model (see Section 3.3). Furthermore, in Section 6.1 we present an optimization of anywhere blocking that can be seen as a very simple but effective form of general caching. Thus, anywhere blocking achieves many of the effects of caching without much of the added complexity.

Donini and Massacci [2000] have used anywhere blocking with caching of unsatisfiable concepts to obtain a tableau algorithm for the DL $\mathcal{ALC}$ that runs in single exponential time. Goré and Nguyen [2007] have presented an algorithm for the DL $\mathcal{SHI}$ that also runs in exponential time and achieves termination solely by caching both satisfiable and unsatisfiable concepts. These algorithms, however, tend to perform very poorly in practice. Furthermore, it is unclear how to extend these algorithms to DLs that provide number restrictions, nominals, and inverse roles, such as $\mathcal{SROIQ}$.

7.3 Relationship with First-Order Calculi

The original hypertableau calculus for first-order logic was subsequently extended with equality and has been implemented in the KRHyper theorem prover [Baumgartner *et al.*, 2008]. The calculus can be used for finite model generation, and it decides function-free clause logic.

Hyperresolution with splitting has been used to decide several description and modal logics [Georgieva *et al.*, 2003; Hustadt and Schmidt, 1999]. These approaches, however, rely on skolemization, which, as we have discussed previously, can be inefficient in practice. Furthermore, these approaches deal with logics that are much weaker than $\mathcal{SROIQ}$; in particular, we are not aware of a hyperresolution-based decision procedure that can handle inverse roles, number restrictions, and nominals.

Our hypertableau calculus is related to the Extended Positive (EP) tableau calculus for first-order logic by Bry and Torge [1998]. Instead of relying on skolemization,

EP satisfies existential quantifiers by introducing new constants, and this is done in a way that makes the calculus complete for finite satisfiability. EP is, however, unlikely to be practical due to a high degree of nondeterminism. Furthermore, EP does not provide a decision procedure for DLs such as $\mathcal{SROIQ}$ that do not enjoy the finite model property [Baader and Nutt, 2007]. Consider, for example, the knowledge base whose TBox contains axioms (7.17) and (7.18), and whose ABox contains assertion (7.19):

$$A \sqsubseteq \exists R.A \tag{7.17}$$

$$\top \sqsubseteq \leq 1\, R^{-}.\top \tag{7.18}$$

$$(\neg A \sqcap \exists R.A)(a) \tag{7.19}$$

EP will try to satisfy the existential quantifier on a by "reusing" a—that is, by adding assertions $R(a, a)$ and $A(a)$. This leads to a contradiction, so EP will backtrack, introduce a fresh individual b, and add assertions $R(a, b)$ and $A(b)$; to satisfy (7.17), it will then also add $\exists R.A(b)$. To satisfy the existential quantifier in the latter assertion, EP will again try to "reuse" a; this will fail, so it will try to "reuse" b by adding an assertion $R(b, b)$. Due to (7.18), however, b will be merged into a, which results in a contradiction; therefore, EP will backtrack, introduce yet another fresh individual c and add the assertions $R(b, c)$, $A(c)$, and $\exists R.A(c)$. By repeating the argument, it is easy to see that EP will generate ever larger models and will not terminate. This is unsurprising since the knowledge base is satisfied only in infinite models. To achieve termination on such knowledge bases, EP would need to be extended with blocking techniques such as the ones described in this paper.

Baumgartner and Schmidt [2006] developed a so-called *blocking* transformation of first-order clauses, which can improve the performance of bottom-up model generation methods. Roughly speaking, the clauses are modified in a way that makes a bottom-up calculus derive $s \approx t$ or $s \not\approx t$ for each term s that is a subterm of t;

then, an application of paramodulation to $s \approx t$ achieves an effect that is analogous to "reusing" s instead of t in the EP tableau calculus. This transformation, however, does not ensure termination for DLs that do not have the finite model property. For example, for the same reasons as explained in the previous paragraph, hyperresolution with splitting does not terminate on the clauses obtained by an application of the blocking transformation to (the clauses corresponding to) (7.17)–(7.19). Furthermore, even for DLs that enjoy the finite model property, an "unlucky" sequence of applications of derivation rules can prevent a bottom-up model generation method with blocking from terminating (please refer to Section 3.4 for more details).

Part III

Classification and Retrieval

Chapter 8

Overview

One of the core services provided by DL reasoners is *classification*, the discovery of all subsumptions between concept names occurring in a knowledge base. In fact, classification is the primary (and often the only) reasoning service exposed by ontology engineering tools [Lutz *et al.*, 2006]. The Protégé-OWL editor,[1] for example, includes a "Reasoning" button which performs classification. The resulting hierarchy of subsumptions is used to organize concept names within all aspects of Protégé's interface, and the subsumptions which arise as implicit consequences of a knowledge base are the primary mechanism authors use to check that the axioms they write are consistent with their intuitions about the structure of the domain. Finally, other reasoning services, such as explanation and query answering, typically exploit a cached version of the classification results; classification is thus usually the first task performed by a reasoner.

For some less expressive DLs, such as members of the $\mathcal{EL}$ family, it may be possible to derive all subsumptions in a single computation [Baader *et al.*, 2005]. In general, however, it is necessary to "deduce" the complete set of subsumptions by performing a number of individual *subsumption tests* between pairs of concepts; each such test can be reduced to a single satisfiability/model-generation problem, which can be solved using the techniques described in Part II.

[1] http://protege.stanford.edu/overview/protege-owl.html

In Chapter 9, we present a novel algorithm which can greatly reduce the number of subsumption tests needed to classify a set of concepts. Our algorithm is also able to exploit *partial information* about the subsumption relation—for example, the set of subsumptions which are explicitly stated in the knowledge base—to further reduce the number of tests. Chapter 10 describes techniques for extracting such partial information from data generated in the course of model generation. Together, these techniques generalize and extend a wide range of commonly-implemented classification optimizations.

8.1 Difficulties

For a set of n concepts, there are a total of n^2 possible subsumptions, thus a naïve representation of classification results can require large amounts of storage. Most systems exploit the transitivity of subsumption and store only the "most specific" subsumers and "most general" subsumees of each classified concept; we call such a compressed representation a *taxonomy*. A taxonomy lends itself to representation as a graph or tree, often called a *subsumption hierarchy*.

It is clear that a set of n concepts can be classified by simply performing all n^2 individual subsumption tests, but for the tree-shaped subsumption hierarchies typically found in realistic knowledge bases much better results can be achieved using algorithms that construct the taxonomy incrementally. Concepts are inserted into the taxonomy one by one, and the correct location for each is found by traversing the partially-constructed hierarchy, performing subsumption tests as each node of the graph is visited.

This kind of algorithm suffers from two main difficulties. First, individual subsumption tests can be computationally expensive—for some complex knowledge bases, even state-of-the-art reasoners may take a long time to perform a single test. Second, even when subsumption tests themselves are cheap, a knowledge base containing a

very large number of concept names will obviously result in a very large taxonomy, and repeatedly traversing this structure can be costly. This latter problem is particularly acute for the relatively flat (i.e., broad and shallow) tree-shaped hierarchies often found in large biomedical knowledge bases. In general, subsumption tests must be performed between every pair of concept names with the same direct subsumer(s), thus broad hierarchies require many such tests. These two difficulties clearly interact: large numbers of concept names require large numbers of subsumption tests, each of which can be expensive.

The first difficulty is usually addressed by using an optimized construction that tries to minimize the number of subsumption tests performed in order to construct the taxonomy. Most implemented systems use an "enhanced traversal" algorithm due to Ellis [1991] and to Baader *et al.* [1994] which adds concept names to the taxonomy one at a time using a two-phase strategy. In the first phase, the most specific subsumers of a concept C are found using a top-down breadth-first search of the partially-constructed taxonomy. In this phase, subtrees of non-subsumers of C are not traversed, which significantly reduces the number of tests performed. The second phase finds the most general subsumees of C using a bottom-up search in a similar way. The algorithm exploits the structure of the knowledge base to identify "obvious" subsumers (so-called *told-subsumers*) of each concept name, and uses this information in a heuristic that chooses the order in which concepts are added, the goal being to construct the taxonomy top-down; it also exploits information from the top-down search in order to prune the bottom-up search.[2]

The second difficulty can be addressed by optimizations that try to identify a subset of the concept names for which complete information about the subsumption relation can be deduced without performing any individual subsumption tests. This can be achieved, e.g., by identifying *completely-defined* concept names [Tsarkov *et*

[2]Other optimizations can be used to decrease the cost of individual subsumption tests (see, e.g., [Tsarkov *et al.*, 2007]), but these techniques are largely orthogonal to classification optimizations.

al., 2007]—those between which only structurally-obvious subsumptions hold, or by applying structural subsumption algorithms to inexpressive fragments of the knowledge base [Möller *et al.*, 2008]. Having constructed part of the taxonomy using such techniques, the remaining concept names can be added using the standard enhanced-traversal algorithm.

8.2 Algorithm Summary

Chapters 9 and 10 together present a new classification algorithm that generalizes and refines the above techniques. Our approach is based on maintaining, and incrementally extending, two sorts of information: a set of pairs of concepts A and B such that we know that A is subsumed by B (the *known subsumptions*), and a set of pairs of concepts such that we know that A is not subsumed by B (the *known non-subsumptions*). The key insight is that information from these two sets can often be combined to derive new information. For example, if we know that A is subsumed by B, and that A is *not* subsumed by C, then we can conclude that B is not subsumed by C. In Section 9.2 we show how to derive the largest possible set of such inferences.

Our classification algorithm, described in Chapter 9, is straightforward: at each stage we pick a pair of concepts $\langle A, B \rangle$ which does not appear in either the set of known subsumptions or the set of known non-subsumptions, perform a subsumption test between the two, and use the result of the test to further extend the sets of known subsumptions and non-subsumptions. Each such test adds at least one pair to one of the sets (i.e. $\langle A, B \rangle$ itself). Eventually, every possible pair of concepts is present in one set or the other, and the set of known subsumptions is equal to the subsumption relation.

An important advantage of our algorithm is that the initial sets of known (non-) subsumptions may be empty, may include partial information about all concepts, and/or may include complete information about some concepts; in all cases the in-

formation is maximally exploited. Such information can be derived from a variety of sources, ranging from syntactic analysis of the knowledge base to existing classification results for a subset of concept names from the knowledge base (e.g. independent modules or classified sub- or super-sets of the KB) to data derived in the course of reasoning.

In Chapter 10 we show how the models constructed by (hyper)tableau-based reasoners in the course of subsumption and satisfiability testing can be used as sources of partial information about the subsumption relation. For example, if a reasoner produces a model containing an individual which is a member of both the concept A and the complement (negation) of the concept B, then A is clearly not subsumed by B; more sophisticated analysis based on the *dependency tracking* structures typically maintained by tableau reasoners also allows detection of subsumptions. The models generated by tableau reasoners are typically very rich sources for this type of information; in fact, for knowledge bases which do not result in nondeterminism, including all Horn-$\mathcal{SHIQ}$ (and thus all $\mathcal{EL}$) knowledge bases, the model constructed by a hypertableau-based reasoner when checking the satisfiability of a concept A will contain sufficient information to determine if A is (not) subsumed by B for all concept names B occurring in the knowledge base.

Our approach to partial-information derivation provides an efficient generalization of the told-subsumer and completely-defined optimizations, both of which derive partial information from structural analysis of the knowledge base. When the known (non-)subsumption information is incomplete, our algorithm incrementally computes additional (non-)subsumption relationships, and maximally exploits the resulting information to refine the sets of known and possible subsumers; this can be seen as a generalization of the search-pruning optimizations introduced by Baader *et al.* [1994].

Chapter 9

Deducing a Quasi-Ordering

In this chapter we present a general-purpose algorithm for deducing a quasi-ordering by testing whether pairs of elements are members of that ordering. Section 9.2 describes how partial information about the ordering can be maximally exploited to avoid redundant tests, and Section 9.3 presents both pseudocode for propagating information from a single test based on this technique, as well as two different algorithms for combining tests to deduce a quasi-ordering: one general-purpose algorithm, and one algorithm based on heuristics derived from properties common to quasi-orderings representing subsumption relations. Section 9.4 provides an example of applying the latter of these algorithms.

9.1 Preliminaries

We first introduce some notation and definitions that will be useful in what follows.

Given a set of elements $U = \{a, b, c, ...\}$, let R be a binary relation over U, i.e., a subset of $U \times U$. We say that there is a *path* from a to b in R if there exist elements $c_0, ..., c_n \in U$ such that $n > 0$, $c_0 = a$, $c_n = b$, and $\langle c_i, c_{i+1} \rangle \in R$ for all $0 \leq i < n$. The *transitive closure* of R is the relation R^+ such that $\langle a, b \rangle \in R^+$ iff there is a path from a to b in R. The *transitive-reflexive closure* R^* of R is the transitive closure of the reflexive extension of R, i.e. $R^+ \cup \{\langle a, a \rangle \mid a \in U\}$.

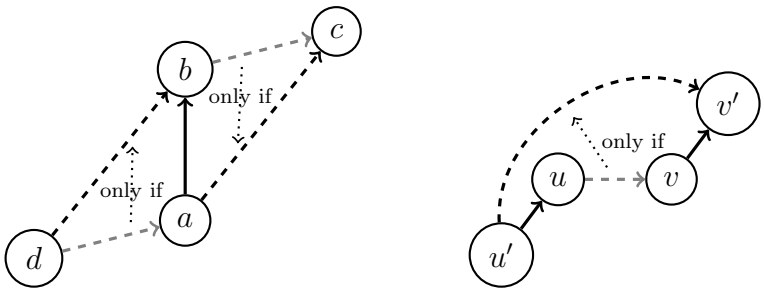
A binary relation is a *quasi-ordering* if it is both reflexive and transitive. Clearly, the subsumption relation on a set of concepts is a quasi-ordering. Note, however, that it is not a *partial-ordering*, because it is not antisymmetric: $A \sqsubseteq B$ and $B \sqsubseteq A$ does not imply that $A = B$ (i.e. semantically equivalent concepts may be syntactically distinct).

The *restriction* of a relation R to a subset D of U is the relation $R[D] = R \cap (D \times D)$. All restrictions of a reflexive relation are reflexive, and all restrictions of a transitive relation are transitive; thus, a restriction of a quasi-ordering is itself a quasi-ordering. Further, if $R \subseteq S$ for relations R and S, then $R[D] \subseteq S[D]$ for all $D \subseteq U$.

Given a universe U, a quasi-ordering R over U, and a finite set of elements $D \subseteq U$, we consider the problem of computing the restriction $R[D]$ via tests of the form $\langle a, b\rangle \in^? R$. If U is the set of (arbitrary) concept expressions which can be represented in logic $\mathcal{L}$, R is the subsumption relation over U, and D is the set of concept names occurring in a knowledge base $\mathcal{K}$ written in language $\mathcal{L}$, then computing $R[D]$ is equivalent to classifying $\mathcal{K}$, and the relevant tests are subsumption tests.

We assume that we begin with partial information about R: we are provided with a set $K = \{\langle a_0, b_0\rangle, ..., \langle a_m, b_m\rangle\}$ where $\langle a_i, b_i\rangle \in R$ for $0 \leq i \leq m$, and also with a set $K_{neg} = \{\langle c_0, d_0\rangle, ..., \langle c_n, d_n\rangle\}$ where $\langle c_i, d_i\rangle \notin R$ for $0 \leq i \leq n$. We call the set K the *known* portion of R. For ease of presentation, we do not operate on the set K_{neg} directly; our presentation instead refers to its complement $U \times U \setminus K_{neg}$, which we denote by P and call the *possible* portion of R. It is thus the case that $K \subseteq R \subseteq P$. If no partial information is available, then $K = \emptyset$ and $P = U \times U$.

