Reasoning for Description Logics around $\mathcal{SHIQ}$ in a Resolution Framework

Ullrich Hustadt[1] Boris Motik[2]
Ulrike Sattler[3]

[1]Department of Computer Science, University of Liverpool
Liverpool, UK
U.Hustadt@csc.liv.ac.uk
[2]FZI Research Center for Information Technologies, University of Karlsruhe
Karlsruhe, Germany
motik@fzi.de
[3]Department of Computer Science, University of Manchester
Manchester, UK
sattler@cs.man.ac.uk

November 25, 2004

Abstract

We present several algorithms for reasoning with description logics closely related to $\mathcal{SHIQ}$. Firstly, we present an algorithm for deciding satisfiability of $\mathcal{SHIQ}$ knowledge bases. Then, to enable representing concrete data such as strings or integers, we devise a general approach for reasoning with concrete domains in the framework of resolution, and apply it to obtain a procedure for deciding $\mathcal{SHIQ}(\mathbf{D})$. For unary coding of numbers, this procedure is worst-case optimal, i.e. it runs in exponential time. Motivated by the prospects of reusing optimization techniques from deductive databases, such as magic sets, we devise an algorithm for reducing $\mathcal{SHIQ}(\mathbf{D})$ knowledge bases to disjunctive datalog programs. Furthermore, we show that so-called *DL-safe* rules can be combined with disjunctive programs obtained by our transformation to increase the expressivity of the logic, without affecting decidability. We show that our algorithms can easily be extended to handle answering conjunctive queries over $\mathcal{SHIQ}(\mathbf{D})$ knowledge bases. Finally, we extend our algorithms to support metamodeling. Since $\mathcal{SHIQ}(\mathbf{D})$ is closely related to OWL-DL, our algorithms provide alternative mechanisms for reasoning in the Semantic Web.

Contents

Chapter 1

Introduction

Description Logics (DLs) are a family of knowledge representation formalisms which allow representing and reasoning with domain knowledge in a formally well-understood way. The first description logic KL-ONE [23] was introduced to address the deficiencies of semantic networks [96] and frame-based knowledge representation systems [79]. In particular, it was based on a formal model-theoretic semantics, to compensate for the vague and imprecise semantics of earlier systems.

A description logic *terminology* consist of *concepts*, i.e. unary predicates representing sets of individuals, and *roles*, i.e. binary predicates representing links between individuals. Concepts can be *atomic*, meaning that they are denoted by name, or *complex*, consisting of concept constructors representing necessary and/or sufficient membership conditions. Apart from a terminology, a description logic *knowledge base* usually has an *assertional* component, which specifies the membership of individuals or pairs of individuals in concepts and roles, respectively. Historically, the fundamental reasoning problem in description logic is to determine whether one concept *subsumes* another, i.e. whether the extension of the former necessarily includes the extension of the latter concept. Apart from subsumption, reasoning problems, such as checking satisfiability of a knowledge base or retrieving individuals belonging to a concept, are also of interest in applications.

After its introduction, it soon turned out that the subsumption problem in KL-ONE is undecidable [106]. From this point on, a large body of description logic research focused on investigating the fundamental tradeoffs between expressivity and computational complexity. This line of research culminated in a detailed taxonomy of complexity/undecidability results for various description logic variants [5, Chapter 5].

In parallel, an important goal of description logic research was to develop practical reasoning algorithms and to realize them in practical knowledge representation and reasoning systems. Initially, reasoning algorithms were based on *structural subsumption*. Roughly speaking, such algorithms transform a concept description to a certain normal form. Subsumption between two concepts is checked by comparing the structure of corresponding normal forms. After initial experiments with systems such as

CLASSIC [20] and LOOM [77], it became evident that sound and complete structural subsumption algorithms are possible only for inexpressive logics.

As a reaction to these deficiencies, tableau algorithms were proposed as an alternative for description logic reasoning in [107]. A tableau algorithm demonstrates satisfiability of a knowledge base by trying to build a model of a knowledge base[1]. If such a model can be built, the knowledge base is evidently satisfiable, and if it cannot, the knowledge base is unsatisfiable. Classical reasoning problems can be reduced to satisfiability checking, and, since tableau algorithms for very expressive logics are available, they are nowadays considered the state-of-the-art. Most contemporary implemented DL systems are based on a tableau calculus.

$\mathcal{SHIQ}$ [61] is a very expressive description logic which, apart from the usual boolean operations on concepts and existential and universal quantification on roles, supports advanced features, such as inverse and transitive roles, role hierarchies and number restrictions. It can be extended with a *concrete domain* [6] to facilitate representing and reasoning with concrete datatypes, such as strings or integers; this extension is denoted as $\mathcal{SHIQ}(\mathbf{D})$. A tableau algorithm for $\mathcal{SHIQ}$ was presented in [61, 62], and it can be easily extended with datatype reasoning in the same way as this was done for the related logic $\mathcal{SHOQ}(\mathbf{D})$ in [59]. $\mathcal{SHIQ}(\mathbf{D})$ is important, since it provides the basis of OWL-DL [91] — a W3C recommendation language for ontology representation in the Semantic Web. Namely, OWL-DL is a notational variant of the $\mathcal{SHOIN}(\mathbf{D})$ description logic, which differs from $\mathcal{SHIQ}(\mathbf{D})$ mainly by supporting *nominals* — concepts consisting exactly of the specified set of individuals.

In [114] it was shown that reasoning in $\mathcal{SHIQ}$ is ExpTime-complete. Moreover, tableau algorithms for $\mathcal{SHIQ}$ run in worst-case doubly non-deterministic exponential time. Because of high worst-case complexity, effective optimization techniques are essential to make reasoning usable in practice. Numerous optimization techniques were presented in [56], along with the evidence for their benefits in reasoning with large knowledge bases. Based on these techniques, the FaCT [57] reasoner was implemented and successfully applied to interesting practical problems. Another state-of-the-art reasoner for $\mathcal{SHIQ}(\mathbf{D})$ is Racer [50], distinguished from FaCT mainly by supporting assertional knowledge.

Description logics were successfully applied in numerous fields, such as information integration [5, Chapter 16] [73, 18, 45], software engineering [5, Chapter 11] and conceptual modeling [5, Chapter 10] [27]. The performance of reasoning algorithms was found to be quite adequate for applications mainly requiring terminological reasoning. However, new applications, such as metadata management in the Semantic Web, require efficient query answering over large ABoxes. So far, attempts have been made to answer queries by reduction to ABox consistency checking, which can then be performed using tableau algorithms. From a theoretical point of view, this approach is quite elegant, but from a practical point of view, it has a significant drawback: as the number of ABox individuals increases, the performance becomes quite poor.

[1] Actually, for some logics tableaux algorithms build finite abstractions of possibly infinite models.

We believe that there are three main reasons why tableau algorithms scale poorly to ABox reasoning. Firstly, tableau algorithms treat all individuals separately: to answer a query, a tableau check is needed for each individual to see whether it is an answer to the query or not. Secondly, usually only a small subset of ABox information is relevant to the query. Thirdly, a tableau-based ABox algorithm tries to construct a "forest model", i.e. a model where each individual in the ABox is the root of a tree of (in the worst case) exponential depth. Clearly, an ABox algorithm that is to handle large numbers of ABox individuals has to restrict these trees to a minimum size. These deficiencies have already been acknowledged by the research community, and certain optimization techniques for instance retrieval have already been developed [51, 52]. However, the performance of query answering is still not satisfactory in practice.

In parallel to description logic research, many techniques for efficient reasoning in deductive databases already exist. For example, the first deficiency outlined above is addressed by managing individuals in sets [2]. This opens the door to various optimization techniques, such as the join order optimization. Consider the query $worksAt(P, I), hasName(I, \text{'FZI'})$. Since the second argument is a constant, it is reasonable to assume that evaluating the subgoal $hasName(I, \text{'FZI'})$ first, and then joining the result with tuples in the *worksAt* relation, will reduce the number of irrelevant tuples considered. Join order optimizations are usually based on database statistics, and have proven to be very effective in practice.

The second deficiency outline above can be addressed by effectively identifying the subset of the ABox which is relevant to the query and then to run the reasoning algorithm only on this subset. Magic sets transformation [17] is a primary technique addressing this problem, and has been used mainly in the context of Horn deductive databases to optimize evaluation of recursive queries. Roughly speaking, the query is modified so that during its evaluation, a set of relevant facts is derived. Evaluating the original query is then limited to this set of facts. The magic sets transformation for disjunctive programs has been presented recently in [48], along with empirical evidence of its usefulness. The significant performance improvements reported there are attributed mainly to the fact that selecting information relevant to the query reduces the number of models of the disjunctive program.

Since techniques for reasoning in deductive databases are now mature, we think it makes sense to investigate whether they can be used to improve ABox reasoning in description logics. Our initial experimental results were reported in [80] and are very promising, exhibiting improvements in performance of one, or sometimes two orders of magnitude. However, the approach presented there was limited to a very simple logic. In particular, the presence of existential quantifiers in terminological cycles was not allowed, due to problems with termination. Other expressive features, such as number restrictions or transitivity, were not considered either.

In this paper, we develop several novel algorithms for reasoning in description logics around $\mathcal{SHIQ}$. Our ultimate goal is to enable reducing $\mathcal{SHIQ}(\mathbf{D})$ knowledge bases to disjunctive datalog programs, while preserving the set of entailed ground facts. Hence, we reduce ABox reasoning in description logic to query answering in disjunctive

datalog, and thus enable exploiting previously mentioned optimization techniques for DL query answering. However, many algorithms on the path to the reduction are interesting in their own right. We give an outline of this paper and its contribution:

- In Chapter 3 we present a decision procedure for checking satisfiability of $\mathcal{SHIQ}$ knowledge bases based on basic superposition [15, 83]. Basic superposition is a clausal refutation calculus optimized for theorem proving with equality. Parameterized by a suitable term ordering and a selection function, basic superposition yields a decision procedure only for a slightly weaker logic $\mathcal{SHIQ}^-$, in which number restrictions are allowed only on roles not having subroles; for full $\mathcal{SHIQ}$, saturation by basic superposition does not terminate. To remedy that, we introduce so-called *decomposition* rule, which transforms certain clauses into simpler ones, thus ensuring termination. We show that decomposition is a very general rule, and can be used with any calculus compatible with the standard notion of redundancy [14]. This decision procedure runs in worst-case exponential time, provided that numbers are coded in unary, so it is worst-case optimal. Please note that the assumption on unary coding of numbers is standard in description logic algorithms, and, to the best of our knowledge, is employed in all existing practical reasoning systems.

- Until now, reasoning with concrete domains has been predominantly studied in the context of tableau algorithms. Since our algorithms operate in the framework of resolution, existing approaches are not directly applicable to our setting. Therefore, in Chapter 4 we present a general approach for reasoning with a concrete domain in the framework of resolution. Our approach is applicable to any calculus whose completeness proof is based on the model generation method [14], so it can easily be combined with basic superposition. We apply this approach to the decision procedure from Chapter 3 and thus obtain a decision algorithm for $\mathcal{SHIQ}(\mathbf{D})$. We show that, assuming a bound on the arity of concrete predicates, adding concrete domains does not increase the complexity of reasoning.

- In Chapter 5 we present an algorithm for reducing $\mathcal{SHIQ}(\mathbf{D})$ knowledge bases to disjunctive datalog programs. This algorithm produces a disjunctive datalog program capable of simulating inference steps of the algorithms presented in Chapter 3 and Chapter 4. Roughly speaking, algorithms from Chapter 3 and Chapter 4 are first used to compute all non-ground consequences of a knowledge base. Thus obtained set of clauses is then transformed in a way to enable simulating remaining ground inferences in disjunctive datalog.

- Based on the algorithm from Chapter 5, in Chapter 6 we analyze the data complexity of reasoning in $\mathcal{SHIQ}(\mathbf{D})$. Data complexity is measured only in the size of the ABox, while assuming that TBox and RBox are fixed in size. In applications where the size of the ABox (i.e. the number of facts) is much bigger than the size of the TBox and RBox (i.e. the size of the schema), data complexity

provides a better estimate of the practical applicability of an algorithm. Surprisingly, the data complexity of satisfiability checking in $\mathcal{SHIQ}(\mathbf{D})$ turns out to be NP-complete, which is much better than the combined complexity, which is ExpTime-complete. Moreover, we identify the Horn-$\mathcal{SHIQ}(\mathbf{D})$ fragment of $\mathcal{SHIQ}(\mathbf{D})$ which does not provide for disjunctive reasoning, but exhibits polynomial data complexity.

- In Chapter 7 we consider a hybrid system consisting of $\mathcal{SHIQ}(\mathbf{D})$ extended with DL-safe rules. The integration of rules and description logics is achieved by allowing concepts and roles to occur as unary and binary predicates, respectively, in the atoms of the rule head or body. To achieve decidability, the DL-safety restriction is imposed, requiring each variable in the rule to occur in an atom in the rule body whose predicate is neither a concept nor a role. Intuitively, the DL-safety requirement makes query answering decidable, since it ensures that rules are applicable only to individuals explicitly occurring in the knowledge base. For query answering, DL-safe rules may simply be appended to the disjunctive datalog program obtained by the reduction from Chapter 5.

- In Chapter 8 we extend our algorithms to handle answering and checking subsumption of conjunctive queries over $\mathcal{SHIQ}(\mathbf{D})$ knowledge bases. Conjunctive queries provide a formal foundation for the vast majority of database queries commonly used in practice [29], so they lend themselves naturally as an expressive query language for description logics.

- In Chapter 9 we consider extending the logic to allow for treating concepts as individuals and vice versa, and thus to provide for metamodeling. We show that, if metamodeling is given semantics as specified in the Semantic Web standard OWL-Full [91], the logic becomes undecidable. Therefore, we propose an alternative semantics for metamodeling based on HiLog [30] — a logic which aims to simulate second-order reasoning in a first-order framework. We show that under such semantics, our algorithms can be easily extended to provide a decision procedure.

Certain parts of this work were published previously. In particular, the resolution decision procedure and the reduction to disjunctive datalog for $\mathcal{SHIQ}^-$ were published in [66], algorithms for reasoning with a concrete domain were published in [65], and reasoning with DL-safe rules was published in [81].

Chapter 2

Preliminaries

2.1 Multi-sorted First-order Logic

We assume standard definitions of first-order logic ([42] is a good text book), which we extend with multi-sorted signatures as usual. Let $\Sigma = (\mathcal{P}, \mathcal{F}, \mathcal{V}, \mathcal{S})$ be a first-order signature, where $\mathcal{P}$ is a finite set of predicate symbols, $\mathcal{F}$ a finite set of function symbols, $\mathcal{V}$ a countable set of variables and $\mathcal{S}$ a finite set of *sorts*. Each n-ary function symbol $f \in \mathcal{F}$ is associated with a sort signature $r_1 \times \ldots \times r_n \to r$, and each n-ary predicate symbol $P \in \mathcal{P}$ is associated with a sort signature $r_1 \times \ldots \times r_n$, where $r_{(i)} \in \mathcal{S}$. The sort of each variable is determined by the function $\mathsf{sort} : \mathcal{V} \to \mathcal{S}$.

We now define the set of *terms* $\mathcal{T}(\Sigma)$, and extend the function sort to terms, as follows: the set $\mathcal{T}(\Sigma)$ is the smallest set such that (i) $\mathcal{V} \subseteq \mathcal{T}(\Sigma)$, and (ii) if $f \in \mathcal{F}$ has the signature $r_1 \times \ldots \times r_n \to r$, and $t_i \in \mathcal{T}(\Sigma)$ with $\mathsf{sort}(t_i) = r_i$ for $i \leq n$, then $t = f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma)$ with $\mathsf{sort}(t) = r$. The set of *atoms* $\mathcal{A}(\Sigma)$ is the smallest set such that, if $P \in \mathcal{R}$ has the signature $r_1 \times \ldots \times r_n$, and $t_i \in \mathcal{T}(\Sigma)$ with $\mathsf{sort}(t_i) = r_i$ for $i \leq n$, then $P(t_1, \ldots, t_n) \in \mathcal{A}(\Sigma)$. Terms (atoms) not containing variables are called ground terms (atoms).

A *position* p is a finite sequence of integers. The empty position is denoted with ϵ. If a position p_1 is a proper prefix of a position p_2, then and p_1 is *above* p_2, and p_2 is *below* p_1. A *subterm* of t at position p, denoted with $t|_p$, is defined inductively as $t|_\epsilon = t$ and, if $t = f(t_1, \ldots, t_n)$, then $t|_{ip} = t_i|_p$. A *replacement* of a subterm of t at position p with s, denoted with $t[s]_p$, is defined inductively as $t[s]_\epsilon = s$ and, if $t = f(t_1, \ldots, t_n)$, then $t[s]_{ip} = f(t_1, \ldots, t_i[s]_p, \ldots, t_n)$.