We can use the result of each test $\langle a, b\rangle \in^? R$ to further refine the bounds on R by either adding $\langle a, b\rangle$ to K or removing it from P; eventually $K[D] = R[D] = P[D]$. We next show, however, that the bounds on R can sometimes be refined without performing additional tests by combining information from K and P.



(a) Simple cases (b) General case

Figure 9.1: Eliminating possible edges: if the solid edges are known to be in quasi-ordering R, then the gray edges can be in R only if the indicated dashed edges are in R.

9.2 Maximizing Partial Information

The key to minimizing the number of explicit tests required to discover $R[D]$ is maximizing the information gained from K and P. To do so, we exploit the knowledge that R is a quasi-ordering. In this case, $K \subseteq R$ obviously implies that $K^* \subseteq R$, so we can use K^* to obtain a tighter lower bound on R. Less obvious is the fact that we can also obtain a tighter upper bound on R by identifying tuples in P which are not consistent with K and the transitivity of R.

For example, consider the case shown in Figure 9.1a. If we know that b is a successor of a in R (i.e., $\langle a, b\rangle \in K$), then an element c can be a successor of b only if it is also a successor of a (if $\langle a, c\rangle \notin P$ then $\langle b, c\rangle \notin R$). Further, a can be a successor of an element d only if b is also a successor of d.

Both of these examples are special cases of the structure shown in Figure 9.1b: if u is a successor of u' and v' is a successor of v, then an edge from u to v would form a path all the way from u' to v', requiring v' to be a successor of u'. Since R is reflexive we can choose $u' = u$ or $v = v'$ to see that v can be a successor of u only if v is a successor of u' and v' is also a successor of u. We use this to formalize a subset $\lfloor P \rfloor_K$ of P, and show that $\lfloor P \rfloor_K$ is the tightest possible upper bound on R.

Definition 20 Let K and P denote two relations such that $K^* \subseteq P$. We define the

reduction $\lfloor P \rfloor_K$ *of* P *due to* K as follows:

$$\lfloor P \rfloor_K = P \cap \{\langle u, v\rangle \mid \forall u', v' : \{\langle u', u\rangle, \langle v, v'\rangle\} \subseteq K^* \rightarrow \langle u', v'\rangle \in P\} \qquad \triangle$$

We now show that $\lfloor P \rfloor_K$ is the tightest possible upper bound on R.

Lemma 16 *Let* K *and* P *denote two relations such that* $K^* \subseteq P$*. (i) For all quasi-orders* R *such that* $K \subseteq R \subseteq P$*, it is the case that* $R \subseteq \lfloor P \rfloor_K$*. (ii) Let* S *be a proper subrelation of* $\lfloor P \rfloor_K$*. Then there exists a quasi-ordering* R *such that* $K \subseteq R \subseteq P$ *and* $R \not\subseteq S$*; i.e.* $\lfloor P \rfloor_K$ *is minimal.*

Proof *(i)* Let $\langle u, v\rangle$ be a tuple in R. For every u', v' such that $\{\langle u', u\rangle, \langle v, v'\rangle\} \subseteq K^*$, $K^* \subseteq R$ implies that $\{\langle u', u\rangle, \langle v, v'\rangle\} \subseteq R$. Because R is transitive and $\langle u, v\rangle \in R$, it must also be the case that $\langle u', v'\rangle \in R$ and thus that $\langle u', v'\rangle \in P$. Consequently, $\langle u, v\rangle \in \lfloor P \rfloor_K$, so $R \subseteq \lfloor P \rfloor_K$.

(ii) Choose elements a and b such that $\langle a, b\rangle \in \lfloor P \rfloor_K$ but $\langle a, b\rangle \notin S$. Let R be the transitive-reflexive closure of the relation $K \cup \{\langle a, b\rangle\}$. Clearly $K \subseteq R$ and $R \not\subseteq S$. Let $\langle u, v\rangle$ be any tuple in R. There are three cases:

1. $\langle u, v\rangle = \langle a, b\rangle$. Then $\langle u, v\rangle \in P$ since $\langle a, b\rangle \in \lfloor P \rfloor_K$ and $\lfloor P \rfloor_K \subseteq P$.

2. $\langle u, v\rangle \in K^+$. Then $\langle u, v\rangle \in P$ since $K^* \subseteq P$.

3. $\langle u, a\rangle \in K^*$ and $\langle b, v\rangle \in K^+$. Then $\langle u, v\rangle \in P$ since $\langle a, b\rangle \in \lfloor P \rfloor_K$.

For any tuple $\langle u, v\rangle \in R$, it is the case that $\langle u, v\rangle \in P$, thus $K \subseteq R \subseteq P$ and $R \not\subseteq S$. $\square$

Note that $\lfloor P \rfloor_K$ itself is not necessarily transitive: given three elements a, b, and c and the relation $P = \{\langle a, a\rangle, \langle b, b\rangle, \langle c, c\rangle, \langle a, b\rangle, \langle b, c\rangle\}$, it is the case that $\lfloor P \rfloor_\emptyset = P$. Of course no transitive subrelation R of P contains both $\langle a, b\rangle$ and $\langle b, c\rangle$.

9.3 Taxonomy Construction and Searching

As described in Section 9.2, given relations K and P such that $K \subseteq R \subseteq P$ for some unknown quasi-ordering R, a tuple $\langle a, b\rangle$ is an element of R if $\langle a, b\rangle \in K^*$, and $\langle a, b\rangle$ is not an element of R if $\langle a, b\rangle \notin \lfloor P \rfloor_K$; the only "unknown" elements of R are the tuples in $\lfloor P \rfloor_K \setminus K^*$. Further, if $\langle a, b\rangle \in \lfloor P \rfloor_K \setminus K^*$, then a test of the form $\langle a, b\rangle \in^? R$ provides additional information which can be used to extend K or restrict P. This suggests the following simple procedure for deducing the restriction $R[D]$ of a quasi-ordering R to domain D:

COMPUTE-ORDERING(K, P, D)

```
1  while K*[D] ≠ ⌊P⌋_K[D]
2        do choose some a, b ∈ D such that ⟨a, b⟩ ∈ ⌊P⌋_K \ K*
3           if ⟨a, b⟩ ∈? R then add ⟨a, b⟩ to K
4                           else remove ⟨a, b⟩ from P
5  return K[D]
```

Completely recomputing K^* and $\lfloor P \rfloor_K$ in each iteration of the above loop is clearly inefficient. Practical implementations would instead maintain both relations and update them as P and K change. Techniques for updating the transitive-reflexive closure of a relation are well-known [Cormen *et al.*, 2001; La Poutré and van Leeuwen, 1988]; we provide below a naïve algorithm that, given $\lfloor P \rfloor_K$, $K' \supseteq K$ and $P' \subseteq P$, computes an updated relation $\lfloor P' \rfloor_{K'}$.

The algorithm exploits the technique for eliminating edges that was described in Section 9.2 and Figure 9.1b: it removes a tuple $\langle u, v\rangle$ from the set of possible tuples $\lfloor P \rfloor_K$ when adding it to the set of known tuples K' would imply, due to the transitivity of R, that some other tuple would be at the same time both known (i.e., in K') and not possible (i.e., not in $\lfloor P' \rfloor_{K'}$). This is done incrementally by considering tuples that have either become known (i.e., are in $K' \setminus K$) or been shown to be impossible (i.e., are in $\lfloor P \rfloor_K \setminus P'$).

First, $\lfloor P \rfloor_K$ is copied to $\lfloor P' \rfloor_{K'}$. Then, in lines 2–4, each tuple $\langle u', v' \rangle$ that has been shown to be impossible is considered and, if there are tuples $\langle u', u \rangle$ and $\langle v, v' \rangle$ in K', then $\langle u, v \rangle$ is clearly not possible either (it would imply that $\langle u', v' \rangle$ is not only possible but known) and so is removed from $\lfloor P \rfloor_K$. Next, in lines 5–13, each tuple $\langle x, y \rangle$ that has become known is considered. There are two possible cases: one where x, y correspond to u', u in Figure 9.1b, and one where they correspond to v, v'. Lines 6–9 deal with the first case: if there are tuples $\langle u, v \rangle$ in $\lfloor P \rfloor_K$ and $\langle v, v' \rangle$ in K', but $\langle u', v' \rangle$ is not in P', then $\langle u, v \rangle$ is clearly not possible (it would imply that $\langle u', v' \rangle$ is not only possible but known) and so is removed from $\lfloor P \rfloor_K$. Lines 10–13 deal similarly with the second case: if there are tuples $\langle u, v \rangle$ in $\lfloor P \rfloor_K$ and $\langle u', u \rangle$ in K', but $\langle u', v' \rangle$ is not in P', then $\langle u, v \rangle$ is removed from $\lfloor P \rfloor_K$.

PRUNE-POSSIBLES($\lfloor P \rfloor_K, K, P', K'$)

```
1  ⌊P'⌋_K' ← ⌊P⌋_K
2  for each ⟨u', v'⟩ ∈ ⌊P⌋_K \ P'
3        do for each u, v such that ⟨u', u⟩ ∈ K' and ⟨v, v'⟩ ∈ K'
4              do remove ⟨u, v⟩ from ⌊P'⌋_K'
5  for each ⟨x, y⟩ ∈ K' \ K
6        do let u' ← x and u ← y
7           for each v such that ⟨u, v⟩ ∈ ⌊P⌋_K
8                 do if there exists v' such that ⟨v, v'⟩ ∈ K' and ⟨u', v'⟩ ∉ P'
9                       then remove ⟨u, v⟩ from ⌊P'⌋_K'
10          let v ← x and v' ← y
11                do for each u such that ⟨u, v⟩ ∈ ⌊P⌋_K
12                      do if there exists u' such that ⟨u', u⟩ ∈ K' and ⟨u', v'⟩ ∉ P'
13                            then remove ⟨u, v⟩ from ⌊P'⌋_K'
14          return ⌊P'⌋_K'
```

In the case where no information about the quasi-ordering $R[D]$ is available other than K and P, the COMPUTE-ORDERING procedure performs well. In many cases, however, some general properties of $R[D]$ can be assumed. In the case where R represents the subsumption relation between concept expressions, for example, $R[D]$ is typically much smaller than $D \times D$ (i.e., subsumptions occur between relatively few

pairs of concept names). In such cases, it is beneficial to use heuristics that exploit the (assumed) properties of $R[D]$ when choosing a and b in line 2 of the above procedure.

We summarize below a variant of COMPUTE-ORDERING which performs well when the restriction to be computed is treelike in structure and little information about the ordering is available in advance. This procedure is designed to perform individual tests in an order similar to the enhanced traversal algorithm; however, it minimizes the number of individual tests performed by maximally exploiting partial information.

The algorithm chooses an element of $a \in D$ for which complete information about $R[D]$ is not yet known. It identifies the subset $V^{\uparrow} \subseteq D$ of elements b for which $\langle a, b\rangle \in R$, and the subset $V^{\downarrow} \subseteq D$ of elements b for which $\langle b, a\rangle \in R$, updating K and P accordingly. In order to compute these sets efficiently, we make use of the subroutines SUCCESSORS and PREDECESSORS, which perform the actual tests. The SUCCESSORS and PREDECESSORS functions are derived from the enhanced traversal algorithm: they perform a breadth-first search of the *transitive reduction* $K^{\diamond}$ of the known subsumptions K—the smallest relation whose transitive closure is K^*. In order to avoid the cost of repeated traversals of $K^{\diamond}$, we restrict the searches to, respectively, the possible successors and predecessors of a. We omit the details of these search routines for the sake of brevity.

COMPUTE-ORDERING-2(K, P, D)

```
1  while K*[D] ≠ ⌊P⌋_K[D]
2      do choose some a, x ∈ D s.t. ⟨a, x⟩ ∈ ⌊P⌋_K \ K* or ⟨x, a⟩ ∈ ⌊P⌋_K \ K*
3         let B be the possible successors of a, i.e. D ∩ {b | ⟨a, b⟩ ∈ ⌊P⌋_K \ K*}
4         if B ≠ ∅ then V↑ ← SUCCESSORS(a, K◇[B])
5                       add ⟨a, b⟩ to K for every element b of V↑
6                       remove ⟨a, b⟩ from P for every element b of B \ V↑
7         let B be the possible predecessors of a, i.e. D ∩ {b | ⟨b, a⟩ ∈ ⌊P⌋_K \ K*}
8         if B ≠ ∅ then V↓ ← PREDECESSORS(a, K◇[B])
9                       add ⟨b, a⟩ to K for every element b of V↓
10                      remove ⟨b, a⟩ from P for every element b of B \ V↓
11 return K[D]
```

9.4 Example

Consider the process of using COMPUTE-ORDERING-2 to discover the subsumption relation $\{\langle a,a\rangle, \langle b,a\rangle, \langle b,b\rangle, \langle c,a\rangle, \langle c,c\rangle, \langle d,a\rangle, \langle d,b\rangle, \langle d,d\rangle\}$ with no initial partial information available. We initialize K to $\{\langle a,a\rangle\langle b,b\rangle, \langle c,c\rangle, \langle d,d\rangle\}$ and P to $\{a,b,c,d\} \times \{a,b,c,d\}$; this situation is shown in the left diagram of Figure 9.2. Each element appears in a tuple occurring in $K^* \setminus \lfloor P \rfloor_K$, so on the first execution of line 2 of COMPUTE-ORDERING-2 we are free to choose any element; assume that we choose d. Then SUCCESSORS performs the tests $d \sqsubseteq^? a$, $d \sqsubseteq^? b$, and $d \sqsubseteq^? c$ (discovering that a and b are the only successors of d), we add $\langle a,d\rangle$ and $\langle b,d\rangle$ to K, and we remove $\langle c,d\rangle$ from P. PREDECESSORS performs the tests $a \sqsubseteq^? d$, $b \sqsubseteq^? d$, and $c \sqsubseteq^? d$ (discovering that d has no predecessors), and we remove each of $\langle a,d\rangle$, $\langle b,d\rangle$, and $\langle c,d\rangle$ from P. Further, because $d \sqsubseteq a$ but $d \not\sqsubseteq c$, we can conclude that $a \not\sqsubseteq c$; similar reasoning shows that $b \not\sqsubseteq d$; $\lfloor P \rfloor_K$ is thus restricted to the set $\{\langle a,a\rangle, \langle a,b\rangle, \langle b,a\rangle, \langle b,b\rangle, \langle c,a\rangle, \langle c,b\rangle, \langle c,c\rangle, \langle d,a\rangle, \langle d,b\rangle, \langle d,d\rangle\}$. The states of K^* and $\lfloor P \rfloor_K$ at this point are shown in the left-center diagram of Figure 9.2.

In the next iteration through the COMPUTE-ORDERING-2 loop we cannot choose d since it does not occur in any tuple of $K^* \setminus \lfloor P \rfloor_K$; assume that we instead choose b. The only possible successor of b is a, so SUCCESSORS searches this one-element subgraph by performing the single test $b \sqsubseteq^? a$ (which returns TRUE), and we add $\langle b,a\rangle$ to K. The element d is already known to be a predecessor of b, so PREDECESSORS searches the subgraph $K^\diamond[\{a,c\}]$ by performing the tests $a \sqsubseteq^? b$ and $c \sqsubseteq^? b$, finding no predecessors, and the two corresponding tuples are removed from P. The states of K^* and $\lfloor P \rfloor_K$ are shown in the right-center diagram of Figure 9.2.

After two iterations, the set $K^* \setminus \lfloor P \rfloor_K$ contains only the pair $\langle c,a\rangle$; if we choose c in the next iteration then SUCCESSORS tests $c \sqsubseteq^? a$, we add $\langle c,a\rangle$ to K, and $K^* = \lfloor P \rfloor_K$. The final subsumption relation is given by K^* and is shown in the right-hand diagram of Figure 9.2.