The set of formulae $\mathcal{L}(\Sigma)$ defined over the signature Σ is the smallest set such that $\top$ and $\bot$ are in $\mathcal{L}(\Sigma)$, $\mathcal{A}(\Sigma) \subseteq \mathcal{L}(\Sigma)$ and, if $\varphi, \varphi_1, \varphi_2 \in \mathcal{L}(\Sigma)$ and $x \in \mathcal{V}$, then $\neg\varphi$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\exists x : \varphi$ and $\forall x : \varphi$ are in $\mathcal{L}(\Sigma)$. As usual, $\varphi_1 \to \varphi_2$ is an abbreviation for $\neg\varphi_1 \vee \varphi_2$, $\varphi_1 \leftarrow \varphi_2$ is equivalent to $\varphi_2 \to \varphi_1$, and $\varphi_1 \leftrightarrow \varphi_2$ is an abbreviation for $(\varphi_1 \to \varphi_2) \wedge (\varphi_1 \leftarrow \varphi_2)$. A variable x in a formula φ is *free* it it does not occur under the scope of a quantifier. If φ does not have free variables, it is *closed*.

The notion of a *subformula* of φ at position p, denoted with $\varphi|_p$, is defined inductively as $\varphi|_\epsilon = \varphi$; $(\varphi_1 \circ \varphi_2)|_{ip} = \varphi_i$ for $\circ \in \{\wedge, \vee, \leftarrow, \to, \leftrightarrow\}$ and $i \in \{1, 2\}$; $\varphi|_{1p} = \psi$

for $\varphi = \neg\psi$, $\varphi = \forall x : \psi$ or $\varphi = \exists x : \psi$. A replacement of a subformula of φ at position p with ψ is denoted as $\varphi[\psi]_p$ and is defined in the obvious way.

The *polarity* of the subformula $\varphi|_p$ at position p in a formula φ, written $\mathsf{pol}(\varphi, p)$, is defined as follows: $\mathsf{pol}(\varphi, \epsilon) = 1$; $\mathsf{pol}(\neg\varphi, 1p) = -\mathsf{pol}(\varphi, p)$; $\mathsf{pol}(\varphi_1 \circ \varphi_2, ip) = \mathsf{pol}(\varphi_i, p)$ for $\circ \in \{\wedge, \vee\}$ and $i \in \{1, 2\}$; $\mathsf{pol}(\exists x : \varphi, 1p) = \mathsf{pol}(\varphi, p)$; $\mathsf{pol}(\forall x : \varphi, 1p) = \mathsf{pol}(\varphi, p)$; $\mathsf{pol}(\varphi_1 \to \varphi_2, 1p) = -\mathsf{pol}(\varphi_1, p)$; $\mathsf{pol}(\varphi_1 \to \varphi_2, 2p) = \mathsf{pol}(\varphi_2, p)$; $\mathsf{pol}(\varphi_1 \leftrightarrow \varphi_2, ip) = 0$ for $i \in \{1, 2\}$.

A substitution σ is a function from $\mathcal{V}$ into $\mathcal{T}(\Sigma)$ for which $\sigma(x) \neq x$ only for a finite number of variables x, and, if $\sigma(x) = t$, then $\mathsf{sort}(x) = \mathsf{sort}(t)$. We often write a substitution σ as a finite set of mappings $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. The result of *applying* a substitution σ to a term t is denoted with $t\sigma$, and is defined recursively as follows: $x\sigma = \sigma(x)$ and $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$. For a substitution σ and a variable x, the substitution σ_x is defined as follows:

$$y\sigma_x = \begin{cases} y\sigma & \text{if } y \neq x \\ y & \text{if } y = x \end{cases}$$

Now the application of a substitution σ to a formula φ, written $\varphi\sigma$, is defined as follows: $A(t_1, \ldots, t_n)\sigma = A(t_1\sigma, \ldots, t_n\sigma)$; $(\varphi_1 \circ \varphi_2)\sigma = \varphi_1\sigma \circ \varphi_2\sigma$ for $\circ = \{\wedge, \vee, \leftarrow, \to, \leftrightarrow\}$; $(\neg\varphi)\sigma = \neg(\varphi\sigma)$; $(\forall x : \varphi)\sigma = \forall x : (\varphi\sigma_x)$; and $(\exists x : \varphi)\sigma = \exists x : (\varphi\sigma_x)$.

A composition of substitutions τ and σ, written $\sigma\tau$, is defined as $x\sigma\tau = (x\sigma)\tau$. The *empty* substitution is denoted as $\{\}$. A substitution σ is called a *variable renaming* if it contains only mappings of the form $x \mapsto y$. A substitution σ is equivalent to θ up to variable renaming if there is a variable renaming η such that $\theta = \sigma\eta$; in such a case, θ is also equivalent to σ up to variable renaming [8]. A substitution σ is *more general* than a substitution θ if there is a substitution η such that $\theta = \sigma\eta$.

A substitution σ is a *unifier* of terms s and t if $s\sigma = t\sigma$. A unifier σ of s and t is called a *most general unifier* if, for any unifier θ of s and t, σ is either more general than θ, or it is equivalent to θ up to variable renaming. The notion of unifiers is extended to atoms in the obvious way. If a most general unifier σ of s and t exists, it is unique up to variable renaming [8], so we write $\sigma = \mathsf{MGU}(s, t)$.

The semantics of multi-sorted first-order logic is defined as follows. An *interpretation* is a pair $I = (\mathcal{D}, \cdot^I)$, where $\mathcal{D}$ is a function assigning to each sort $s \in \mathcal{S}$ an interpretation domain $\mathcal{D}_s$ such that, if $r_i, r_j \in \mathcal{S}$ and $r_i \neq r_j$, then $\mathcal{D}_{r_i} \cap \mathcal{D}_{r_j} = \emptyset$, and $\cdot^I$ is a function assigning to each predicate symbol A with a signature $r_1 \times \ldots \times r_n$ an interpretation relation $A^I \subseteq \mathcal{D}_{r_1} \times \ldots \times \mathcal{D}_{r_n}$, and to each function symbol f with a signature $r_1 \times \ldots \times r_n \to r$ an interpretation function $f^I : \mathcal{D}_{r_1} \times \ldots \times \mathcal{D}_{r_n} \to \mathcal{D}_r$. A *variable assignment* is a function B assigning to each variable $x \in \mathcal{V}$ a value from $\mathcal{D}_{\mathsf{sort}(x)}$. An x-variant of B, written B_x, is a variable assignment which assigns the same values as B to all variables, except possibly to the variable x. The value of a term $t \in \mathcal{T}(\Sigma)$ under I and B, written $t^{I,B}$, is defined as follows: if $t = x$, then $t^{I,B} = B(x)$, and if $t = f(t_1, \ldots, t_n)$, then $t^{I,B} = f^I(t_1^{I,B}, \ldots, t_n^{I,B})$. The value of a formula φ under I and B, written $\varphi^{I,B}$, is defined as follows: $[\top]^{I,B} = t$, $[\bot]^{I,B} = f$, $[A(t_1, \ldots, t_n)]^{I,B} = t$ if and only if $(t_1^{I,B}, \ldots, t_n^{I,B}) \in A^I$; $[\neg\varphi]^{I,B} = \neg\varphi^{I,B}$; $[\varphi_1 \circ \varphi_2]^{I,B} = \varphi_1^{I,B} \circ \varphi_2^{I,B}$ for

$\circ \in \{\wedge, \vee\}$; $[\exists x : \varphi]^{I,B} = t$ if and only if $\varphi^{I,B_x} = t$ for some B_x; and $[\forall x : \varphi]^{I,B} = t$ if and only if $\varphi^{I,B_x} = t$ for all B_x. If φ is closed, then $\varphi^{I,B}$ does not depend on the choice of B, so we write φ^I instead. For a closed formula φ, an interpretation I is a *model* of φ, written $I \models \varphi$, if $\varphi^I = t$. A closed formula φ is *valid*, written $\models \varphi$, if $I \models \varphi$ for all interpretations I; φ is *satisfiable* if $I \models \varphi$ for at least one interpretation I; and φ is *unsatisfiable* if an interpretation I, such that $I \models \varphi$, does not exist. A closed formula φ_1 *entails* a formula φ_2, written $\varphi_1 \models \varphi_2$, if $I \models \varphi_2$ for each interpretation I for which $I \models \varphi_1$. It is well-known that $\varphi_1 \models \varphi_2$ if and only if $\varphi_1 \wedge \neg\varphi_2$ is unsatisfiable. Formulae φ_1 and φ_2 are *equisatisfiable* if φ_1 is satisfiable if and only if φ_2 is satisfiable.

Let φ be a closed first-order formula, and Λ a set of positions in φ. Then $\mathsf{Def}_\Lambda(\varphi)$ is the *definitional normal form* of φ with respect to Λ and is defined inductively as follows, where p is maximal in $\Lambda \cup \{p\}$ with respect to the prefix ordering on positions, Q is a new predicate not occurring in φ, $x_1, \ldots, x_n$ are the free variables of $\varphi|_p$, and $\circ$ is $\rightarrow$ if $\mathsf{pol}(\varphi, p) = 1$, $\leftarrow$ if $\mathsf{pol}(\varphi, p) = -1$ and $\leftrightarrow$ if $\mathsf{pol}(\varphi, p) = 0$:

$$\begin{aligned} \mathsf{Def}_\emptyset(\varphi) &= \varphi \\ \mathsf{Def}_{\Lambda\cup\{p\}}(\varphi) &= \mathsf{Def}_\Lambda(\varphi[Q(x_1, \ldots, x_n)]_p) \wedge \forall x_1, \ldots, x_n : Q(x_1, \ldots, x_n) \circ \varphi|_p \end{aligned}$$

It is well-known [93, 9, 86] that, for any Λ, φ and $\mathsf{Def}_\Lambda(\varphi)$ are equisatisfiable, and that $\mathsf{Def}_\Lambda(\varphi)$ can be computed in polynomial time.

Let φ be a formula and p a position in φ, such that either $\mathsf{pol}(\varphi, p) = 1$ and $\varphi|_p = \exists x : \psi$, or $\mathsf{pol}(\varphi, p) = -1$ and $\varphi|_p = \forall x : \psi$, where $x, x_1, \ldots, x_n$ are exactly the free variables of ψ. Then $\varphi[\psi\{x \mapsto f(x_1, \ldots, x_n)\}]_p$, where f is a new function symbol not occurring in φ, is a formula obtained by *skolemization* of φ at position p. The formula $\mathsf{sk}(\varphi)$ is obtained from φ by skolemization at all positions where skolemization is possible. Usually, we assume that $\mathsf{sk}(\varphi)$ is computed by *outer skolemization*, where a position p is skolemized before any position below p. It is well-known that φ and $\mathsf{sk}(\varphi)$ are equisatisfiable [42].

The subset of ground terms of $\mathcal{F}(\Sigma)$ is called the *Herbrand universe* HU of Σ. A Herbrand interpretation is an interpretation I where (i) $\mathcal{D}_r \subseteq \mathcal{F}(\Sigma)$ such that for each $t \in \mathcal{F}(\Sigma)$, we have $t \in \mathcal{D}_r$ if and only if $\mathsf{sort}(t) = r$, and (ii) function symbols are interpreted by themselves, i.e. for each $f \in \mathcal{F}$ and $t_i \in HU$, we have $f^I(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$. The Herbrand base HB of Σ is the set of all ground atoms built over the Herbrand universe of Σ. A Herbrand interpretation I can equivalently be considered a subset of the Herbrand base, i.e. $I \subseteq HB$. A formula φ is satisfiable if and only if $\mathsf{sk}(\varphi)$ is satisfiable in a Herbrand interpretation [42] (this holds for multi-sorted logic as well).

A *multiset* M over a set N is a function $M : N \rightarrow \mathbb{N}_0$, where $\mathbb{N}_0$ is the set of all non-negative integers. A multiset is finite if $M(x) \neq 0$ for a finite number of x; in the rest, we consider only finite multisets. For two multisets M_1 and M_2, $M_1 \subseteq M_2$ if $M_1(x) \leq M_2(x)$ for each $x \in N$, and $M_1 = M_2$ if $M_1 \subseteq M_2$ and $M_2 \subseteq M_1$. Furthermore, the union of M_1 and M_2 is defined as $(M_1 \cup M_2)(x) = M_1(x) + M_2(x)$, and the intersection is defined as $(M_1 \cap M_2)(x) = \min(M_1(x), M_2(x))$. A multiset M is empty, written $M = \emptyset$, if $M(x) = 0$ for all $x \in N$.

A *literal* is an atom A or a negated atom $\neg A$. A *clause* is a multiset of literals and is usually written as $C = L_1 \vee \ldots \vee L_n$. A clause C is semantically equivalent to $\forall \mathbf{x} : C$, where $\mathbf{x}$ is the vector of free variables of C. Satisfiability of clauses is usually considered in a Herbrand interpretation I, as follows: for a ground clause C^g, $I \models C^g$ if a literal $A_i \in C^g$ exists, such that $A_i \in I$, or else a literal $\neg A_j \in C^g$ exists, such that $A_j \notin I$; and for a non-ground clause C, $I \models C$ if and only if $I \models C^g$ for each ground instance C^g of C. For a first-order formula φ, $\mathsf{Cls}(\varphi)$ is the set of clauses obtained by *clausification* of φ, i.e. by transforming $\mathsf{sk}(\varphi)$ into conjunctive normal form by exhaustive application of de Morgan's laws. It is well-known that φ is satisfiable if and only if $\mathsf{Cls}(\varphi)$ is satisfiable in a Herbrand model. A clause C is *safe* if each variable x occurring in any literal of C occurs also in a negative literal of C.

Unless otherwise noted, we denote atoms by letters A and B, clauses by C and D, literals by L, predicates by P, R, S, T and U, constants by a, b and c, variables by x, y and z, and terms by s, t, u, v and w.

2.2 Relations and Orderings

For a set of objects D, a (binary) relation R on D is a subset of $D \times D$. R is *total* if, for any two $x, y \in D$, either $R(x, y)$ or $R(y, x)$ or $x = y$. An *inverse* relation of R is defined as $R^- = \{(y, x) \mid (x, y) \in R\}$. A relation R is (i) *reflexive* if $x \in D$ implies $(x, x) \in R$; (ii) *symmetric* if $R^- \subseteq R$; (iii) *asymmetric* if $(x, y) \in R$ implies $(y, x) \notin R$; (iv) *antisymmetric* if $(x, y) \in R$ and $(y, x) \in R$ implies $x = y$; and (v) *transitive* if $(x, y) \in R$ and $(y, z) \in R$ implies $(x, z) \in R$. A $\circ$-closure, written R°, where $\circ$ can be a combination of reflexive, symmetric or transitive, is the smallest relation on D fulfilling the property $\circ$ such that $R \subseteq R^\circ$. The transitive closure of R is usually written as R^+, and the reflexive-transitive closure of R is usually written as R^*.

A relation R is *well-founded* if there is no infinite sequence $(\alpha_0, \alpha_1), (\alpha_1, \alpha_2), \ldots$ of tuples in R. An object $\alpha \in D$ is in *normal form* w.r.t. R if R does not contain an element (α, β), for any β. An object β is a normal form of α w.r.t. R if β is in normal form w.r.t. R and $(\alpha, \beta) \in R^*$. For a general relation R, an object can have any number of normal forms.

A *partial ordering* $\succeq$ over D is a reflexive, antisymmetric and transitive relation on D. A *strict ordering* $\succ$ over D is an irreflexive and transitive relation on D. A *multiset extension* of a strict ordering $\succ$ to multisets on D, written $\succ_{mul}$, is defined as follows: $M \succ_{mul} N$ if $M \neq N$ and if $N(x) > M(x)$ for some x, then there is some $y \succ x$ for which $M(y) > N(y)$. It is well-known that for finite multisets, if $\succ$ is total, then $\succ_{mul}$ is total as well.

A *term ordering* is an ordering where $D = \mathcal{T}(\Sigma)$, for some multi-sorted first-order signature Σ. A term ordering $\succ$ is *stable under substitutions* if for all terms s and t and all substitutions sigma σ, $s \succ t$ implies $s\sigma \succ t\sigma$; it is *stable under contexts* if for all terms s, t and u and all positions p, $s \succ t$ implies $u[s]_p \succ u[t]_p$; it satisfies the *subterm property* if $u[s]_p \succ s$ for all terms u and s and all positions $p \neq \epsilon$. A *rewrite ordering* if an ordering stable under contexts and stable under substitutions. A *reduction ordering*

is a well-founded rewrite ordering, and a *simplification ordering* is a reduction ordering with a subterm property.

The *lexicographic path ordering* (LPO) [34, 7] is a term ordering induced over a precedence on function symbols $>_P$. Each LPO has the subterm property and, if $>_P$ is total, then LPO is total on ground terms. It is defined as follows: $s \succ_{lpo} t$ if

1. t is a variable occurring as a proper subterm of s or
2. $s = f(s_1, \ldots, s_m)$, $t = g(t_1, \ldots, t_n)$, and at least one of the following holds:
   (a) $f >_P g$ and, for all i with $1 \leq i \leq n$, we have $s \succ_{lpo} t_i$ or
   (b) $f = g$ and, for some j, we have $(s_1, \ldots, s_{j-1}) = (t_1, \ldots t_{j-1})$, $s_j \succ_{lpo} t_j$, and $s \succ_{lpo} t_k$ for all k with $j < k \leq n$ or
   (c) $s_j \succeq_{lpo} t$ for some j with $1 \leq j \leq m$.