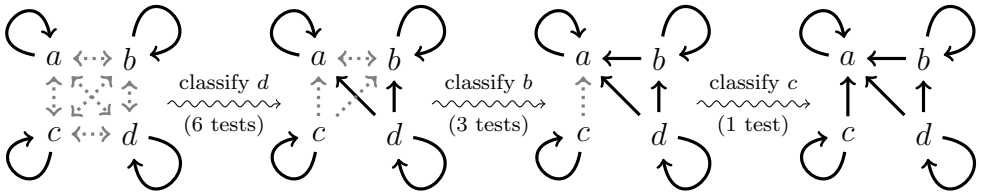


Figure 9.2: As classification proceeds, known edges (denoted by solid black arrows) are discovered, and possible edges (denoted by dotted gray arrows) are eliminated.

Compute-Ordering-2 thus classifies this knowledge base using 10 subsumption tests instead of the 16 required by a naïve brute-force approach. Note, however, that the results of the seven tests $a \sqsubseteq^? c$, $a \sqsubseteq^? d$, $b \sqsubseteq^? a$, $b \sqsubseteq^? d$, $c \sqsubseteq^? a$, $c \sqsubseteq^? b$, and $d \sqsubseteq^? b$ are sufficient to extend K and restrict P such that $K^* = \lfloor P \rfloor_K$, providing a full classification for this ontology. Identifying such a minimal set of tests is, however, extremely difficult without prior knowledge of the final taxonomy.

Chapter 10

Extracting Subsumption Information From Models

We next turn our attention to the specific case of identifying all subsumptions between the concept names occurring in a knowledge base $\mathcal{K}$. Instead of treating a reasoning service as an oracle that answers boolean queries of the form "is A subsumed by B w.r.t. $\mathcal{K}$?" (which we will write $\mathcal{K} \models^? A \sqsubseteq B$), we consider how information generated internally by common reasoning algorithms can be exploited to discover information about the subsumption quasi-ordering.

10.1 Identifying Non-Subsumptions

Most modern reasoners for Description Logics, including HermiT [Shearer *et al.*, 2008], Pellet [Parsia and Sirin, 2004], and FaCT++ [Tsarkov and Horrocks, 2006], transpose subsumption queries into consistency tests. In particular, to determine if $\mathcal{K} \models A \sqsubseteq \bot$, these reasoners test whether the knowledge base $\mathcal{K} \cup \{A(a)\}$ is consistent, where a is a fresh individual not appearing in $\mathcal{K}$.

All of the above reasoners perform such consistency tests by trying to construct (an abstraction of) a model of the knowledge base. We begin by summarizing the semantics of interpretations and models from Definition 1 and Definition 4.

Given sets of atomic concepts N_C, atomic roles N_R and individuals N_I, an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty set $\Delta^{\mathcal{I}}$ and an *interpretation function*

$\cdot^{\mathcal{I}}$ which maps every element of N_C to a subset of $\Delta^{\mathcal{I}}$, every element of N_R to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and every element of N_I to an element of $\Delta^{\mathcal{I}}$; the interpretation function is further extended to all concepts constructed from N_C, N_R, and N_I. An interpretation $\mathcal{I}$ is a model of an axiom $A \sqsubseteq B$ if $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ (similar definitions hold for other kinds of statement); it is a model of a knowledge base $\mathcal{K}$ if it models every statement in $\mathcal{K}$. We now introduce two further notions related to models.

Definition 21 Let A and B be concepts. A model $\mathcal{I}$ of $\mathcal{K}$ is a *witness for the satisfiability of A w.r.t. $\mathcal{K}$* if $A^{\mathcal{I}}$ is nonempty; it is a *witness for the non-subsumption $A \not\sqsubseteq B$ w.r.t. $\mathcal{K}$* if $A^{\mathcal{I}} \not\subseteq B^{\mathcal{I}}$, i.e., if there exists $i \in \Delta^{\mathcal{I}}$ s.t. $i \in A^{\mathcal{I}}$ and $i \notin B^{\mathcal{I}}$. △

One method for finding a model of a knowledge base, if such a model exists, is detailed in Part II of this thesis, however other model-construction methods exist, including the more traditional tableau algorithms used by Pellet and FaCT++. All of these procedures typically represent the model being constructed as an ABox, i.e., as a set of assertions of the form $C(x)$ and $R(x, y)$ for individuals x, y, concepts C, and roles R. For logics which can induce infinite models, reasoners implicitly depend upon an *unraveling* theory which describes the relationship between the ABox constructed and a model. The unraveling for the calculus given in Part II is detailed in Definition 15, but most unravelings provide similarly straightforward interpretation of the vast majority of assertions within an ABox: an ABox containing the assertion $C(x)$ represents a model in which $x^{\mathcal{I}} \in C^{\mathcal{I}}$. To construct a witness for the satisfiability of a concept A, tableau reasoners initialize the ABox with an assertion $A(x)$ and perform ABox construction in a goal-directed manner by adding further assertions only as necessary in order to ensure that the ABox represents a model of $\mathcal{K}$.

Assuming that the construction is successful, the resulting ABox/model provides a rich source of information. For example, for any concepts A and B such that $A(x)$ and $(\neg B)(x)$ are both in the ABox, it is the case that $x^{\mathcal{I}} \in A^{\mathcal{I}}$ and $x^{\mathcal{I}} \notin B^{\mathcal{I}}$;

thus the model is a witness for the non-subsumption $\mathcal{K} \models A \not\sqsubseteq B$. In many tableau settings, the non-presence of $B(x)$ in the ABox is sufficient to conclude the relevant non-subsumption; in fact, when using a hypertableau algorithm, this is *always* the case.[1] We formalize the detection of non-subsumptions in the hypertableau setting as follows.

Lemma 17 *Let $\mathcal{K}$ be a knowledge base, and (T, λ) a derivation for $\mathcal{K}$ as given by Definition 13. If there exists a clash-free ABox $\mathcal{A}$ such that $\lambda(t) = \mathcal{A}$ for leaf t of T, and $A(x) \in \mathcal{A}$ and $B(x) \notin \mathcal{A}$ for B an atomic concept, A any concept, and x an individual which is not indirectly blocked in $\mathcal{A}$, then $\mathcal{K} \not\models A \sqsubseteq B$.*

Proof Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be the interpretation given by applying the unravelling specified by Definition 15 to $\mathcal{A}$. Then $x^{\mathcal{I}} \in A^{\mathcal{I}}$ and $x^{\mathcal{I}} \notin B^{\mathcal{I}}$. By Lemma 11, $\mathcal{I}$ is a model of $\mathcal{K}$, so $\mathcal{K} \not\models A \sqsubseteq B$. □

The goal-directed nature of the ABox construction used by the hypertableau calculus (and other common tableau calculi) means that the ABoxes constructed are typically quite small. As a result, these ABoxes tend to be extremely rich in non-subsumption information: in a typical witness for the satisfiability of A, i.e., a model $\mathcal{I}$ of $\mathcal{K}$ with $i \in A^{\mathcal{I}}$, there will be relatively few other concept names B such that $i \in B^{\mathcal{I}}$, and thus $\mathcal{I}$ will identify the vast majority of concept names in $\mathcal{K}$ as non-subsumers of A. For this reason, it is almost always more efficient to record the set $P_A = \{B \mid i \in A^{\mathcal{I}} \text{ and } i \in B^{\mathcal{I}} \text{ for some } i\}$ of "possible subsumers" of A.

10.2 Identifying Subsumptions

While single models allow us to detect non-subsumptions, additional information about the space of possible models is required in order to identify subsumption re-

[1] Traditional tableau algorithms sometimes employ an optimization called *negative absorption* [Horrocks, 2007], which complicates the way in which conclusions can be drawn from the non-presence of assertions. In such cases, consulting not only the ABox but also the absorption data used during its construction is necessary to extract such information.

lationships. Sound and complete tableau reasoning algorithms systematically explore the space of all "canonical" models (typically tree- or forest-shaped models), on the basis that, if any model exists, then one of these canonical models also exists; our formalization in Part II characterizes this space as a *derivation*—a tree whose nodes are labeled by ABoxes—as given by Definition 13. In particular, when $\mathcal{K}$ includes disjunctions or other sources of nondeterminism, it may be necessary to choose between several possible ways of modeling such statements, and to backtrack and try other possible choices if the construction fails; i.e. the path first traversed through a derivation tree might lead to a clash, and other branches from that path must be explored.

For such algorithms, it is usually easy to show that, if the ABox was initialized with $A(x)$, the construction did not involve any nondeterministic choices (the derivation is linear), and the resulting ABox includes the assertion $B(x)$, then it is the case that in any model $\mathcal{I}$ of $\mathcal{K}$, $i \in A^{\mathcal{I}}$ implies $i \in B^{\mathcal{I}}$, i.e., that $\mathcal{K} \models A \sqsubseteq B$. Moreover, as we have already seen in Section 10.1, such an ABox is (at least in the hypertableau case) a witness to the non-subsumption $\mathcal{K} \models A \not\sqsubseteq C$ for all concept names C such that $C(x)$ is not in the ABox. Thus, when testing the satisfiability of a concept A, it may be possible to derive *complete information* about the subsumers of A. Again, we formalize the detection of subsumptions only for the case of the hypertableau calculus.

Lemma 18 *Let $\mathcal{K}$ be a knowledge base, and (T, λ) a derivation for $\mathcal{K}$ as given by Definition 13. If an individual a occurs in $\mathcal{K}$ only in a single assertion of the form $A(a)$, and for every leaf t of T it is the case that either $B(a) \in \lambda(t)$ or $\lambda(t)$ contains a clash, then $\mathcal{K} \models A \sqsubseteq B$.*

Proof Assume that $\mathcal{K} \not\models A \sqsubseteq B$. Then there exists some model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{K}$ such that $i \in A^{\mathcal{I}}$ and $i \notin B^{\mathcal{I}}$ for some $i \in \Delta^{\mathcal{I}}$. We construct the knowledge base

$\mathcal{K}' = \mathcal{K} \cup \{A(a'), \neg B(a')\}$ with a' a fresh individual not occurring in $\mathcal{K}$. By extending $\mathcal{I}$ such that $a'^{\mathcal{I}} = i$ we produce a model for $\mathcal{K}'$, so $\mathcal{K}'$ is consistent. Further, the knowledge base $\mathcal{K}'' = \mathcal{K}' \setminus \{A(a)\}$ is also consistent by monotonicity. The individual a no longer occurs in $\mathcal{K}''$, so we are free to replace a' with a in all assertions of $\mathcal{K}''$ to produce the consistent knowledge base $\mathcal{K}'''$.

Finally, we construct a new tree T' by extending T with a new child t' for every leaf t of T such that $\lambda(t)$ is clash-free, and a new labeling function λ' such that $\lambda'(t) = \lambda(t) \cup \{\neg B(a)\}$ for every node t in T, and $\lambda'(t) = \lambda(t') \cup \{\neg B(a), \bot\}$ for every fresh leaf t, where t' is the parent of t. It is easy to see that (T', λ') is a derivation for $\mathcal{K}'''$: since $\mathcal{K} \subset \mathcal{K}'''$, every rule applicable to (T, λ) and $\mathcal{K}$ is also applicable to (T', λ') and $\mathcal{K}'''$, and for each leaf t of T such that $\lambda(t)$ is clash-free, the *Hyp*-rule can be applied to the clause $B(x) \wedge \neg B(x) \rightarrow \bot$ with mapping σ such that $\sigma(x) = a$. Thus $\mathcal{K}'''$ is consistent, but there exists a derivation for $\mathcal{K}'''$ containing no clash-free branches; this contradicts Lemma 10. □

The hypertableau calculus is designed to reduce nondeterminism, and avoids it completely when dealing with Horn-$\mathcal{SHIQ}$ ontologies; for such ontologies it is thus able to derive complete information about the subsumers of a concept A using a single satisfiability test, which constructs an entire (linear) derivation tree. This allows a hypertableau reasoner to derive all relevant subsumption relationships in a Horn ontology as a side effect of performing satisfiability tests on each of the concept names.

This idea can be extended so as to also derive useful information from nondeterministic constructions by exploiting the dependency labeling typically used to enable "dependency-directed backtracking"—an optimization which reduces the effects of nondeterminism in reasoning [Horrocks, 1997]. In the resulting ABoxes, each assertion is labelled with the set of choices among derivation branches on which it depends. An empty label indicates that the relevant assertion is present in all branches of the derivation, regardless of any choices made during the construction process. Thus, if

the ABox is initialized with $A(x)$, an empty-labelled assertion $B(x)$ in the resulting ABox can be treated in the same way as if the construction had been completely deterministic. Performing a satisfiability test on A may, therefore, allow some subsumers of A to be identified even when nondeterministic choices are made during reasoning. In practice, almost all of the actual subsumers of A can usually be identified in this way.

It is easy to see that this idea is closely related to, and largely generalizes, the told subsumer and completely-defined optimizations described in Section 8.1. For a completely defined concept name A, a satisfiability test on A will be deterministic (and typically rather trivial), and so will provide complete information about the subsumers of A. Similarly, if B is a told subsumer of A, then an ABox initialized with $A(x)$ will always produce $B(x)$, and almost always deterministically. (It is theoretically possible that $B(x)$ will be added first due to some nondeterministic axiom in the ontology).

Chapter 11

Related Work

Computing a quasi- (or partial-) ordering for a set of n incomparable elements clearly requires n^2 individual tests in the worst case—naïvely comparing all pairs is thus "optimal" by the simplest standard. The literature therefore focuses on a slightly more sophisticated metric which considers both the number of elements in the ordering as well as the *width* of the ordering—the maximum size of a set of mutually incomparable elements. Faigle and Turán [1985] have shown that the number of comparisons needed to deduce an ordering of n elements with width w is at most $O(wn \log(n/w))$ and Daskalakis *et al.* provide an algorithm which approaches this bound by executing $O(n(w + \log n))$ comparisons [2007]. Taxonomies, however, tend to resemble trees in structure, and the width of a subsumption ordering of n elements is generally close to $n/2$. Further, the algorithms of Faigle and Turán as well as Daskalakis *et al.* rely on data structures which require $O(nw)$ storage space even in the best case, and thus exhibit quadratic performance when constructing a taxonomy.

A taxonomy-construction strategy which performs well for tree-like relations is described by Ellis [1991]: elements are inserted into the taxonomy one at a time by finding, for each element, its subsumers using a breadth-first search of all previously-inserted elements top-down, and then its subsumees using a breadth-first search bottom-up. Baader *et al.* further refine this technique to avoid redundant subsumption tests during each search phase: during the top search phrase, a test $\mathcal{K} \models^? A \sqsubseteq B$

is performed only if $\mathcal{K} \models A \sqsubseteq C$ for all subsumers C of B [1994]. This can be seen as a special case of our $\lfloor P \rfloor_K$ pruning of possible subsumers, with the restriction that it only applies to subsumption tests performed in a prescribed order.

The traversal algorithms described by Ellis and by Baader *et al.* perform subsumption tests between every pair of siblings in the final taxonomy. Such algorithms are thus very inefficient for taxonomies containing nodes with large numbers of children; completely flat taxonomies result in a quadratic number of subsumption tests. Haarslev and Möller propose a way to avoid the inefficiencies of these traversals by *clustering* a group of siblings $A_1, ..., A_n$ by adding the concept $\bigsqcap_{1 \leq i \leq n} A_i$ to the taxonomy [2001a]. Our approach can easily incorporate this technique by including partial or complete information about such new concepts in K and P. In practice, however, the partial information extracted from tableau models almost always includes non-subsumptions between siblings in such flat hierarchies, so these refinements are unnecessary.