2.3 Rewrite Systems

A good text-book introduction to rewrite systems is [7]; an overview of the theory of rewrite systems can be found in [34]. A *rewrite system* R is a set of rewrite rules $s \Rightarrow t$ where s and t are terms. A *rewrite relation* induced by R, written as $\Rightarrow_R$, is the smallest relation such that $s \Rightarrow t \in R$ implies $u[s\sigma]_p \Rightarrow_R u[t\sigma]_p$, for all terms s, t and u, all substitutions σ and all positions p. For two terms s and t, we write $s \Downarrow_R t$ if there is a term u such that $s \Rightarrow_R^* u$ and $t \Rightarrow_R^* u$. If $\Downarrow_R$ and symmetric-reflexive-transitive closure of $\Rightarrow_R$ coincide, then R is *confluent*. For a confluent well-founded rewrite system R, each element α has a unique normal form w.r.t. $\Rightarrow_R$, which is denoted as $\mathsf{nf}_R(\alpha)$.

For a confluent well-founded rewrite system R consisting of ground rewrite rules only, R^* is the smallest set of ground equalities $s \approx t$ such that, for all ground terms s and t, if $\mathsf{nf}_R(s) = \mathsf{nf}_R(t)$, then $s \approx t \in R^*$.

2.4 Basic Superposition Calculus

Paramodulation is one of the fundamental techniques for theorem proving with equality. In order to improve its performance, in [10] the superposition calculus has been presented, where stronger ordering restrictions restrict unnecessary inferences. However, further optimizations of paramodulation and superposition were presented in [15], by adding a so called "basicness" restriction. These optimizations are very general, but a simplified version, called basic superposition, may be found in [11, 13]. A very similar calculus, based on an inference model with constrained clauses, was presented in [83].

The idea of the basic superposition calculus is to render superposition inferences into terms introduced by previous unification steps redundant. In practice, this technique has been shown essential for solving some particularly difficult problems in first-

order logic with equality [78]. Furthermore, basic superposition shows that superposition into arguments of Skolem function symbols is not necessary for completeness. Namely, any Skolem function symbol f occurs in the original clause set with variable arguments. Hence, for any term $f(t)$, if t is not a variable, it was introduced by a previous unification step.

It is a common practice in equational theorem proving to consider logical theories consisting of the equality predicate exclusively. This simplifies the theoretical treatment without losing generality. Literals $P(t_1, \ldots, t_n)$, where P is not the equality predicate, are encoded as $P(t_1, \ldots, t_n) \approx \top$, so predicate symbols actually become function symbols. It is well-known that this transformation preserves satisfiability. To avoid considering terms where predicate symbols occur as proper subterms, one usually employs a multi-sorted framework: all predicate symbols and the symbol $\top$ are of a sort different from the sort of function symbols and variables. In the rest, $P(t_1, \ldots, t_n)$ is considered a syntactic shortcut for $P(t_1, \ldots, t_n) \approx \top$. When ambiguities do not arise, we still call P a predicate symbol, and not a function symbol (although it technically is a function symbol due to the encoding). Finally, the predicate $\approx$ has built-in symmetry: a literal $s \approx t$ may also be interpreted as $t \approx s$.

The inference rules are formulated by breaking a clause into two parts: (i) the *skeleton* clause C and (ii) the substitution σ representing the cumulative effects of previous unifications. These two components together are called a *closure*, written as $C \cdot \sigma$, which is logically equivalent to a clause $C\sigma$. A closure $C \cdot \sigma$ can, for convenience, equivalently be represented as $C\sigma$, where the terms occurring at variable positions of C are *marked*[1] by $[\,]$. Any position at or below a marked position is called a *substitution position*. Basic superposition can be summarized as a calculus where superposition into a substitution position is not necessary.

The following closure is logically equivalent to the clause $P(f(y)) \vee g(b) \approx b$. On the left-hand side the closure is represented by a skeleton and a substitution explicitly, whereas on the right-hand side it is represented by marking the positions of variables in the skeleton.

$$(P(x) \vee z \approx b) \cdot \{x \mapsto f(y), z \mapsto g(b)\} \equiv P([f(y)]) \vee [g(b)] \approx b$$

Note that all variable positions are always marked, so we usually do not show this for readability purposes. A closure $C \cdot \sigma$ is *ground* if $C\sigma$ is ground. To technically simplify the presentation, we consider each closure to be in the *standard form*, which is the case if the following conditions are satisfied:

- the substitution σ does not contain trivial mappings of the form $x \mapsto y$ and
- all variables from $\mathsf{dom}(\sigma)$ occur in C.

[1] In [15], framing was used for marked positions. We decided to use a different notation, because framing introduced problems with the text layout. Our notation should not be confused with the notation for modalities in multi-modal logic.

A closure $C \cdot \sigma$ can be brought into the standard form in the following way: if $x \mapsto t$ is a mapping in σ violating some of the conditions above, then let σ' be $\sigma \setminus \{x \mapsto t\}$, and replace the closure with $C\{x \mapsto t\} \cdot \sigma'\{x \mapsto t\}$.

A closure $(C\sigma_1) \cdot \sigma_2$ is a *retraction* of a closure $C \cdot \sigma$ if $\sigma = \sigma_1\sigma_2$. Intuitively, a retraction is obtained by moving some marked positions lower in the closure. For example, the following is a retraction of the example above:

$$(P(x) \vee g(z) \approx b) \cdot \{x \mapsto f(y), z \mapsto b\} \equiv P([f(y)]) \vee g([b]) \approx b$$

Basic Superposition Parameters. The basic superposition calculus is parameterized with a term ordering $\succ$ and a selection function. An *admissible* term ordering $\succ$ is a reduction ordering total on ground terms, such that $\top$ is the smallest element. An ordering $\succ$ can be extended to an ordering on literals (ambiguously denoted also with $\succ$) by identifying each positive literal $s \approx t$ with a multiset $\{\{s\}, \{t\}\}$, each negative literal $s \not\approx t$ with a multiset $\{\{s, t\}\}$, and comparing these multisets by a two-fold multiset extension $(\succ_{mul})_{mul}$ over $\succ$. The literal ordering obtained in such a way is total on ground literals. The literal $L \cdot \sigma$ is *(strictly) maximal* with respect to a closure $C \cdot \sigma$ if there is no literal $L' \in C$ such that $L'\sigma \succ L\sigma$ ($L'\sigma \succeq L\sigma$) (observe that this definition does not assume that $L \in C$). Similarly, for a closure $C \cdot \sigma$ and a literal $L \in C$, $L \cdot \sigma$ is (strictly) maximal in $C \cdot \sigma$ if and only if it is (strictly) maximal with respect to $(C \setminus L) \cdot \sigma$.

A *selection function* selects a (possibly empty) subset of negative literals in a closure. There are no other constraints on the selection function.

Inference Rules. The basic superposition is a refutation calculus: for N a set of closures saturated up to redundancy, N is unsatisfiable if and only if it contains the empty closure. Intuitively, "saturated up to redundancy" means that any further inference with closures from N produces a closure that has already been derived. To present the rules of basic superposition, we make a technical assumption that all premises are variable-disjoint, and that all premises are expressed using the same substitution. A literal $L \cdot \theta$ is *(strictly) eligible for superposition* in a closure $(C \vee L) \cdot \theta$ if there are no selected literals in $(C \vee L) \cdot \theta$ and $L \cdot \theta$ is (strictly) maximal with respect to $C \cdot \theta$. A literal $L \cdot \theta$ is *eligible for resolution* in a closure $(C \vee L) \cdot \theta$ if it is selected in $(C \vee L) \cdot \theta$ or there are no selected literals in $(C \vee L) \cdot \theta$ and $L \cdot \theta$ is maximal with respect to $C \cdot \theta$. The basic superposition calculus, $\mathcal{BS}$ for short, consists of the following rules:

Positive superposition:
$$\frac{(C \vee s \approx t) \cdot \rho \quad (D \vee w \approx v) \cdot \rho}{(C \vee D \vee w[t]_p \approx v) \cdot \theta}$$

where (i) $\sigma = \mathsf{MGU}(s\rho, w\rho|_p)$ and $\theta = \rho\sigma$, (ii) $t\theta \not\succeq s\theta$ and $v\theta \not\succeq w\theta$, (iii) $(s \approx t) \cdot \theta$ is strictly eligible for superposition in $(C \vee s \approx t) \cdot \theta$, (iv) $(w \approx v) \cdot \theta$ is strictly eligible for superposition in $(D \vee w \approx v) \cdot \theta$, (v) $s\theta \approx t\theta \not\succeq w\theta \approx v\theta$, (vi) $w|_p$ is not a variable.

Negative superposition:
$$\frac{(C \vee s \approx t) \cdot \rho \quad (D \vee w \not\approx v) \cdot \rho}{(C \vee D \vee w[t]_p \not\approx v) \cdot \theta}$$

where (i) $\sigma = \mathsf{MGU}(s\rho, w\rho|_p)$ and $\theta = \rho\sigma$, (ii) $t\theta \not\succeq s\theta$ and $v\theta \not\succeq w\theta$, (iii) $(s \approx t) \cdot \theta$ is strictly eligible for superposition in $(C \vee s \approx t) \cdot \theta$, (iv) $(w \not\approx v) \cdot \theta$ is eligible for resolution in $(D \vee w \not\approx v) \cdot \theta$, (v) $w|_p$ is not a variable.

Reflexivity resolution:
$$\frac{(C \vee s \not\approx t) \cdot \rho}{C \cdot \theta}$$

where (i) $\sigma = \mathsf{MGU}(s\rho, t\rho)$ and $\theta = \rho\sigma$, (ii) $(s \not\approx t) \cdot \theta$ is eligible for resolution in $(C \vee s \not\approx t) \cdot \theta$.

Equality factoring:
$$\frac{(C \vee s \approx t \vee s' \approx t') \cdot \rho}{(C \vee t \not\approx t' \vee s' \approx t') \cdot \theta}$$

where (i) $\sigma = \mathsf{MGU}(s\rho, s'\rho)$ and $\theta = \rho\sigma$, (ii) $t\theta \not\succeq s\theta$ and $t'\theta \not\succeq s'\theta$, (iii) $(s \approx t) \cdot \theta$ is eligible for superposition in $(C \vee s \approx t \vee s' \approx t') \cdot \theta$.

Ordered Hyperresolution:
$$\frac{E_1 \ldots E_n \quad N}{(C_1 \vee \ldots \vee C_n \vee D) \cdot \theta}$$

where (i) E_i are of the form $(C_i \vee A_i) \cdot \rho$, for $1 \leq i \leq n$, (ii) N is of the form $(D \vee \neg B_1 \vee \ldots \vee \neg B_n) \cdot \rho$, (iii) σ is the most general substitution such that $A_i\theta = B_i\theta$ for $1 \leq i \leq n$ and $\theta = \rho\sigma$, (iv) each $A_i \cdot \theta$ is strictly eligible for superposition in E_i, (v) each $\neg B_i \cdot \theta$ is eligible for resolution in N.

In the ordered hyperresolution inference rule, the closures E_i are called the *electrons* or the *side premises*, and the closure N is called the *nucleus* or the *main premise*. In [15, 83] $\mathcal{BS}$ has been presented without the hyperresolution rule. However, as noted in [10], hyperresolution is actually a "macro:" it combines the effects of n negative superpositions of $(A_i \approx \top) \cdot \rho$ from E_i into $(B_i \not\approx \top) \cdot \rho$ of N, resulting in $(\top \not\approx \top) \cdot \theta$, which is immediately eliminated by reflexivity resolution. Furthermore, notice that a positive superposition of a main premise into a positive literal $(B \approx \top) \cdot \rho$ results in a tautology $(\top \approx \top) \cdot \theta$, which can be eliminated. Hence, ordered hyperresolution captures all inferences from several premises involving literals with predicates other than the equality predicate. One might also define ordered factoring, which combines equality resolution on $(C \vee A \approx \top \vee B \approx \top) \cdot \rho$ with reflexivity resolution. We decided not to do this to keep the presentation simpler.

To understand how inference rules of $\mathcal{BS}$ govern the propagation of markers, consider the example of superposition from $[f(x)] \approx [g(x)]$ into $R(x', f(x'))$. We first represent the premises by showing the skeleton and the substitution explicitly: the first premise is equivalent to $(y \approx z) \cdot \{y \mapsto f(x), z \mapsto g(x)\}$, and the second one to $R(x, f(x')) \cdot \{\}$. By the definition of the positive superposition rule, the conclusion is $R(x', z) \cdot \{z \mapsto g(x')\}$, which can be rewritten conveniently as $R(x', [g(x')])$.

Completeness of Basic Superposition. We now briefly overview the completeness proof of the basic superposition. We base our presentation on the proof by Nieuwenhuis and Rubio from [83, 84], which is compatible with the one from [15].

The literal ordering is extended to closures by a multiset extension, where closures are treated as multisets of literals. We denote thus obtained ordering ambiguously with $\succ$. Observe that, since the literal ordering is total on ground literals, the closure ordering is total on ground closures.

Let $C \cdot \sigma$ be a closure and τ a ground substitution. The set of *succeedent-top-left variables* of $C \cdot \sigma$ w.r.t. τ, written as $\mathsf{stlvars}(C \cdot \sigma, \tau)$, is defined as the set of all variables x occurring in a literal $x \approx s \in C$, such that $x\sigma\tau \succ s\sigma\tau$.

Let R be a ground and convergent rewrite system and τ a ground substitution. A variable x occurring in the skeleton C of $C \cdot \sigma$ is *variable irreducible w.r.t. R* if (i) $x\sigma\tau$ is irreducible by rewrite rules in R or (ii) for all $x \approx s \in C$, we have $x \in \mathsf{stlvars}(C \cdot \sigma, \tau)$ and $x\sigma\tau$ is irreducible by those rules $l \Rightarrow r$ from R for which $x\sigma\tau \approx s\sigma\tau \succ l \approx r$. A ground instance $C \cdot \sigma\tau$ is *variable irreducible w.r.t. R* if all variables x from C are variable irreducible w.r.t. R. Let $\mathsf{irred}_R(C \cdot \sigma)$ be the set of all variable irreducible ground instances of $C \cdot \sigma$ w.r.t. R. For a set of closures N, let $\mathsf{irred}_R(N)$ be the set of all variable irreducible ground instances of closures in N w.r.t. R. Finally, let $\mathsf{irred}_R(N)^{\prec D}$ ($\mathsf{irred}_R(N)^{\preceq D}$) be the subset of closures of $\mathsf{irred}_R(N)$ smaller than (or equal to) a ground closure D (w.r.t. ordering on closures $\prec$).

Let ξ by a $\mathcal{BS}$ inference with premises $D_1 \cdot \sigma$ and $D_2 \cdot \sigma$ and a conclusion $C \cdot \rho$. For a rewrite system R and a ground substitution τ such that $\xi\tau$ is a ground instance of ξ, $\xi\tau$ is *variable irreducible w.r.t. R* if all $D_1 \cdot \sigma\tau$, $D_2 \cdot \sigma\tau$ and $C \cdot \rho\tau$ are variable irreducible w.r.t. R.

The notion of redundancy for $\mathcal{BS}$ is defined as follows. A closure $C \cdot \sigma$ is redundant in N if, for all rewrite systems R and all ground substitutions τ such that $C \cdot \sigma\tau$ is variable irreducible w.r.t. R, we have $R \cup \mathsf{irred}_R(N)^{\preceq C \cdot \sigma\tau} \models C \cdot \sigma\tau$. An inference ξ with premises $D_1 \cdot \sigma$ and $D_2 \cdot \sigma$ and a conclusion $C \cdot \rho$ is redundant in N if, for all rewrite systems R and all ground substitutions τ such that $\xi\tau$ is a variable irreducible ground instance of ξ w.r.t. R, $R \cup \mathsf{irred}_R(N)^{\prec D} \models C \cdot \rho\tau$, where $D = \max(D_1 \cdot \sigma\tau, D_2 \cdot \sigma\tau)$. The set of closures N is saturated up to redundancy by $\mathcal{BS}$ if all inferences from premises in N are redundant in N.

A set of closures N is *well-constrained* if $\mathsf{irred}_R(N) \cup R \models N$ for any rewrite system R. If, for all $C \cdot \rho \in N$, ρ is the empty substitution, then N is well-constrained: any variable reducible position of a ground instance of $C \cdot \rho$ can be reduced with rules from R to a closure in $\mathsf{irred}_R(N)$. Furthermore, if N' is obtained from a well-constrained set N by a sound inference rule, then N' is also well-constrained.

Let N be the set of closures obtained by saturating a well-constrained set N_0 up to redundancy by $\mathcal{BS}$; then N is satisfiable if it does not contain the empty closure. Namely, using a variant of the model building technique [15, 83], one may generate a ground convergent rewrite system R_N, which uniquely defines the Herbrand model ${R_N}^*$ such that $s \approx t \in {R_N}^*$ if and only if $\mathsf{nf}_R(s) = \mathsf{nf}_R(t)$. Furthermore, ${R_N}^* \models \mathsf{irred}_{R_N}(N)$.

Finally, since N_0 is well-constrained, N is also well-constrained. Since $R_N \subseteq R_N{}^*$, it follows that $R_N{}^* \models N$.