Baader *et al.* also describe techniques for identifying subsumers without the need for multiple subsumption tests by analyzing the syntax of concept expressions in an knowledge base: if a KB contains an axiom of the form $A \sqsubseteq B \sqcap C$ where A and B are atomic concepts, then B is a "told subsumer" of A, as are all the told subsumers of B. The simplification and absorption techniques described by Horrocks [1997] increase the applicability of such analysis. Haarslev *et al.* further extend this analysis to detect non-subsumption: an axiom of the form $A \sqsubseteq \neg B \sqcap C$ implies that A and B are disjoint, thus neither atomic concept subsumes the other (unless both are unsatisfiable) [2001a]. Tsarkov *et al.* describe a technique for precisely determining the subsumption relationships between "completely defined concepts"—atomic concepts whose definitions contain only conjunctions of other completely defined concepts [2007]. These optimizations can be seen as special cases of (non-)subsumption information being derived from (possibly incomplete) calculi as described in Chapter 10.

Part IV

Evaluation

Chapter 12

Empirical Results

Based on the techniques described in this thesis, we have implemented a prototype DL reasoner called HermiT.[1] In order to estimate how well our approach performs in practice, we have compared HermiT with two state-of-the-art tableau reasoners on several practical problems. The objective of this evaluation was *not* to establish the superiority of HermiT, but rather to compare the behavior of various novel aspects of our approach with that of the traditional algorithms used in many existing systems, and to demonstrate the usefulness of our calculus on realistic problems. As such, we have structured our testing to try to isolate the performance impacts of different optimizations.

12.1 Reasoning Performance

In this section, we focus specifically on the performance of the hypertableau reasoning calculus described in Part II. It is important to understand that HermiT is a prototype, and as such does not always outperform the well-established reasoners. In particular, HermiT may be uncompetitive on ontologies where specialized optimizations are needed for good performance. For example, HermiT cannot process the SNOMED CT ontology due to the very large number of concepts, while many other reasoners can classify the ontology easily. These reasoners, however, employ

[1] http://hermit-reasoner.com/

techniques that are quite different from the standard tableau algorithm. For example, on an $\mathcal{EL}^{++}$ ontology such as SNOMED CT, Pellet uses the reasoning algorithm by Baader *et al.* [2005], and other reasoners employ specialized techniques as well [Haarslev *et al.*, 2008].

In an attempt to focus on core reasoning performance, we have not made use of the advanced classification techniques from Part III, applying only the technique for extracting partial information from models from Chapter 10 to cache results of subsumption tests in cases where reasoning is entirely deterministic. We hoped that this would approximate an optimization present in both of the other reasoners which avoids true subsumption tests by caching pseudo-models [Haarslev *et al.*, 2001a; Horrocks, 1997]; this is discussed further in Section 12.1.3. Empirical evaluation of the techniques from Part III is deferred to Section 12.2.

Similarly, artificial test problems such as those used in the TANCS comparison at the Tableaux'98 conference [Balsiger and Heuerding, 1998; Balsiger *et al.*, 2000] and the DL'98 workshop [Horrocks and Patel-Schneider, 1998b] are often either easy for reasoners employing particular optimizations or are only difficult due to the fact that they encode large propositional satisfiability problems [Horrocks and Patel-Schneider, 1998a]. Since our goal was to demonstrate the usefulness of the hypertableau calculus on realistic problems, we have chosen to ignore such ontologies and test problems, as they mainly test specialized calculi and optimizations that are applicable to various sublanguages of $\mathcal{SROIQ}$. Instead, we focus our evaluation on practical ontologies in which the main difficulty is due to nontrivial reasoning problems encountered during classification.

In addition to the hypertableau calculus described in Chapter 5, HermiT also implements the optimizations from Chapter 6 and the well-known dependency directed backtracking optimization [Horrocks, 2007]. Thus, HermiT fully supports $\mathcal{SROIQ}$ and it can perform both satisfiability and subsumption testing as well as knowledge

base classification. An extensive discussion of implementation techniques is beyond the scope of this thesis; we only comment briefly on the implementation of anywhere blocking.

12.1.1 Implementing Anywhere Blocking

When used in conjunction with subset blocking, described in Section 6.3, anywhere blocking can be more costly than ancestor blocking. In this case, determining the blocking status of an individual may require examination of all individuals in an ABox and not just the individual's ancestors; computing the blocking status of all individuals could thus result in a quadratic number of comparisons (with respect to the total number of individuals).

As previously mentioned, however, subset blocking is applicable only when reasoning over knowledge bases encoded in relatively inexpressive logics that do not include inverse roles, and the technique presents additional challenges in the hypertableau setting. When blocking is instead based on exact matching between labels, as is the case for the pairwise blocking strategy given by Definition 13 for $\mathcal{SROIQ}$, the single blocking strategy given by Definition 16 for $\mathcal{SHOQ}^+$, and the full single blocking strategy given by Definition 18 for $\mathcal{SHOI}$, the quadratic number of comparisons can be avoided by maintaining an associative array or hash table in which individuals are indexed by their four blocking labels.

In HermiT, this table is created by scanning all individuals in $\mathcal{A}$ in the increasing sequence of the ordering $\prec$. For each individual s in $\mathcal{A}$, if the parent of s is blocked, then s is indirectly blocked; otherwise, the algorithm queries the hash table for an individual whose blocking labels are equal to those of s. If the hash table contains such an individual t, then s is directly blocked in $\mathcal{A}$ by t; otherwise, s is not blocked in $\mathcal{A}$ so it is added into the hash table. The blocking status of all individuals in $\mathcal{A}$ can thus be determined with a linear number of hash table lookups (against with respect

to the number of individuals in $\mathcal{A}$).

12.1.2 Test Procedure

We used Pellet 2.0.0rc4 [Parsia and Sirin, 2004] and FaCT++ 1.2.2 [Tsarkov and Horrocks, 2006] as reference implementations of the $\mathcal{SHOIQ}$ tableau algorithm [Horrocks and Sattler, 2007]. Pellet employs ancestor blocking, while FaCT++ has recently been extended with anywhere blocking. At the time of testing, however, the implementation of anywhere blocking in FaCT++ was known to be incorrect,[2] so we switched this feature off and used FaCT++ with ancestor blocking as well. To measure the effects of ancestor vs. anywhere blocking, we also used HermiT-Anc—a version of HermiT with ancestor blocking.

Knowledge bases encoded as OWL ontologies can be split across multiple files that are joined together using *import* statements. Due to inconsistencies among reasoners with respect to handling of import statements, we run benchmarks over only ontologies contained within a single file. Because much of the test data available makes use of imports, we have "localized" ontologies by parsing them, resolving and parsing all imports, merging the main and imported ontologies together, and re-serializing the ontology, all using the OWL API (version 2.2.1, from the 104 Protege release). The version of the OWL API used generates files with invalid namespaces in a number of cases, and the resulting files sometimes still contain import statements in addition to all the axioms from the imported ontology; these errors were corrected by hand. Each test ontology in our test corpus can thus be parsed as a single file using the OWL API. All test ontologies are available online.[3]

We used a collection of 392 test ontologies that we assembled from three independent sources.

[2] Personal communication with Dmitry Tsarkov.

[3] http://www.comlab.ox.ac.uk/people/Rob.Shearer/2010/hermit-benchmarks.zip

- The Gardiner ontology suite [Gardiner *et al.*, 2006] is a collection of OWL ontologies gathered from the Web and includes many of the most commonly-used OWL ontologies. A number of files have names which are not valid URIs; in additional to the localization described above, we chose to rename these files.

- The Open Biological Ontologies (OBO) Foundry[4] is a collection of biology and life science ontologies. Imports are handled specially in these files: each main ontology contains only axioms, and comes with two associated wrappers, one containing import statements using relative path names to local files, and one containing full URIs to ontologies on the web. We produced single-file versions of all ontologies using the wrapper naming local imports. In cases where the original OBO ontology does not import any other files, the local imports file is empty (and does not even import the main file). In these cases we simply used the main ontology file.

- We also included a number of versions of the GALEN ontology [Rector and Rogers, 2006], a large and complex biomedical ontology which has been the subject of reasoner optimization for many years. We test on a number of versions with very different performance characteristics: the full version from the web (which none of the tested reasoners could classify), the "undoctored" version originally studied by Ian Horrocks during development of the FaCT system, a "doctored" version which was modified such that it was classifiable by FaCT, and "not-GALEN", a relatively easy-to-process recent variant.

For each reasoner and ontology, we parsed the ontology using the OWL API, loaded the ontology into the reasoner, classified the ontology, and wrote the classified taxonomy to disk. Only the times required for loading and classification were measured, and the two times were added to produce summary results. All tests were

[4] http://obofoundry.org/

performed on a 2.2 GHz MacBook Pro with 2 GB of physical memory. A classification attempt was aborted if it exhausted all available memory (Java tools were allowed to use 1 GB of heap space), or if it exceeded a timeout of 30 minutes.

The taxonomies generated by each reasoner occasionally contain differences, but almost all such situations are simply due to differing API conventions: FaCT++ sometimes makes classes direct parents of $\bot$ despite the existence of named subclasses, and HermiT inserts the names of unsupported datatypes into the hierarchy.

There were a few cases in which it appears that the version of Pellet tested misses some inferences; these cases were investigated manually, and it does appear that Pellet's results are incorrect while FaCT++ and HermiT produce the correct results. Investigation of such correctness issues is beyond the scope of this thesis.

12.1.3 Results

The three reasoners exhibited negligible differences in performance on most of the test ontologies. Therefore, we defer full results for all 392 ontologies to Appendix A and discuss here only the test results for "interesting" ontologies—that is, ontologies that can be classified by at least one of the tested reasoners, and that are either not trivial or on which the tested reasoners exhibited a significant difference in performance. These include several ontologies from the OBO corpus (Molecule Role, XP Uber Anatomy, XP Plant Anatomy, Cellular Component, Gazetteer, CHEBI), two versions of the National Cancer Institute (NCI) Thesaurus [Hartel *et al.*, 2005], two versions of the GALEN medical terminology ontology, two versions of the Foundational Model of Anatomy (FMA) [Golbreich *et al.*, 2006], the Wine ontology from the OWL Guide,[5] two SWEET ontologies developed at NASA,[6] and a version of the DOLCE ontology developed at the Institute of Cognitive Science and Technology of the Italian National Research Council.[7] Basic statistical information about these ontologies is summarized

[5] http://www.w3.org/TR/owl-guide/
[6] http://sweet.jpl.nasa.gov/ontology/
[7] http://www.loa-cnr.it/DOLCE.html

Table 12.1: Statistics of "Interesting" Ontologies

Ontology Name	Classes	Roles	Individuals	Number of Axioms TBox	RBox	ABox	Expressivity
Molecule Role	8849	2	128056	9243	1	128056	$\mathcal{ALE}+$
XP Uber Anatomy	11427	82	88955	14669	80	88955	$\mathcal{ALEHIF}+$
XP Plant Anatomy	19145	82	86099	35770	87	86099	$\mathcal{SHIF}$
XP Regulators	25520	4	155169	42896	3	155169	$\mathcal{SH}$
Cellular Component	27889	4	163244	47345	3	163244	$\mathcal{SH}$
NCI-1	27653	70	0	46800	140	0	$\mathcal{ALE}$
Gazetteer	150979	2	214804	167349	2	214804	$\mathcal{ALE}+$
GALEN-doctored	2748	413	0	3937	799	0	$\mathcal{ALEHIF}+$
GALEN-undoctored	2748	413	0	4179	800	0	$\mathcal{ALEHIF}+$
CHEBI	20977	9	243972	38375	2	243972	$\mathcal{ALE}+$
FMA-Lite	75141	2	46225	119558	3	46225	$\mathcal{ALEI}+$
SWEET Phenomena	1728	145	171	2419	239	491	$\mathcal{SHOIN}(\mathcal{D})$
SWEET Numerics	1506	177	113	2184	305	340	$\mathcal{SHOIN}(\mathcal{D})$
Wine	138	17	206	355	40	494	$\mathcal{SHOIN}(\mathcal{D})$
DOLCE-Plans	118	264	27	265	948	68	$\mathcal{SHOIN}(\mathcal{D})$
NCI-2	70576	189	0	100304	290	0	$\mathcal{ALCH}(\mathcal{D})$
FMA-Constitutional	41648	168	85	122695	395	86	$\mathcal{ALCOIF}(\mathcal{D})$

in Table 12.1.

We noticed that, for all three reasoners, classification times may vary from run to run. For Pellet and HermiT, this is due to Java's collection library: the order of iteration over collections often depends on the objects' hash codes, and these may vary from run to run; that, in turn, may change the order in which the derivation rules are applied, and some orders may be better than others. We conjecture that FaCT++ is susceptible to similar variations. While the times may vary, we have not noticed a case where an ontology might be successfully classified in one run, but not in another. Therefore, in Table 12.2 we present the classification times for the "interesting" ontologies that we obtained on one particular run; these times can be taken as being "typical." We identified four groups of ontologies, which we delineate in Table 12.1 and Table 12.2 by horizontal lines.

On the ontologies in the first group, HermiT performs similarly to HermiT-Anc, which suggests little impact of anywhere blocking on the performance. Consequently,

Table 12.2: Results of Performance Evaluation

Ontology Name	Classification Times (seconds)			
	HermiT	HermiT-Anc	Pellet	FaCT++
Molecule Role	3.3	3.4	25.7	304.5
XP Uber Anatomy	5.4	4.9	—	86.0
XP Plant Anatomy	12.8	11.2	87.2	22.9
XP Regulators	14.1	17.1	35.4	66.6
Celular Component	18.6	18.0	40.5	76.7
NCI-1	14.1	14.4	23.2	3.0
Gazetteer	131.9	132.3	—	—
GALEN-doctored	8.8	456.3	—	15.9
GALEN-undoctored	126.3	—	—	—
CHEBI	24.2	—	—	397.0
FMA-Lite	107.2	—	—	—
SWEET Phenomena	13.5	11.2	—	0.2
SWEET Numerics	76.7	72.6	3.7	0.2
Wine	343.7	524.6	19.5	162.1
DOLCE-Plans	1075.1	—	105.1	—
NCI-2	—	—	172.0	60.7
FMA-Constitutional	—	—	—	616.7

Note: entry — means that reasoner was unable to classify the ontology either due to time out or memory exhaustion.

we believe that HermiT outperforms the other reasoners mainly due to the reduced nondeterminism of the hypertableau calculus. As shown in Table 12.1, Molecule Role, XP Uber Anatomy, and NCI-1 do not use disjunctions, so HermiT classifies them entirely deterministically and caches all subsumption test results using a linear number of core consistency tests. FaCT++ outperforms HermiT on NCI-1 because FaCT++ classifies this ontology using the *completely defined concepts* optimization [Tsarkov and Horrocks, 2005a] instead of DL reasoning. This optimization enables FaCT++ to use simpler structural reasoning techniques on ontologies that satisfy certain syntactic constraints. Similar classification optimizations for HermiT based on the techniques from Part III are discussed in Section 12.2.

On the ontologies in the second group, HermiT-Anc is significantly slower than

HermiT. This suggests that anywhere blocking significantly improves performance since it prevents the construction of large models. Pellet runs out of memory on all ontologies in this group; furthermore, FaCT++ cannot process two of them and is significantly slower than HermiT on CHEBI. FaCT++, however, is faster than HermiT-Anc on CHEBI and GALEN-doctored, and we conjecture that this is mainly due to the ordering heuristics [Tsarkov and Horrocks, 2005b] used by FaCT++ in selecting which expansion rule to apply at each point in a derivation. The superior performance of HermiT on the ontologies in this group is mainly due to the fact that all of these ontologies can be classified entirely deterministically. Furthermore, HermiT's classification time is in most cases dominated by only the first subsumption test, as the caching of blocking labels described in Section 6.1 makes subsequent tests easy.

On the ontologies in the third group, HermiT is significantly slower than the other reasoners. As Table 12.1 shows, all ontologies in this group contain nominals, which prevents HermiT from caching blocking labels. Furthermore, due to nominals, the ABox must be taken into account during classification, and HermiT currently reapplies the hypertableau rules to the entire ABox in each run. Effectively, HermiT does not reuse any computation between different hypertableau runs. The other two reasoners, however, use the *completion graph caching* optimization [Sirin *et al.*, 2006], in which the tableau rules are first applied to the entire ABox, and the resulting completion graph is used as a starting point in each subsequent run. Such an optimization could also be incorporated into a hypertableau reasoner such as HermiT.