Redundancy Elimination. Based on the general redundancy notion for basic superposition, several effective *redundancy elimination rules* have been presented in [15], providing means for deleting certain closures or replacing them with simpler ones, without jeopardizing completeness. We overview the most important rules next.

A closure $C \cdot \sigma$ is *reduced modulo substitution η relative to* a closure $D \cdot \theta$ if, for each rewrite systems R and each ground substitution τ, $C \cdot \sigma\eta\tau$ is variable irreducible w.r.t. R whenever $D \cdot \theta\tau$ is variable irreducible w.r.t. R. Checking this condition is difficult, since one needs to consider all ground substitutions and all rewrite systems. However, approximate checks suitable for practice are known. One of them involves the notion of *η-domination*: for two terms $s \cdot \sigma$ and $t \cdot \theta$, we say that s is η-dominated by t, written $s \cdot \sigma \sqsubseteq_\eta t \cdot \theta$, if and only if $s\sigma\eta = t\theta$ and, whenever some variable x from σ occurs in s at position p, then p is in t at or below a variable.

For example, let $s \cdot \sigma = f(g(x), [g(y)])$ and $t \cdot \theta = f([g(c)], [g(h(z))])$. For a substitution $\eta = \{x \mapsto c, y \mapsto h(z)\}$, obviously $s\sigma\eta = t\theta$. Furthermore, each marked position from s can be overlaid at or inside a marked position of t, so $s \cdot \sigma \sqsubseteq_\eta t \cdot \theta$.

This notion can be extended to literals: $(s \approx t) \cdot \sigma \sqsubseteq_\eta (w \approx v) \cdot \theta$ if and only if $s \cdot \sigma \sqsubseteq_\eta w \cdot \theta$ and $t \cdot \sigma \sqsubseteq_\eta v \cdot \theta$, or $s \cdot \sigma \sqsubseteq_\eta v \cdot \theta$ and $t \cdot \sigma \sqsubseteq_\eta w \cdot \theta$. The definition is analogous for negative literals. Furthermore, a positive literal does not η-dominate a negative literal and vice versa. The extension to closures is performed like this: $C \cdot \sigma \sqsubseteq_\eta D \cdot \theta$ if and only if, for each literal $L_1 \cdot \sigma$ from $C \cdot \sigma$, there exists a distinct literal $L_2 \cdot \theta$ from $D \cdot \theta$, such that $L_1 \cdot \sigma \sqsubseteq_\eta L_2 \cdot \theta$. Note that $D \cdot \theta$ is allowed to have more literals than $C \cdot \sigma$.

Now if $C \cdot \sigma \sqsubseteq_\eta D \cdot \theta$, then $C \cdot \sigma$ is reduced relative to $D \cdot \theta$ modulo η. It may happen that, for some η, it holds that $L'\sigma\eta = L\theta$, but not $L' \cdot \sigma \sqsubseteq_\eta L \cdot \theta$. One can make $L' \cdot \sigma$ reduced relative to $L \cdot \theta$ by replacing $L' \cdot \sigma$ with a retraction $L'' \cdot \sigma'$ by instantiating those positions from L that do not overlay into a substitution position of L'. In this way, the application of a simplification or deletion rule may be enabled, while retracting from σ as little information as possible.

With these notions we can finally present the simplification rules. Closure $C \cdot \sigma$ is a *basic subsumer* of $D \cdot \theta$ if there is a substitution η such that $C\sigma\eta \subseteq D\theta$ and $C \cdot \sigma$ is reduced relative to $D \cdot \theta$ modulo η. Additionally, if $C \cdot \sigma$ has fewer literals than $D \cdot \theta$, then $D \cdot \theta$ may be deleted.

A closure $(C \vee A \vee B) \cdot \sigma$ can be replaced with $(C \vee A) \cdot \sigma$ if $A \cdot \sigma \sqsubseteq_{\{\}} B \cdot \sigma$. This is called *duplicate literal deletion*.

A closure $C \cdot \sigma$ can be deleted if $C\sigma$ is a tautology, meaning that $\models C\sigma$. This is called *tautology deletion*. Testing whether $C\sigma$ is a tautology requires itself theorem proving. However, the following simple syntactic checks are typically sufficient: $C \cdot \sigma$ is a tautology if it contains a pair of literals $(s \approx t) \cdot \sigma$ and $(s' \not\approx t') \cdot \sigma$, such that $s\sigma = s'\sigma$ and $t\sigma = t'\sigma$, or a literal of the form $(s \approx t) \cdot \sigma$ with $s\sigma = t\sigma$.

A closure $(C \vee x \not\approx s) \cdot \sigma$, where $x\sigma \succ s\sigma$ is called a *basic tautology* and can be safely deleted. For example, if $f(x) \succ g(x)$, then the closure $[f(x)] \not\approx g(x)$ is a basic tautology. Note that $f(x) \not\approx g(x)$ is not a basic tautology, since $f(x)$ does not occur at a substitution position.

All presented redundancy elimination rules are decidable. In fact, duplicate literal deletion and tautology deletion can be performed in time polynomial in the number of literals. It is well-known that the subsumption check is NP-complete in the number of literals [46], and η-domination can be checked in polynomial time. Finally, the complexity of basic tautology deletion is determined by the complexity of checking ordering constraints.

2.5 Splitting

In some proofs we apply an additional splitting inference rule, presented below. It is borrowed from the semantic tableau calculus and performs an explicit case analysis. If some closure consists of two parts not having variables in common, one may separately assume that either part is true. If unsatisfiability is proved in both cases, the initial closure set is evidently unsatisfiable. Each of the cases introduced by the splitting rule is called a *branch*.

Splitting:
$$\frac{N \cup \{C \vee D\}}{N \cup \{C\} \;\mid\; N \cup \{D\}}$$

where (i) N is a set of closures, (ii) C and D do not have variables in common.

2.6 Disjunctive Datalog

In this section we briefly present the syntax and semantics of disjunctive datalog. This presentation is standard and may be found in [38, 48].

Let Σ be a first-order signature such that (i) for each function symbol $f \in \mathcal{F}(\Sigma)$ the arity of f is zero and $(ii) \approx \in \mathcal{P}(\Sigma)$ is a special equality predicate with the arity of two. A *disjunctive datalog program with equality* P is a finite set of rules of the form

$$A_1 \vee ... \vee A_n \leftarrow B_1, ..., B_m$$

where $n \geq 0$, $m \geq 0$, and A_i and B_i are atoms defined over Σ. Furthermore, each rule must be *safe*, that is, each variable occurring in a head literal must occur in a body literal as well. For a rule r, the set of atoms $\mathsf{head}(r) = \{A_i\}$ is called the *rule head*, whereas the set of atoms $\mathsf{body}(r) = \{B_i\}$ is called the *rule body*. A ground rule with an empty body is called a *fact*.

Typical definitions of a disjunctive datalog program, e.g. from [38, 48], allow negated atoms in the body. This negation is usually non-monotonic, and is different from negation in first-order logic. Our approach produces only positive disjunctive datalog programs, so we omit non-monotonic negation from the definitions.

The semantics of disjunctive datalog programs is defined as follows. The *ground instance* of P over the Herbrand universe of P, written $\mathsf{ground}(P, HU)$, is the set of ground rules obtained by replacing variables in each rule of P with constants from HU in all possible ways. The *Herbrand base* HB of P is the set of all ground atoms defined over predicates from $\mathcal{P}(\Sigma)$. An *interpretation* M of P is a subset of HB. We say that some ground atom A is true in an interpretation M if $A \in M$; A is false in M if $A \notin M$. An interpretation M is a *model* of P if, for each rule $r \in \mathsf{ground}(P, HU)$, if $\mathsf{body}(r) \subseteq M$, then $\mathsf{head}(r) \cap M \neq \emptyset$ and if all atoms from M involving the $\approx$ predicate yield an equality relation (i.e. a relation that is reflexive, symmetric, transitive, and, for any predicate symbol $R \in \mathcal{P}(\Sigma)$, if $R(\ldots, a, \ldots) \in M$ and $a \approx b \in M$, then $R(\ldots, b, \ldots) \in M$ as well).

A model M is *minimal* if no subset of M is a model. The semantics of P is the set of all minimal models of P, which is denoted by $\mathcal{MM}(P)$. Finally, the notion of query answering is defined as follows. A ground literal A is a *cautious answer* of P (written $P \models_c A$) if all minimal models of the program contain A; A is a *brave answer* of P (written $P \models_b A$) if at least one minimal model of the program contains A. First-order entailment coincides with cautious entailment for positive ground atoms.