The version of HermiT tested was unable to classify the two ontologies in the fourth group. Both of these ontologies include disjunctions, which prevents application of the limited subsumption-caching optimization employed by this version of HermiT; for these ontologies HermiT performs classification using the algorithm by Baader *et al.* [1994] and not the techniques from Part III. All subsumption tests are

straightforward (each test takes less than 50 ms); however, the resulting taxonomy is rather shallow, so HermiT makes an almost quadratic number of tests. Both Pellet and FaCT++, however, use more optimized versions of the classification algorithm that reduce the number of tests that need to be performed. In the case of these two ontologies, performance appears to depend primarily upon classification, making comparisons of core reasoning performance difficult. Newer versions of HermiT implementing some of the techniques from Part III are able to classify both of these ontologies, in 137 and 947 seconds, respectively.

To summarize, although HermiT is not better than Pellet and FaCT++ on all ontologies, our results clearly demonstrate the practical potential of both reduced nondeterminism due to the hypertableau calculus and reduced model sizes due to anywhere blocking. In fact, anywhere blocking can mean the difference between success and failure on complex ontologies, which suggests that and-branching is a more significant source of inefficiency in practice than or-branching. Anywhere blocking is applicable to tableau calculi as well (as mentioned earlier, FaCT++ already contains a preliminary version of it), so we believe that our results can be used to improve the performance of tableau reasoners as well without the need for a major redesign. Conversely, most of the optimizations used in tableau reasoners can be used with the hypertableau algorithm, and incorporating them into HermiT would probably make HermiT competitive with Pellet and FaCT++ in those cases where HermiT is currently slower.

12.2 Classification Performance

In order to determine whether the techniques from Part III are likely to improve classification performance in practice we conducted two experiments using large ontologies derived from life-science applications.

12.2.1 Comparison with the Enhanced Traversal Method

First, we compared the performance of the COMPUTE-ORDERING-2 procedure described in Chapter 9 with the enhanced traversal algorithm of Baader *et al.* [1994]. In order to analyze how much improvement is due to the information extracted directly from models and how much is due to our new approach to taxonomy construction, we extend the enhanced traversal algorithm such that it first performs a satisfiability test on every class name and constructs a cache of information derived from the resulting models using the techniques described in Chapter 10. During the subsequent taxonomy construction, subsumption tests are performed only if the relevant subsumption relationship cannot be determined by consulting the cache. Note that this caching technique strictly subsumes the "told subsumer" and "primitive component" optimizations described by Baader *et al.* [1994].

We implemented both algorithms within the HermiT reasoner and performed testing using an OWL version of the well-known US National Cancer Institute thesaurus (NCI), a large but simple ontology containing 27,653 classes; this ontology is listed as NCI-1 in Table 12.1 and the discussion of reasoning tests in Section 12.1.3. The models constructed by HermiT during satisfiability testing of these classes provide complete information about the subsumption ordering for this ontology, so both algorithms are able to classify it without performing any additional tests. To study how the algorithms compare when less-than-complete information is available, we limited the amount of information extracted from HermiT's models. Both classification algorithms were provided with only a subset of the actual subsumption relation as "known" subsumptions, and a superset of the actual subsumption relation as "possible" subsumptions; the sizes of these two sets were varied independently. The number of subsequent subsumption tests required for classification as well as the total running times (including both classification and satisfiability testing) for each implementation are given in Table 12.3.

Table 12.3: Algorithm Comparison

Relation Size		ET		New	
Known	Possible	Tests	Seconds	Tests	Seconds
335 476	335 476	0	190	0	17
335 476	2 244 050	152 362	246	24 796	22
335 476	4 147 689	303 045	257	49 308	31
335 476	6 046 804	455 054	292	73 945	33
335 476	7 940 847	606 205	305	98 613	34
251 880	335 476	80 878	634	19 773	28
251 880	2 244 050	439 002	740	50 143	32
251 880	4 147 689	794 513	809	79 038	40
251 880	6 046 804	1 151 134	836	107 416	46
251 880	7 940 847	1 506 752	919	136 190	50
168 052	335 476	143 913	1079	62 153	62
168 052	2 244 050	673 768	1267	146 823	91
168 052	4 147 689	1 201 904	1320	226 670	93
168 052	6 046 804	1 729 553	1414	304 784	98
168 052	7 940 847	-	-	381 330	130

As the table shows, our simple implementation of the enhanced traversal algorithm (ET) is substantially slower than the new algorithm even when complete information is available; this is the result of the "insertion sort" behavior of ET described in Chapter 11.

When complete information is not available, our algorithm consistently reduces the number of subsumption tests needed to fully classify the ontology by an order of magnitude.

12.2.2 Overall Performance

In a second experiment, we extended HermiT with both our taxonomy-construction algorithm and our subsumption-information-extraction techniques and compared this implementation with the widely-used Description Logic classifiers FaCT++ and Pellet. Both of these other systems are quite mature and implement a wide range of optimizations to both taxonomy construction and subsumption reasoning; we were thus able to compare our new algorithm with existing state-of-the-art implementa-

tions.

We performed tests using not only NCI, but also the Gene Ontology (GO) and a version of the GALEN ontology of medical terminology.[8] Both NCI and GO have been specifically constructed to fall within the language fragment which existing reasoners are able to classify quickly; GALEN, in contrast, necessitates substantially more difficult subsumption testing but contains an order of magnitude fewer class names.

In order to estimate how the different systems would behave with more expressive ontologies, for each ontology $\mathcal{O}$ we constructed two extensions: $\mathcal{O}^{\exists}$, which adds the single axiom $\top \sqsubseteq \exists R.A$ for a fresh property name R and fresh class name A, and $\mathcal{O}^{\sqcup}$ which adds the axiom $\top \sqsubseteq A \sqcup B$ for fresh class names A and B. For NCI we constructed a further extension $\text{NCI}^{\exists\forall}$ by adding the axioms $\top \sqsubseteq \exists R.A$ and $C \sqsubseteq \forall R.B$ for each of the 17 most general class names C occurring in the ontology.[9]

Each of these extensions increases the complexity of individual subsumption tests and reduces the effectiveness of optimizations that try to avoid performing some or all of the tests that would otherwise be needed during classification. The addition of a single GCI containing an existential, for example, prevents the application of structural subsumption algorithms, such as the completely-defined-concept optimization used by FaCT++ [Tsarkov *et al.*, 2007], while adding disjunction to the knowledge base prohibits the use of reasoning algorithms specific to deterministic KBs, such as the one implemented by Pellet. The combination of existentials and universals reduces the applicability of the pseudo-model merging method [Haarslev *et al.*, 2001a; Horrocks, 1997], which is used by both reasoners to avoid subsumption testing.

The number of class names occurring in each ontology as well as the number of

[8] Both of these ontologies were included in the tests described in Section 12.1 as a part of the Gardiner corpus, but all four reasoners tested were able to classify them in similar times, so they were not listed among the "interesting" ontologies. The results for both of these ontologies are listed under the names **go-daily-termdb.owl.20Feb06** and **horrocks-OWL-Ontologies-galen.owl** in Appendix A.

[9] These test ontologies are available at `http://www.comlab.ox.ac.uk/people/Rob.Shearer/2010/classification-benchmarks.tgz`.

Table 12.4: System Comparison

Ontology	Classes	FaCT++		Pellet		HermiT	
		Tests	Seconds	Tests	Seconds	Tests	Seconds
NCI	27 653	4 506 097	2.3	-	16.1	27 653	22
NCI$^{\exists}$	27 654	8 658 610	4.4	-	16.7	27 654	21.0
NCI$^{\sqcup}$	27 655	8 687 327	5.1	10 659 876	95.4	48 389	37.0
NCI$^{\exists\forall}$	27 656	18 198 060	473.9	10 746 921	1098.3	27 656	20.8
GO	19 529	26 322 937	8.6	-	6.0	19 529	9.2
GO$^{\exists}$	19 530	26 904 495	12.7	-	6.9	19 530	9.7
GO$^{\sqcup}$	19 531	26 926 653	15.5	21 280 377	170.0	32 614	15.2
GALEN	2749	313 627	11.1	131 125	8.4	2749	3.3
GALEN$^{\exists}$	2750	327 756	473.5	170 244	9.7	2750	3.5
GALEN$^{\sqcup}$	2751	329 394	450.5	175 859	9.8	4657	40.5

tests performed (including all class satisfiability and subsumption tests) and the total time taken by each reasoner to fully classify each ontology are shown in Table 12.4. The Pellet system makes use of a special-purpose reasoning procedure for ontologies that fall within the $\mathcal{EL}$ fragment [Baader *et al.*, 2005]. This procedure computes all subsumptions using a single computation and does not perform individual subsumption tests; for such ontologies we do not, therefore, list the number of subsumption tests performed by Pellet.

As Table 12.4 shows, HermiT's new classification algorithm dramatically reduces the number of subsumption tests performed when classifying these ontologies. This does not, however, always result in faster performance. This is largely due to optimizations used by the other reasoners which greatly reduce the cost of subsumption testing for simple ontologies: the overwhelming majority of subsumption tests performed by FaCT++, for example, can be answered using the pseudo-model merging technique described by Horrocks [1997] and by Haarslev *et al.* [2001a].

Most of these optimizations could equally well be used in HermiT, but in the existing implementation each subsumption test performed by HermiT is far more costly. The number of subsumption tests performed by HermiT is, however, far smaller than for the other reasoners, and its performance also degrades far more gracefully as the

complexity of an ontology increases: adding a single GCI or disjunction to an ontology can prevent the application of special-case optimizations in Pellet and FaCT++, greatly increasing the cost of subsumption testing and, due to the very large number of tests being performed, vastly increasing the time required for classification. The $\text{NCI}^{\exists\forall}$ ontology, for example, eliminates any benefit from the pseudo-model merging optimization (since no two pseudo-models can be trivially merged), and this causes the classification time to increase by roughly two orders of magnitude for both Pellet and FaCT++. In contrast, HermiT's classification time is unaffected. The relatively poor performance of HermiT on the $\text{GALEN}^{\sqcup}$ ontology is due to the fact that the underlying satisfiability testing implementation is not highly optimized for the case when there are large numbers of branching points, even if no backtracking is actually required. The GALEN ontology induces notoriously large models with large numbers of individuals in the ABox, and the additional GCI forces the creation of a new branching point for every individual in the ABox.

Chapter 13

Conclusion

Our goal in this thesis has been to develop techniques for Description Logic reasoning which allow for the implementation of reasoners that exhibit superior classification performance on the types of ontologies encountered in practice. In particular, we investigated strategies for minimizing inefficiencies in common tableau consistency testing algorithms arising as a result of or-branching, and-branching, and at-most branching. Further, we explored new approaches to the implementation of classification services for description logics.

We chose to focus on techniques applicable to classifying knowledge bases encoded in $\mathcal{SROIQ}$, a highly expressive description logic that forms the logical underpinning of the semantic web language OWL 2. $\mathcal{SROIQ}$ includes support for inverse roles, nominals, number restrictions, and complex role inclusion axioms (among other constructs), and the combination of these features presents unique challenges in the design of reasoning systems.

Although sound, complete, and efficient handling of $\mathcal{SROIQ}$ ontologies was a key objective, many of the techniques developed can be usefully applied to other logics and in other settings.

13.1 Contributions

Part II of this thesis presented a novel hypertableau-based calculus for reasoning over

knowledge bases encoded in the description logic $\mathcal{SROIQ}$. Our new calculus is based on translating a $\mathcal{SROIQ}$ knowledge base into a set of assertions and a set of first-order clauses of a particular form. The clauses are then repeatedly matched against the set of assertions, which is extended at each step. Given a knowledge base $\mathcal{K}$, our calculus enables the derivation of a representation of a model of $\mathcal{K}$ in nondeterministic triple exponential time, if such a model exists.

This calculus offers a number of advantages over previously-known algorithms for DL consistency testing. The translation to clauses results in far less nondeterminism (or-branching) than traditional tableau algorithms; in many cases, including those for all Horn knowledge bases, hypertableau reasoning is entirely deterministic. Our algorithm's use of anywhere blocking also substantially reduces the sizes of the ABoxes generated in the course of a derivation, mitigating the effect of and-branching on performance. Finally, our novel *NI*-rule eliminates inefficiencies due to at-most branching. We have shown that our calculus is amenable to a wide range of optimizations, and can be adapted to deal very efficiently with a number of logics less expressive than $\mathcal{SROIQ}$.

Part III presented a new approach to classification based on several novel techniques. We provided a procedure for combining the results of individual binary tests (such as subsumption tests) to derive the maximum possible information about a quasi-ordering (such as a subsumption relation). Using this technique, we described a general-purpose algorithm for reducing the number of individual tests required to compute a taxonomy. Our new approach is applicable in a wide variety of domains beyond the classification of concept names in a Description Logic knowledge base.

We also provided techniques specific to knowledge base classification which go beyond the use of a consistency testing procedure as a black box. We showed how models generated during consistency testing can be used as rich sources of subsumption and non-subsumption information. In many cases, our approach is able to derive com-

plete information about the subsumption relation using an extremely small number of consistency tests.

We implemented our new techniques in a prototypical reasoner called HermiT; we analyzed various aspects of this implementation's performance in Part IV. We showed that across a wide range of test data, HermiT does not perform significantly worse than reasoners based on more traditional approaches, and often performs much better. In some cases, HermiT is able to classify ontologies that no other reasoner can handle.

13.2 Significance

As ontologies written in OWL 2 are becoming increasingly common and widely-used, the need for highly-efficient $\mathcal{SROIQ}$ reasoners is only increasing. The approach to DL reasoning described in this thesis represents a new state of the art, and the foundation upon which we expect future implementations to be based.

Even mature existing implementations have begun adoption of the novel techniques described here. Newer versions of the FaCT++ and Pellet reasoners, for example, now use our anywhere blocking strategy instead of the ancestor blocking used previously, and the developers of both reasoners have articulated plans to implement our new classification algorithm.

The HermiT system developed as a demonstration of our techniques has already become one of the most widely-deployed reasoners available. In fact, HermiT is now the standard reasoner distributed with the Protégé editor for OWL.

13.3 Future Work

Despite the performance improvements observed due to the work presented here, there are still ontologies encountered in practice that defeat HermiT (as well as traditional

tableau reasoners). Our testing indicates that while our techniques for reducing nondeterminism have been very effective, ontologies with large numbers of cyclic axioms are able to induce extremely large ABoxes due to and-branching, even when anywhere blocking is employed. We thus expect future work to focus on even more sophisticated blocking strategies and other techniques to reduce the sizes of generated ABoxes.

Our classification algorithm has proven to be a huge improvement over previously-known techniques for taxonomy construction, however substantial opportunities for further research remain. Our procedure for combining partial information to discover tighter limits on the subsumption relation has been shown to be optimal, however meaningful complexity bounds on the number of tests required to fully compute a taxonomy are elusive. As we have seen, a completely naïve search routine is optimal if only the number of elements in the taxonomy is considered; bounds based on more sophisticated metrics (e.g. the size of the subsumption relation or its transitive reduction) may be more enlightening.

Further, preliminary testing demonstrates that when significant partial information is available, the "most subsumption tests return false" heuristic upon which the top-down/bottom-up taxonomy search procedure is based may not be applicable; in many cases a completely random choice of subsumption tests performs equally well or better. An exploration of the fragments of realistic ontologies' subsumption relations which are not immediately derived as partial information could yield more appropriate heuristics.

Finally, although the calculus presented in this thesis is designed to efficiently handle large ABoxes, we have not focused on optimizations specific to either ontologies containing large numbers of individual names or ABox-specific reasoning tasks such as realization or query answering. We believe that our techniques are easily adapted to such problems, and suggest in particular that information about named individuals could be extracted from the ABoxes derived by our calculus in the same way that

we extract subsumption information. Such optimizations, as well as the implementation of previously-known ABox optimizations within the HermiT reasoner, are left to future work.

Appendix A

Reasoning Performance Data

We provide here timing data for all reasoning tests performed as a part of the evaluation described in Section 12.1. A full description of the testing procedure is provided in Section 12.1.2, and an analysis of these results is presented in Section 12.1.3.

Ontology	Classification Times (msec)			
	HermiT	HT-Anc	Pellet	FaCT++
cob.owl	112	141	75	10
disease-ontology.owl	115	125	64	9
ncbi-taxonomy.owl	118	135	75	9
mosquito-insecticide-resistance.owl	131	146	74	16
oboInOwl.owl	139	140	81	14
rdf-imsmd-technicalv1p2	140	169	75	10
psi-mi.owl	141	151	68	14
zea-mays-anatomy.owl	143	139	73	11
2001-05-rdf-ds-datastore-schema	145	169	101	10
rdf-imsmd-annotationv1p2	147	127	76	7
dvm-daml-exp-ont.daml	147	151	82	12
human-phenotype.owl	148	127	72	14
gem-elements-TomsSmallOnt.owl	149	144	73	8
	149	230	58	10
gem-gemtype-psi-mod.owl	153	151	125	10
	154	139	60	12
ont-kissology	155	168	65	7
2001-02-acls-ns	157	171	88	11
DAML-ArtOntology.daml	160	154	113	17
2003-01-geo-wgs84-pos	160	176	81	10
ontologies-ittalks-assertions	161	194	87	9
DAML-Imaging.daml	161	164	103	8
rdf-imsmd-generalv1p2	161	145	91	10

Note: an entry — means that reasoner was unable to classify the ontology either due to time out or memory exhaustion.

Ontology	Classification Times (msec)			
	HermiT	HT-Anc	Pellet	FaCT++
dvm-daml-agent-ont.daml	163	156	83	9
projects-plus-DAML-onts-cs1.1.daml	165	169	120	14
rdf-imsmd-metametadatav1p2	166	169	64	8
2002-04-geonames-geonames-ont	166	190	100	31
2005-04-wikipedia-wikiont.owl	168	239	103	11
daml-ontologies-sri-basic-1-0-Time.daml	169	155	86	13
ImageFingerprintingOntology-web.daml	169	177	94	12
rdf-imsmd-rightsv1p2	169	135	74	8
2002-04-classification-classification-ont	169	167	92	11
2000-10-swap-util-sniffSchema	170	150	83	7
2001-03-daml+oil	170	217	166	15
pharmacogenomics.owl	171	149	61	9
schemas-meta-rdf-	171	166	85	14
rdf-imsmd-rootv1p2	173	167	79	8
net-bloggercode-	173	191	98	13
net-schemas-quaffing-	173	168	86	8
transmission.owl	174	181	95	16
2000-01-rdf-schema	174	184	76	10
BriefingsOntology.daml	175	179	111	15
2003-02-usps-usps-ont.owl	176	147	81	8
gem-A2AStandard-	177	152	94	10
2000-10-annotationType	177	152	94	9
net-inkel-rdf-schemas-lang-1.1	177	201	80	11
gem-NISO-Z3919-	177	153	88	10
dvm-daml-bib-ont.daml	177	142	82	10
ontos-compontos-tourism-I3.daml	178	198	246	50
atlas-employment-categories.daml	178	173	116	16
2001-06-expenses-amex-ont	179	204	94	20
ont-2004-01-rcc	179	203	86	11
dc-dcmitype-	179	180	165	10
ont-homework-atlas-publications.daml	181	181	87	14
DAML-pptOntology.daml	181	149	88	10
ro-proposed.owl	182	171	91	15
library-wordnet-wordnet-20000620.rdfs	182	191	91	18
ontologies-ittalks-event	182	186	92	8
2004-02-skos-mapping	183	199	87	9
2001-09-countries-iso-3166-ont	183	224	87	8
relationship.owl	184	174	72	30
2000-03-13-eor	184	245	96	9
ontologies-ittalks-topic	185	209	84	18
2002-03-ranks-rank-ont	185	215	73	9
ont-homework-atlas-date.daml	187	196	111	15

Note: an entry — means that reasoner was unable to classify the ontology either due to time out or memory exhaustion.

Ontology	Classification Times (msec)			
	HermiT	HT-Anc	Pellet	FaCT++
services-owl-s-1.1-Service.owl	187	181	96	11
2000-10-daml-ont	187	202	109	15
community-xmlns-2006-gallery	188	203	96	9
ro-ucdhsc.owl	188	162	75	17
2001-09-countries-fips-10-4-ont	188	184	106	10
2002-09-milservices-milservices-ont	188	217	91	24
golbeck-daml-baseball.daml	188	181	77	12
golbeck-daml-running.daml	189	196	116	15
2001-02-projectplan-projectplan	189	215	76	12
2000-10-annotation-ns	190	178	81	8
RDF-relational.owl	190	230	106	10
dc-elements-1.1-	190	194	96	10
ontology-research.owl	190	233	144	10
8080-umls-UMLSinDAML-NET-SRDEF.daml	190	254	149	22
geoCoordinateSystems20040307.owl	192	170	101	8
ontos-compontos-tourism-II1.daml	192	237	307	65
1999-02-22-rdf-syntax-ns	193	187	82	13
2002-11-08-ccpp-schema	193	168	82	10
2002-10-sndl-unit-ont	193	207	102	12
ont-2004-01-agent	193	210	79	11
2004-12-q-contentlabel	193	163	80	10
2001-10-office-office	194	221	125	10
rdf-imsmd-lifecyclev1p2	194	182	95	9
2002-03-usnships-ship-ont	195	211	87	20
communityreview-scientific-review-o	195	255	91	38
ontos-compontos-tourism-III1.daml	196	220	255	46
ontos-compontos-tourism-II4.daml	197	205	232	44
projects-plus-DAML-onts-cs1.0.daml	197	210	191	16
http—amk.ca-xml-review-1.0	198	207	69	11
2002-02-chiefs-chiefs-ont	198	197	103	12
dav-ontologies-policyContainmentTest.owl	198	188	135	11
2001-06-expenses-check-ont	199	201	90	26
rdf-imsmd-educationalv1p2	199	152	96	10
2002-03-darpadir-darpadir-ont	200	215	93	14
ont-homework-cmu-ri-courses-ont.daml	201	255	191	35
ont-homework-atlas-cmu.daml	202	211	144	16
glapizco-technical.owl	203	189	140	10
2000-10-swap-pim-doc.rdf	203	175	95	19
projects-plus-DAML-onts-beer1.0.daml	203	226	155	20
2003-06-sw-vocab-status-ns	203	193	72	9
2001-10-html-airport-ont	203	198	83	8
sioc-ns	204	472	111	9

Note: an entry — means that reasoner was unable to classify the ontology either due to time out or memory exhaustion.

Ontology	Classification Times (msec)			
	HermiT	HT-Anc	Pellet	FaCT++
HomeWork1-ResearchProjectOntology.daml	204	196	109	11
DAML-ATO98MessageSet-Ontology.owl	205	330	101	11
2001-06-expenses-eecr-ont	205	201	92	32
2002-07-owl	208	176	157	14
webscripter-project.o.daml	208	250	168	21
projects-plus-DAML-onts-personal1.0.daml	209	183	105	20
projects-DAML-ksl-daml-desc.daml	210	253	152	15
net-schemas-book	211	206	75	10
research-AgentCities-ontologies-pubs	211	213	184	27
webscripter-todo.o.daml	212	272	200	17
ontos-compontos-tourism-II3.daml	212	203	250	53
2001-06-expenses-trip-ont	213	233	99	33
2002-08-nasdaq-nasdaq-ont	213	250	104	32
2001-10-html-zipcode-ont	213	200	112	15
golbeck-web-trust.daml	214	199	102	10
2002-10-units-units-ont	215	214	106	12
webscripter-snapshot.o.daml	215	272	173	19
owl-gforge-site	216	253	124	14
ontology-project.owl	216	260	142	27
fungal-anatomy.owl	216	195	152	136
2001-01-gedcom-gedcom	217	249	146	14
2004-02-skos-extensions	217	182	73	12
ont-vcard	217	193	98	11
2002-03-metrics-metrics-ont	217	216	94	31
2002-05-mcda-mcda-ont	219	222	127	30
TestPizzaOntology.owl	219	238	200	13
ontology-contact.owl	219	249	96	21
projects-DAML-ksl-daml-instances.daml	220	226	121	13
webscripter-publication.o.daml	220	259	181	22
2003-05-subway-subway-ont	220	180	119	11
ont-homework-cmu-ri-project-ont.daml	220	235	185	17
2001-10-html-airport-ont	222	218	83	8
projects-plus-DAML-onts-general1.0.daml	223	188	157	19
ontos-compontos-tourism-III3.daml	223	233	274	67
webscripter-person.o.daml	224	246	152	15
ontology-news.owl	224	267	96	21
ont-homework-cmu-ri-center-ont.daml	224	233	194	18
2000-10-swap-pim-contact	225	226	163	23
2002-02-telephone-1-areacodes-ont	225	217	101	38
2001-10-cvslog-cvslog-ont	225	274	101	30
yzou1-daml-acldaml.daml	225	232	136	33
ont-trust.owl	226	203	89	10

Note: an entry — means that reasoner was unable to classify the ontology either due to time out or memory exhaustion.

Ontology	Classification Times (msec)			
	HermiT	HT-Anc	Pellet	FaCT++
biosphere.owl	226	320	107	11
net-ontology-beer	227	239	252	13
bilateria-mrca.owl	227	274	309	18
ontology-person.owl	227	264	207	21
plasmodium-life-cycle.owl	228	249	126	18
projects-plus-DAML-onts-univ1.0.daml	228	189	167	14
communityreview-abstract-review-o	228	247	109	37
2002-10-hazardous-hazardous-cargo-ont	228	229	169	20
bfo.owl	228	266	185	18
DAML-DynamicOntology1.owl	229	204	143	12
2001-03-daml+oil	230	195	164	12
tools-tools-ont	230	224	106	36
webscripter-person.o.daml	232	204	154	12
tools-tools-ont	232	220	107	22
gem-qualifiers-	232	265	237	13
projects-plus-DAML-onts-docmnt1.0.daml	232	199	107	23
webscripter-division.o.daml	237	221	157	18
ont-2004-01-time	237	282	139	24
2003-vegetarian.owl	238	233	228	12
2000-10-swap-pim-contact.rdf	239	200	162	42
2004-08-Presentations.owl	240	227	148	20
ont-2004-01-person	241	306	141	23
ont-homework-cmu-ri-people-ont.daml	241	251	202	21
DAML-terroristAttackTypes.daml	242	241	225	42
yzou1-daml-acl.daml	242	238	164	25
ontology-association.owl	243	271	99	10
worm-development.owl	244	227	137	14
2002-03-agents-mcda	245	239	117	24
2002-12-cal-ical	246	271	188	18
HomeWork3-SurveyOntology.daml	246	280	215	42
projects-integration-projects-20010811	247	256	138	33
ontology-conference.owl	247	263	94	22
2003-04-agents-enpmap	248	214	83	26
webscripter-document.o.daml	249	256	193	14
plugins-owl-owl-library-camera.owl	250	186	128	15
ro-bfo-bridge.owl	251	243	244	26
ontology-event.owl	251	289	114	21
2001-10-html-nyse-ont	251	225	85	39
ontologies-talk-ont	252	322	129	37
golbeck-web-www04photo.owl	252	242	160	22
DAML-Military.owl	253	246	207	46
2001-03-earl-0.95.rdf	253	254	240	45

Note: an entry — means that reasoner was unable to classify the ontology either due to time out or memory exhaustion.

Ontology	Classification Times (msec)			
	HermiT	HT-Anc	Pellet	FaCT++
dc-terms-	255	223	228	15
2001-10-html-nyse-ont	256	219	92	21
ont-homework-cmu-ri-labgroup-ont.daml	258	297	174	15
2003-owl-geo-geoFeatures20040307.owl	261	236	223	12
ont-homework-cmu-ri-publications-ont.daml	263	356	229	38
ro-bfo-bridge1-1.owl	263	265	262	18
HomeWork3-BriefingOntology.daml	264	218	178	13
ontologies-ittalks-person	264	265	113	20
evidence-code.owl	268	269	177	35
2003-01-movienight-movienight-ont	269	222	122	39
caro.owl	269	255	218	23
ont-USRegionState.daml	269	257	280	26
2004-02-skos-core	269	243	228	31
owl-library-shuttle-crew-ont.owl	271	269	220	14
2001-10-agenda-agenda-ont	272	265	148	36
2001-08-baseball-baseball-ont	272	271	287	30
net-rss-2.0-enc	273	254	94	13
ont-currency.daml	275	260	282	36
net-ontology-order.owl	275	235	132	21
DAML-Government.owl	275	267	516	75
infectious-disease-ontology.owl	283	278	194	15
ontologies-ittalks-talk	284	286	118	23
image.owl	286	356	345	17
CinemaAndMovies.daml	288	237	188	54
webscripter-event.o.daml	288	317	229	21
ontology-photo.owl	289	275	91	22
projects-plus-DAML-onts-tseont.daml	289	303	173	44
dictyostelium-discoideum-anatomy.owl	292	389	310	34
ontologies-profile-ont	294	320	156	24
DAML-Elements.owl	294	327	356	41
2002-03-agents-agent-ont	295	301	208	41
ontologies-ittalks-address	300	212	66	8
fly-development.owl	301	3471	519	5918
DAML-ATO-Mission-Models.owl	301	295	331	24
plugins-owl-owl-library-koala.owl	302	286	185	31
temporal-gramene.owl	303	302	307	57
DAML-ATO-Ontology.owl	304	357	477	43
ontology-publication.owl	306	352	195	23
cmu-ri-employmenttypes-ont.daml	307	391	320	40
yeast-phenotype.owl	309	321	218	28
2003-02-UserModelOntology.daml	309	305	299	55
ns-doap	310	288	207	14

Note: an entry — means that reasoner was unable to classify the ontology either due to time out or memory exhaustion.

Ontology	Classification Times (msec)			
	HermiT	HT-Anc	Pellet	FaCT++
space-0.1-	313	278	143	12
mao.owl	319	305	230	36
spatial.owl	320	329	1020	64
8080-ontologies-wsdl-ont.daml	322	290	304	28
2004-02-wsa-ServiceModel.owl	325	321	339	37
gold	326	378	416	19
2001-06-map-map-ont	327	326	160	38
DAML-Communications.owl	327	312	346	50
mdabulaish-woodontology-woodontology.xml	329	313	245	14
plugins-owl-owl-library-generations.owl	331	362	168	15
DAML-WMD.owl	332	332	449	41
2004-02-wsa-ResourceModel.owl	337	357	357	28
2004-02-wsa-Extensions.owl	338	334	378	40
caro-to-bfo.owl	342	312	387	25
2004-02-wsa-MessageModel.owl	343	322	356	24
http—oiled.man.ac.uk-ontologies-ka	343	352	366	63
2004-02-wsa-PolicyModel.owl	347	318	354	23
BPMO-2004-03-03-cdl-Countries	351	346	180	42
2003-owl-geo-geoRelations20040307.owl	352	354	270	—
plugins-owl-owl-library-ka.owl	358	373	304	31
systems-biology.owl	362	372	326	43
horrocks-OWL-Ontologies-ka.owl	370	308	352	27
2000-10-swap-infoset-infoset-diagram.rdf	374	317	410	75
po-temporal.owl	375	382	424	153
release-biopax-level1.owl	380	373	295	25
ontology-support	382	356	193	39
mouse-pathology.owl	383	403	227	26
2001-06-itinerary-itinerary-ont	385	396	168	50
2003-09-factbook-factbook-ont	385	397	285	70
PizzaOntology.owl	392	388	193	235
9090-RDF-VRP-Examples-moviedatabase.rdf	393	442	277	26
pan-damltime-time-entry.owl	393	466	201	43
2003-07-umls-	396	420	579	50
ranwezs-ontologies-soccerV2.0.daml	399	418	400	21
mkhedr-Ontologies-Fuzzy	408	402	274	—
spider-anatomy.owl	413	462	324	89
amphibian-anatomy.owl	419	557	386	158
Downloads-Level2v0.94-biopax-level2.owl	429	448	305	39
2001-11-IsaViz-graphstylesheets	429	427	224	15
plugins-owl-owl-library-people+pets.owl	435	432	347	—
horrocks-OWL-Ontologies-mad-cows.owl	437	450	374	26
ontologies-booze	449	420	751	—

Note: an entry — means that reasoner was unable to classify the ontology either due to time out or memory exhaustion.

Ontology	Classification Times (msec)			
	HermiT	HT-Anc	Pellet	FaCT++
Aingeru-OperOnt.owl	462	538	604	86
plant-environment.owl	466	454	459	75
flybase-vocab.owl	467	481	332	47
services-owl-s-1.1-Process.owl	477	509	301	37
Files-DLPOnts-DOLCE-Lite-397.owl	481	474	371	41
2004-owl-mindswappers	482	391	492	39
loggerhead-nesting.owl	483	396	288	51
ontologies-2006-01-copyrightonto.owl	491	507	527	21
horrocks-Ontologies-tambis.daml	496	498	960	273
amino-acid-2005-10-11-amino-acid.owl	496	496	644	72
2003-07-cns-	496	488	698	110
rex.owl	498	485	479	93
services-owl-s-1.1-Profile.owl	499	463	366	36
DAML-Economy.owl	501	462	586	77
DAML-Transportation.owl	507	504	710	51
ont-USCity.daml	509	487	528	143
pathway.owl	518	480	492	84
plugins-owl-owl-library-travel.owl	518	481	295	—
tick-anatomy.owl	529	545	518	231
ontologies-pizza-pizza-20041007.owl	531	514	623	46
evoc.owl	534	533	454	59
plugins-owl-owl-library-resume.owl	606	607	734	—
protein.owl	612	639	851	270
plant-trait.owl	622	615	712	184
services-owl-s-1.1-Profile.owl	624	454	348	51
services-owl-s-1.1-Grounding.owl	633	472	357	49
envo.owl	694	820	866	548
po-anatomy.owl	695	732	605	218
xenopus-anatomy.owl	709	659	954	271
cereal-anatomy.owl	715	641	647	287
DAML-SUMO.owl	721	711	813	101
2002-01-p3prdfv1	726	565	548	82
ont-AirportCodes.daml	814	805	768	381
fix.owl	842	897	570	143
cell.owl	849	760	815	175
DAML-Mid-level-ontology.owl	859	843	1796	190
quality-prerelease.owl	861	898	688	220
wbs-ontology-2004-08-fgdc-csdgm.owl	876	854	1004	81
resource-drm-orel-orel0-5.owl	892	651	716	101
human-dev-anat-abstract.owl	902	849	792	858
quality.owl	914	874	714	233
psi-ms.owl	940	857	827	202

Note: an entry — means that reasoner was unable to classify the ontology either due to time out or memory exhaustion.

Ontology	Classification Times (msec)			
	HermiT	HT-Anc	Pellet	FaCT++
mosquito-anatomy.owl	953	948	2120	3883
unit.owl	954	991	1008	245
worm-phenotype.owl	1024	829	927	306
doc-chimaera-ontologies-wines.daml	1024	1019	653	51
sequence.owl	1051	1061	3858	1007
sequence-xp.owl	1069	1046	3709	1008
teleost-anatomy.owl	1091	1284	16526	1750
adult-mouse-anatomy.owl	1126	1097	1077	581
go-xrf-metadata.owl	1146	1132	606	594
sequence-prerelease.owl	1174	1134	3917	1072
provenance.owl	1180	1147	—	—
how-HydrologicUnits-2003-09-hu	1229	1345	1168	443
ontology-portal	1251	1334	1384	135
event.owl	1253	1248	1530	895
medaka-anatomy-development.owl	1269	1240	1310	3433
zebrafish-anatomy-prerelease.owl	1315	1332	1954	1470
brenda.owl	1353	1262	1437	2316
zebrafish-anatomy.owl	1472	1391	2004	1572
kmi-basic-portal-ontology.owl	1561	2524	813	—
cellular-component.owl	1571	1548	1580	817
cellular-component-xp-self.owl	1657	1436	1679	679
sao.owl	1734	1561	1661	262
2001-02-geofile-geofile-ont	1805	1834	23556	11679
uberon.owl	1869	1955	2322	446078
worm-anatomy.owl	2189	2244	3001	18777
human-dev-anat-staged.owl	2194	2045	5320	48323
ontologies-IEDMv1.0.owl	2409	2175	2424	414
gene-regulation.owl	2428	7929	11247	236
molecular-function.owl	2748	2916	14143	—
mammalian-phenotype.owl	2760	2782	5434	2367
Files-DLPOnts-ExtDnS-397.owl	2775	—	38323	711
emap.owl	3279	3827	8076	64977
fly-taxonomy.owl	3316	3070	3074	424
molecule-role.owl	3328	3391	25744	304549
quality-bfo-bridge.owl	3498	3495	1557	250
horrocks-OWL-Ontologies-galen.owl	4285	5155	12091	14314
ontologies-tambis-full.owl	4308	127208	1698	—
fly-anatomy.owl	4741	5352	64836	2637
plugins-owl-owl-library-not-galen.owl	4894	5788	90515	—
molecular-function-xp-uber-anatomy.owl	5442	4886	—	86038
not-galen.owl	6882	6249	86342	—
go-daily-termdb.owl.20Feb06	8195	7569	9695	8676

Note: an entry — means that reasoner was unable to classify the ontology either due to time out or memory exhaustion.

Ontology	Classification Times (msec)			
	HermiT	HT-Anc	Pellet	FaCT++
biological-process.owl	8326	9639	18639	11602
go-xp-regulation.owl	8748	9730	19866	12716
galen-ians-full-doctored.owl	8833	456269	—	15871
biological-process-xp-cell.owl	9512	10354	42590	12285
plugins-owl-owl-library-MGEDOntology.owl	10102	—	2590	—
bp-xp-cell.owl	11599	11082	54698	15333
biological-process-xp-self.owl	11930	11002	51095	13889
bp-xp-cellular-component.owl	12580	12340	55729	17034
biological-process-xp-cellular-component.owl	12643	12129	47868	15217
biological-process-xp-plant-anatomy.owl	12848	11178	87172	22884
sweet-phenomena.owl	13494	11198	—	208
molecular-function-xp-regulators.owl	14085	17126	35354	66570
2003-CancerOntology-nciOncology.owl	14096	14357	23279	2977
2003-04-01-cyc	14473	15290	—	—
biological-process-xp-uber-anatomy.owl	16338	15439	—	102884
cellular-component-xp-go.owl	18568	17995	40540	76679
teleost-taxonomy.owl	19949	18978	40876	1092120
mged.owl	23801	—	63954	—
chebi.owl	24222	—	—	397017
Files-DLPOnts-Information-397.owl	60459	—	63188	—
sweet-numerics.owl	76700	72650	3667	166
ontology-space.owl	—	93124	4120	151
fma-lite-prerelease.owl	104331	—	—	—
fma-lite.owl	107274	—	—	—
galen-ians-full-undoctored.owl	126316	—	—	—
gazetteer.owl	131891	132340	—	—
ncithesaurus.owl	—	—	172048	60654
ontologies-MGEDOntology.owl	—	—	190491	—
2001-sw-WebOnt-guide-src-wine	329752	547055	19786	190722
TR-2003-CR-owl-guide-20030818-food	331751	579475	20060	161298
TR-2003-CR-owl-guide-20030818-wine	343725	524568	19475	162099
FMA-constitutionalPartForNS.owl.20Feb06	—	—	—	616689
Files-DLPOnts-Plans-397.owl	1075099	—	105118	—

Note: an entry — means that reasoner was unable to classify the ontology either due to time out or memory exhaustion.

References

[Andrka *et al.*, 1998] Hajnal Andrka, Istvn Nmeti, and Johan van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27:217–274, 1998. 10.1023/A:1004275029985.

[Baader and Nutt, 2007] Franz Baader and Werner Nutt. Basic Description Logics. In Baader et al. [2007], chapter 2, pages 47–100.

[Baader and Sattler, 2001] Franz Baader and Ulrike Sattler. An Overview of Tableau Algorithms for Description Logics. *Studia Logica*, 69:5–40, 2001.

[Baader *et al.*, 1994] Franz Baader, Bernhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, and Enrico Franconi. An Empirical Analysis of Optimization Techniques for Terminological Representation Systems or: Making KRIS Get a Move on. *Applied Artificial Intelligence. Special Issue on Knowledge Base Management*, 4:270–281, 1994.

[Baader *et al.*, 1996] Franz Baader, Martin Buchheit, and Bernhard Hollunder. Cardinality Restrictions on Concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.

[Baader *et al.*, 2005] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ Envelope. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 364–369, Edinburgh, UK, July 30–August 5 2005. Morgan Kaufmann Publishers.

[Baader *et al.*, 2007] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2nd edition, August 2007.

[Baader, 2003] Franz Baader. Terminological Cycles in a Description Logic with Existential Restrictions. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 325–330. Morgan Kaufmann, 2003.

[Balsiger and Heuerding, 1998] Peter Balsiger and Alain Heuerding. Comparison of Theorem Provers for Modal Logics — Introduction and Summary. In Harrie de Swart, editor, *Proceedings of the 2nd International Conference on Analytic Tableaux and Related Methods (TABLEAUX'98)*, volume 1397 of *LNAI*, pages 25–26. Springer, 1998.

[Balsiger *et al.*, 2000] Peter Balsiger, Alain Heuerding, and Stefan Schwendimann. A Benchmark Method for the Propositional Modal Logics K, KT, S4. *Journal of Automated Reasoning*, 24(3):297–317, 2000.

[Baumgartner and Schmidt, 2006] Peter Baumgartner and Renate A. Schmidt. Blocking and Other Enhancements for Bottom-Up Model Generation Methods. In Ulrich Furbach and Natarajan Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *LNCS*, pages 125–139, Seattle, WA, USA, August 17–20 2006. Springer.

[Baumgartner *et al.*, 1996] Peter Baumgartner, Ulrich Furbach, and Ilkka Niemelä. Hyper Tableaux. In *Proceedings of the European Workshop on Logics in Artificial Intelligence (JELIA '96)*, number 1126 in LNAI, pages 1–17, Évora, Portugal, September 30–October 3 1996. Springer.

[Baumgartner *et al.*, 2008] Peter Baumgartner, Ulrich Furbach, and Björn Pelzer. The Hyper Tableaux Calculus with Equality and an Application to Finite Model Computation. *Journal of Logic and Computation*, 2008.

[Borgida, 1996] Alex Borgida. On the Relative Expressiveness of Description Logics and Predicate Logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.

[Bry and Torge, 1998] Franois Bry and Sunna Torge. A Deduction Method Complete for Refutation and Finite Satisfiability. In Jürgen Dix, Luis F. del Cerro, and Ulrich Furbach, editors, *Proceedings European Workshop on Logics in Artificial Intelligence (JELIA '98)*, volume 1489 of *LNCS*, pages 122–138, Dagstuhl, Germany, October 12–15 1998. Springer.

[Buchheit *et al.*, 1993] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable Reasoning in Terminological Knowledge Representation Systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[Cormen *et al.*, 2001] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition.* The MIT Press and McGraw-Hill Book Company, 2001.

[Daskalakis *et al.*, 2007] Constantinos Daskalakis, Richard M. Karp, Elchanan Mossel, Samantha Riesenfeld, and Elad Verbin. Sorting and Selection in Posets. *CoRR*, abs/0707.1532, 2007.

[Derriere *et al.*, 2006] Sebastian Derriere, André Richard, and Andrea Preite-Martinez. An Ontology of Astronomical Object Types for the Virtual Observatory. In *Proceedings of the 26th meeting of the IAU: Virtual Observatory in Action: New Science, New Technology, and Next Generation Facilities*, pages 17–18, Prague, Czech Republic, August 21–22 2006.

[Ding and Haarslev, 2006] Yu Ding and Volker Haarslev. Tableau Caching for Description Logics with Inverse and Transitive Roles. In Bijan Parsia, Ulrike Sattler,

and David Toman, editors, *Proceedings of the 2006 International Workshop on Description Logics (DL 2006)*, volume 189 of *CEUR Workshop Proceedings*, Windermere, UK, May 30–June 1 2006.

[Donini and Massacci, 2000] Francesco M. Donini and Fabio Massacci. EXPTIME tableaux for $\mathcal{ALC}$. *Artificial Intelligence*, 124(1):87–138, 2000.

[Donini *et al.*, 1998] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. AL-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.

[Donini, 2007] Francesco M. Donini. Complexity of Reasoning. In Baader et al. [2007], chapter 3, pages 101–141.

[Ellis, 1991] Gerard Ellis. Compiled Hierarchical Retrieval. In *6th Annual Conceptual Graphs Workshop*, pages 285–310, 1991.

[Faddoul *et al.*, 2008] Jocelyne Faddoul, Nasim Farsinia, Volker Haarslev, and Ralf Möller. A Hybrid Tableau Algorithm for $\mathcal{ALCQ}$. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikos M. Avouris, editors, *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI 2008)*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 725–726, Patras, Greece, July 21-25 2008. IOS Press.

[Faigle and Turán, 1985] Ulrich Faigle and György Turán. Sorting and Recognition Problems for Ordered Sets. In Kurt Mehlhorn, editor, *STACS*, volume 182 of *Lecture Notes in Computer Science*, pages 109–118, Saarbrücken, Germany, January 3–5 1985. Springer.

[Gardiner *et al.*, 2006] Tom Gardiner, Ian Horrocks, and Dmitry Tsarkov. Automated Benchmarking of Description Logic Reasoners. In *Proceedings of the 2006 Description Logic Workshop (DL 2006)*, volume 189 of *CEUR Workshop Proceedings*, 2006.

[Georgieva *et al.*, 2003] Lilia Georgieva, Ullrich Hustadt, and Renate A. Schmidt. Hyperresolution for Guarded Formulae. *Journal of Symbolic Computation*, 36(1–2):163–192, 2003.

[Goble *et al.*, 2001] Carole A. Goble, Deborah L. McGuinness, Ralf Möller, and Peter F. Patel-Schneider, editors. *Working Notes of the 2001 International Description Logics Workshop (DL-2001)*, volume 49 of *CEUR Workshop Proceedings*, Stanford, CA, USA, August 1-3 2001. CEUR-WS.org.

[Golbreich *et al.*, 2006] Christine Golbreich, Songmao Zhang, and Olivier Bodenreider. The Foundational Model of Anatomy in OWL: Experience and Perspectives. *Journal of Web Semantics*, 4(3):181–195, 2006.

[Goodwin, 2005] John Goodwin. Experiences of using OWL at the Ordnance Survey. In *Proceedings of the 2005 International Workshop on OWL: Experiences and Directions (OWLED 05)*, volume 188 of *CEUR WS Proceedings*, Galway, Ireland, November 11–12 2005.

[Goré and Nguyen, 2007] Rajeev Goré and Linh Anh Nguyen. EXPTIME Tableaux with Global Caching for Description Logics with Transitive Roles, Inverse Roles and Role Hierarchies. In *Proceedings of the 16th International Conference on Automated Reasoning with Tableaux and Related Methods (TABLEAUX 2007)*, volume 4548 of *LNCS*, pages 133–148, Aix en Provence, France, July 3–6 2007. Springer.

[Haarslev and Möller, 2001a] Volker Haarslev and Ralf Möller. High Performance Reasoning with Very Large Knowledge Bases: A Practical Case Study. In Bernhard Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 161–166, 2001.

[Haarslev and Möller, 2001b] Volker Haarslev and Ralf Möller. Optimizing Reasoning in Description Logics with Qualified Number Restrictions. In Goble et al. [2001].

[Haarslev and Möller, 2001c] Volker Haarslev and Ralf Möller. RACER System Description. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR 2001)*, volume 2083 of *LNAI*, pages 701–706, Siena, Italy, June 18–23 2001. Springer.

[Haarslev *et al.*, 2001a] Volker Haarslev, Ralf Möller, and Anni-Yasmin Turhan. Exploiting Pseudo Models for TBox and ABox Reasoning in Expressive Description Logics. In *Proceedings of the 2001 International Joint Conference on Automated Reasoning (IJCAR 2001)*, pages 61–75, 2001.

[Haarslev *et al.*, 2001b] Volker Haarslev, Martina Timmann, and Ralf Möller. Combining Tableaux and Algebraic Methods for Reasoning with Qualified Number Restrictions. In Goble et al. [2001].

[Haarslev *et al.*, 2008] Volker Haarslev, Ralf Möller, and Sebastian Wandelt. The Revival of Structural Subsumption in Tableau-Based Description Logic Reasoners. In Franz Baader, Carsten Lutz, and Boris Motik, editors, *Proceedings of the 21st International Workshop on Description Logics (DL 2008)*, volume 353 of *CEUR Workshop Proceedings*, Dresden, Germany, May 13–16 2008.

[Hartel *et al.*, 2005] Frank W. Hartel, Sherri de Coronado, Robert Dionne, Gilberto Fragoso, and Jennifer Golbeck. Modeling a Description Logic Vocabulary for Cancer Research. *Journal of Biomedical Informatics*, 38(2):114–129, 2005.

[Horrocks and Patel-Schneider, 1998a] Ian Horrocks and Peter F. Patel-Schneider. Comparing Subsumption Optimizations. In Enrico Franconi, Giuseppe De Giacomo, Robert M. MacGregor, Werner Nutt, and Christopher A. Welty, editors,

Proceedings of the 1998 Description Logic Workshop (DL'98), volume 11 of *CEUR Workshop Proceedings*, pages 90–94, Povo–Trento, Italy, June 6–8 1998.

[Horrocks and Patel-Schneider, 1998b] Ian Horrocks and Peter F. Patel-Schneider. DL Systems Comparison. In Enrico Franconi, Giuseppe De Giacomo, Robert M. MacGregor, Werner Nutt, and Christopher A. Welty, editors, *Proceedings of the 1998 International Workshop on Description Logic (DL'98)*, volume 11 of *CEUR Workshop Proceedings*, pages 55–57, Povo–Trento, Italy, June 6–8 1998.

[Horrocks and Sattler, 2001] Ian Horrocks and Ulrike Sattler. Ontology Reasoning in the $\mathcal{SHOQ}(\mathbf{D})$ Description Logic. In Bernhard Nebel, editor, *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 199–204, Seattle, WA, USA, August 4–10 2001. Morgan Kaufmann Publishers.

[Horrocks and Sattler, 2004] Ian Horrocks and Ulrike Sattler. Decidability of $\mathcal{SHIQ}$ with Complex Role Inclusion Axioms. *Artificial Intelligence*, 160(1–2):79–104, December 2004.

[Horrocks and Sattler, 2007] Ian Horrocks and Ulrike Sattler. A Tableau Decision Procedure for $\mathcal{SHOIQ}$. *Journal of Automated Reasoning*, 39(3):249–276, 2007.

[Horrocks *et al.*, 2000a] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–263, 2000.

[Horrocks *et al.*, 2000b] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In David McAllester, editor, *Proceedings of the 17th International Conference on Automated Deduction (CADE-17)*, volume 1831 of *LNAI*, pages 482–496, Pittsburgh, USA, June 17–20 2000. Springer.

[Horrocks, 1997] Ian Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.

[Horrocks, 1998] Ian Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In Anthony G. Cohn, Lenhard Schubert, and Stuart C. Shapiro, editors, *Proceedings of the 6th International Conference on the Principles of Knowledge Representation and Reasoning (KR '98)*, pages 636–647, Trento, Italy, June 2–5 1998. Morgan Kaufmann Publishers.

[Horrocks, 2007] Ian Horrocks. Implementation and Optimization Techniques. In Baader et al. [2007], chapter 9, pages 313–358.

[Hudek and Weddell, 2006] Alexander K. Hudek and Grant Weddell. Binary Absorption in Tableaux-Based Reasoning for Description Logics. In Bijan Parsia, Ulrike Sattler, and David Toman, editors, *Proceedings of the 2006 International Workshop on Description Logics (DL 2006)*, volume 189 of *CEUR Workshop Proceedings*, Windermere, UK, May 30–June 1 2006.

[Hustadt and Schmidt, 1999] Ullrich Hustadt and Renate A. Schmidt. Issues of Decidability for Description Logics in the Framework of Resolution. In Ricardo Caferra and Gernot Salzer, editors, *Selected Papers from Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *LNAI*, pages 191–205. Springer, 1999.

[Hustadt *et al.*, 2005] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Data Complexity of Reasoning in Very Expressive Description Logics. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 466–471, Edinburgh, UK, July 30–August 5 2005. Morgan Kaufmann Publishers.

[Kazakov, 2008] Yevgeny Kazakov. RIQ and SROIQ are Harder than SHOIQ. In Gerhard Brewka and Jérôme Lang, editors, *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR 2008)*, pages 274–284. AAAI Press, 2008.

[Knublauch *et al.*, 2004] Holger Knublauch, Ray W. Fergerson, Natalya F. Noy, and Mark A. Musen. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In *Proceedings of the 2004 International Semantic Web Conference (ISWC 2004)*, pages 229–243, Hiroshima, Japan, November 7–11 2004.

[Kutz *et al.*, 2006] Oliver Kutz, Ian Horrocks, and Ulrike Sattler. The Even More Irresistible $\mathcal{SROIQ}$. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings of the 10th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 68–78, Lake District, UK, June 2–5 2006. AAAI Press.

[La Poutré and van Leeuwen, 1988] J. La Poutré and J. van Leeuwen. Maintenance of transitive closures and transitive reductions of graphs. In Herbert Gttler and Hana-Jrgen Schneider, editors, *Graph-Theoretic Concepts in Computer Science*, volume 314 of *Lecture Notes in Computer Science*, pages 106–120. Springer Berlin / Heidelberg, 1988. 10.1007/3-540-19422-3_9.

[Lacy *et al.*, 2005] Lee Lacy, Gabriel Aviles, Karen Fraser, William Gerber, Alice Mulvehill, and Robert Gaskill. Experiences Using OWL in Military Applications. In *Proceedings of the 2005 International Workshop on OWL: Experiences and Directions (OWLED 05)*, volume 188 of *CEUR WS Proceedings*, Galway, Ireland, November 11–12 2005.

[Lehmann, 1992] Fritz Lehmann. Semantic networks. *Computers Mathematics with Applications*, 23(2-5), 1992.

[Lutz *et al.*, 2006] Carsten Lutz, Franz Baader, Enrico Franconi, Domenico Lembo, Ralf Möller, Riccardo Rosati, Ulrike Sattler, Boontawee Suntisrivaraporn, and Sergio Tessaris. Reasoning Support for Ontology Design. In Bernardo Cuenca Grau, Pascal Hitzler, Conor Shankey, and Evan Wallace, editors, *OWLED*, volume 216 of *CEUR Workshop Proceedings*, Athens, Georgia, USA, November 10-11 2006. CEUR-WS.org.

[Lutz *et al.*, 2007] Carsten Lutz, Dirk Walther, and Frank Wolter. Conservative Extensions in Expressive Description Logics. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI 07)*, pages 453–458, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers.

[Möller *et al.*, 2008] Ralf Möller, Volker Haarslev, and Sebastian Wandelt. The Revival of Structural Subsumption in Tableau-based Reasoners. In Franz Baader, Carsten Lutz, and Boris Motik, editors, *Description Logics*, volume 353 of *CEUR Workshop Proceedings*, Dresden, Germany, May 13–16 2008. CEUR-WS.org.

[Motik *et al.*, 2007] Boris Motik, Rob Shearer, and Ian Horrocks. Optimized Reasoning in Description Logics using Hypertableaux. In Frank Pfenning, editor, *Proceedings of the 21st Conference on Automated Deduction (CADE-21)*, volume 4603 of *LNAI*, pages 67–83, Bremen, Germany, July 17–20 2007. Springer.

[Motik *et al.*, 2008] Boris Motik, Rob Shearer, and Ian Horrocks. Optimizing the Nominal Introduction Rule in (Hyper)Tableau Calculi. In Franz Baader, Carsten Lutz, and Boris Motik, editors, *Proceedings of the 21st International Workshop on Description Logics (DL 2008)*, volume 353 of *CEUR (http://ceur-ws.org/)*, Dresden, Germany, May 13–16 2008.

[Motik *et al.*, 2009] Boris Motik, Rob Shearer, and Ian Horrocks. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research*, 36:165–228, 2009.

[Motik, 2006] Boris Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univesität Karlsruhe, Germany, 2006.

[Nonnengart and Weidenbach, 2001] Andreas Nonnengart and Christoph Weidenbach. Computing Small Clause Normal Forms. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 6, pages 335–367. Elsevier Science, 2001.

[Parsia and Sirin, 2004] Bijan Parsia and Evren Sirin. Pellet: An OWL-DL Reasoner, November 7–11 2004.

[Patel-Schneider *et al.*, 2004] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL Web Ontology Language: Semantics and Abstract Syntax, W3C Recommendation, February 10 2004.
http://www.w3.org/TR/owl-semantics/.

[Plaisted and Greenbaum, 1986] David A. Plaisted and Steven Greenbaum. A Structure-Preserving Clause Form Translation. *Journal of Symbolic Logic and Computation*, 2(3):293–304, 1986.

[Rector and Rogers, 2006] Alan L. Rector and Jeremy Rogers. Ontological and Practical Issues in Using a Description Logic to Represent Medical Concept Systems: Experience from GALEN. In Pedro Barahona, Franois Bry, Enrico Franconi, Nicola

Henze, and Ulrike Sattler, editors, *Tutorial Lectures of the 2nd International Summer School 2006*, volume 4126 of *LNCS*, pages 197–231, Lisbon, Portugal, September 4–8 2006. Springer.

[Robinson, 1965] Alan Robinson. Automatic Deduction with Hyper-Resolution. *Int. Journal of Computer Mathematics*, 1:227–234, 1965.

[Ruttenberg *et al.*, 2005] Alan Ruttenberg, Jonathan Rees, and Joanne Luciano. Experience Using OWL DL for the Exchange of Biological Pathway Information. In *Proceedings of the 2005 International Workshop on OWL: Experiences and Directions (OWLED 05)*, volume 188 of *CEUR WS Proceedings*, Galway, Ireland, November 11–12 2005.

[Schmidt and Hustadt, 2003] Renate A. Schmidt and Ullrich Hustadt. A Principle for Incorporating Axioms into the First-Order Translation of Modal Formulae. In Franz Baader, editor, *Proceedings of the 19th International Conference on Automated Deduction (CADE-19)*, volume 2741 of *LNAI*, pages 412–426, Miami Beach, FL, USA, July 28–August 2 2003. Springer.

[Schmidt-Schauß and Smolka, 1991] Manfred Schmidt-Schauß and Gert Smolka. Attributive Concept Descriptions with Complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[Shearer and Horrocks, 2009] Rob Shearer and Ian Horrocks. Exploiting Partial Information in Taxonomy Construction. In Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, editors, *Proceedings of the 8th International Semantic Web Conference (ISWC 2009)*, volume 5823 of *LNCS*, pages 569–584, Chantilly, VA, USA, October 25–29 2009. Springer.

[Shearer *et al.*, 2008] Rob Shearer, Boris Motik, and Ian Horrocks. HermiT: A Highly-Efficient OWL Reasoner. In Alan Ruttenberg, Ulrile Sattler, and Cathy Dolbear, editors, *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2008 EU)*, volume 432 of *CEUR (http://ceur-ws.org/)*, Karlsruhe, Germany, October 26–27 2008.

[Shearer *et al.*, 2009] Rob Shearer, Ian Horrocks, and Boris Motik. Exploiting Partial Information in Taxonomy Construction. In Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler, editors, *Proceedings of the 22nd International Workshop on Description Logics (DL 2009)*, volume 477 of *CEUR (http://ceur-ws.org/)*, Oxford, UK, July 27–30 2009.

[Sidhu *et al.*, 2005] Armandeep S. Sidhu, Tharam S. Dillon, Elizabeth Chang, and Baldev S. Sidhu. Protein Ontology Development using OWL. In *Proceedings of the 2005 International Workshop on OWL: Experiences and Directions (OWLED 05)*, volume 188 of *CEUR WS Proceedings*, Galway, Ireland, November 11–12 2005.

[Sirin *et al.*, 2006] Evren Sirin, Bernardo Cuenca Grau, and Bijan Parsia. From Wine to Water: Optimizing Description Logic Reasoning for Nominals. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 90–99, Lake District, UK, June 2–5 2006. AAAI Press.

[Soergel *et al.*, 2004] Dagobert Soergel, Boris Lauser, Anita Liang, Frehiwot Fisseha, Johannes Keizer, and Stephen Katz. Reengineering Thesauri for New Applications: The AGROVOC Example. *Journal of Digital Information*, 4(4), 2004.

[Tobies, 2000] Stephan Tobies. The Complexity of Reasoning with Cardinality Restrictions and Nominals in Expressive Description Logics. *Journal of Artificial Intelligence Research*, 12:199–217, 2000.

[Tobies, 2001] Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, Germany, 2001.

[Tsarkov and Horrocks, 2004] Dmitry Tsarkov and Ian Horrocks. Efficient Reasoning with Range and Domain Constraints. In Volker Haarslev and Ralf Möller, editors, *Proceedings of the 2004 International Workshop on Description Logics (DL 2004)*, volume 104 of *CEUR Workshop Proceedings*, Whistler, BC, Canada, June 6–8 2004.

[Tsarkov and Horrocks, 2005a] Dmitry Tsarkov and Ian Horrocks. Optimised Classification for Taxonomic Knowledge Bases. In Ian Horrocks, Ulrike Sattler, and Frank Wolter, editors, *Proceedings of the 2005 International Workshop on Description Logics (DL 2005)*, Edinburgh, Scotland, UK, July 26–28 2005.

[Tsarkov and Horrocks, 2005b] Dmitry Tsarkov and Ian Horrocks. Ordering Heuristics for Description Logic Reasoning. In L. Pack Kaelbling and Alessandro Saffiotti, editors, *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 609–614, Edinburgh, UK, July 30–August 5 2005. Morgan Kaufmann Publishers.

[Tsarkov and Horrocks, 2006] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *LNAI*, pages 292–297, Seattle, WA, USA, August 17–20 2006. Springer.

[Tsarkov *et al.*, 2007] Dmitry Tsarkov, Ian Horrocks, and Peter F. Patel-Schneider. Optimising Terminological Reasoning for Expressive Description Logics. *Journal of Automated Reasoning*, 39:277–316, 2007.

[Wu and Haarslev, 2008] Jiewen Wu and Volker Haarslev. Planning of Axiom Absorption. In Franz Baader, Carsten Lutz, and Boris Motik, editors, *Proceedings of the 21st International Workshop on Description Logics (DL 2008)*, volume 353 of *CEUR Workshop Proceedings*, Dresden, Germany, May 13–16 2008.