The size of a rule r is defined as $|r| = 1 + \sum_{1 \leq i \leq n} |A_i| + \sum_{1 \leq j \leq m} |B_j|$, where the size of atoms A_i and B_j is defined as $|S(t_1, \ldots, t_n)| = 1 + n$: predicates and terms are encoded with one symbol, and the leading 1 in the definition of $|r|$ accounts for the implication symbol separating the head from the body. The size of a program P, written $|P|$, is the sum of the sizes of all its rules.

2.7 Description Logic $\mathcal{SHIQ}$

Definition 2.7.1. *Let N_{R_a} be the set of abstract role names. The set of $\mathcal{SHIQ}$* abstract roles *is the set $N_{R_a} \cup \{R^- \mid R \in N_{R_a}\}$. Let $\mathsf{Inv}(R) = R^-$ and $\mathsf{Inv}(R^-) = R$ for $R \in N_{R_a}$. A $\mathcal{SHIQ}$* RBox *$\mathcal{R}$ over N_{R_a} is a finite set of transitivity axioms $\mathsf{Trans}(R)$ and abstract role inclusion axioms $R \sqsubseteq S$, such that $R \sqsubseteq S \in \mathcal{R}$ implies $\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \in \mathcal{R}$, and $\mathsf{Trans}(R) \in \mathcal{R}$ implies $\mathsf{Trans}(\mathsf{Inv}(R)) \in \mathcal{R}$.*

Let $\sqsubseteq^*$ denote the reflexive-transitive closure of $\sqsubseteq$. A role R is transitive *if $\mathsf{Trans}(S) \in \mathcal{R}$ or $\mathsf{Trans}(\mathsf{Inv}(S)) \in \mathcal{R}$ for some S with $S \sqsubseteq^* R$ and $R \sqsubseteq^* S$; R is* simple *if there is no role S such that $S \sqsubseteq^* R$ and S is transitive; R is* complex *if it is not simple.*

Let N_C be a set of atomic *concept names. The set of $\mathcal{SHIQ}$* concepts *over N_C and N_{R_a} is defined inductively as the minimal set for which the following holds: $\top$ and $\bot$ are $\mathcal{SHIQ}$ concepts, each atomic concept name $A \in N_C$ is a $\mathcal{SHIQ}$ concept, and, if C and D are $\mathcal{SHIQ}$ concepts, R is an abstract role, S is an abstract simple role and n is an integer, then $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$, $\forall R.C$, $\leq n\, S.C$, $\geq n\, S.C$, are also $\mathcal{SHIQ}$ concepts.*

A $\mathcal{SHIQ}$ TBox *$\mathcal{T}$ over N_C and $\mathcal{R}$ is a finite set of* concept inclusion axioms *$C \sqsubseteq D$ or* concept equivalence axioms *$C \equiv D$, where C and D are $\mathcal{SHIQ}$ concepts.*

Let N_{I_a} be a set of abstract individual names. *A $\mathcal{SHIQ}$* ABox $\mathcal{A}$ *is a set of concept and abstract role* membership axioms $C(a)$, $R(a,b)$, $\neg S(a,b)$, *and* (in)equality axioms $a \approx b$ *and* $a \not\approx b$, *where C is a $\mathcal{SHIQ}$ concept, R is an abstract role, S is an abstract simple role and a and b are abstract individuals.*

A $\mathcal{SHIQ}$ knowledge base KB is a triple $(KB_{\mathcal{R}}, KB_{\mathcal{T}}, KB_{\mathcal{A}})$, where $KB_{\mathcal{R}}$ is an RBox, $KB_{\mathcal{T}}$ is a TBox, and $KB_{\mathcal{A}}$ is an ABox.

Definition 2.7.1 differs from typical definitions in two aspects. First, since OWL-DL lacks the unique name assumption (UNA), we do not incorporate UNA into the definition of $\mathcal{SHIQ}$, but allow the user to axiomatize it by including an inequality axiom $a_i \not\approx a_j$ for each pair of distinct abstract individuals, respectively, cf. [5, page 60]. Second, usual definitions do not provide for ABox axioms involving negative roles. We allow such assertions as they allow checking entailment of ground role facts.

Definition 2.7.2. *The semantics of a $\mathcal{SHIQ}$ knowledge base KB is given by the mapping π which transforms KB axioms into a first-order formula, as shown in Table 2.1. Each atomic concept is mapped into a unary predicate and each abstract role is mapped into a binary predicate.*

The basic inference problem for $\mathcal{SHIQ}$ is checking satisfiability *of KB, that is, determining whether a first-order model of $\pi(KB)$ exists. Other interesting inference problems can be reduced to satisfiability as follows, where α denotes a new abstract individual not occurring in the knowledge base:*

- Concept satisfiability. *A concept C is satisfiable with respect to KB if and only if there exists a model of KB in which the interpretation of C is not empty. This is the case if and only if $KB \cup \{C(\alpha)\}$ is satisfiable.*

- Subsumption. *A concept C is subsumed by a concept D with respect to KB if and only if $\pi(KB) \models \pi(C \sqsubseteq D)$. This is the case if and only if $KB \cup \{(C \sqcap \neg D)(\alpha)\}$ is unsatisfiable.*

- Instance checking. *An individual a is an instance of a concept C with respect to KB if and only if $\pi(KB) \models \pi(C(a))$. This is the case if and only if $KB \cup \{\neg C(a)\}$ is unsatisfiable.*

- Role checking. *A simple abstract role S relates individuals a and b with respect to KB if and only if $\pi(KB) \models \pi(S(a,b))$. This is the case if and only if $KB \cup \{\neg S(a,b)\}$ is unsatisfiable.*

The semantics of description logics is usually given by a direct model-theoretic semantics. An interpretation $I = (\triangle^I, \cdot^I)$ consists of a domain set $\triangle^I$ and an interpretation function $\cdot^I$, which assigns to each individual a an element a^I, to each atomic concept A a set $A^I \subseteq \triangle^I$, and to each role R a relation $R^I \subseteq \triangle^I \times \triangle^I$. The semantics of concepts and axioms is given in Table 2.2, where C and D are concepts, R and S are roles and a and b are individuals. These two semantics are equivalent, as first shown

Table 2.1: Semantics of $\mathcal{SHIQ}$ by Mapping to FOL

Mapping Concepts to FOL		
$\pi_y(\top, X)$	$=$	$\top$
$\pi_y(\bot, X)$	$=$	$\bot$
$\pi_y(A, X)$	$=$	$A(X)$
$\pi_y(\neg C, X)$	$=$	$\neg \pi_y(C, X)$
$\pi_y(C \sqcap D, X)$	$=$	$\pi_y(C, X) \wedge \pi_y(D, X)$
$\pi_y(C \sqcup D, X)$	$=$	$\pi_y(C, X) \vee \pi_y(D, X)$
$\pi_y(\forall R.C, X)$	$=$	$\forall y : R(X, y) \rightarrow \pi_x(C, y)$
$\pi_y(\exists R.C, X)$	$=$	$\exists y : R(X, y) \wedge \pi_x(C, y)$
$\pi_y(\leq n\, R.C, X)$	$=$	$\forall y_1, \ldots, y_{n+1} : \bigwedge R(X, y_i) \wedge \bigwedge \pi_x(C, y_i) \rightarrow \bigvee y_i \approx y_j$
$\pi_y(\geq n\, R.C, X)$	$=$	$\exists y_1, \ldots, y_n : \bigwedge R(X, y_i) \wedge \bigwedge \pi_x(C, y_i) \wedge \bigwedge y_i \not\approx y_j$
Mapping Axioms to FOL		
$\pi(C \sqsubseteq D)$	$=$	$\forall x : \pi_y(C, x) \rightarrow \pi_y(D, x)$
$\pi(C \equiv D)$	$=$	$\forall x : \pi_y(C, x) \leftrightarrow \pi_y(D, x)$
$\pi(R \sqsubseteq S)$	$=$	$\forall x, y : R(x, y) \rightarrow S(x, y)$
$\pi(\mathsf{Trans}(R))$	$=$	$\forall x, y, z : R(x, y) \wedge R(y, z) \rightarrow R(x, z)$
$\pi(C(a))$	$=$	$\pi_y(C, a)$
$\pi((\neg)R(a, b))$	$=$	$(\neg)R(a, b)$
$\pi(a \circ b)$	$=$	$a \circ b$ for $\circ \in \{\approx, \not\approx\}$
Mapping KB to FOL		
$\pi(R)$	$=$	$\forall x, y : R(x, y) \leftrightarrow R^{-}(y, x)$
$\pi(KB_{\mathcal{R}})$	$=$	$\bigwedge_{\alpha \in KB_{\mathcal{R}}} \pi(\alpha) \wedge \bigwedge_{R \in N_{R_a}} \pi(R)$
$\pi(KB_{\mathcal{T}})$	$=$	$\bigwedge_{\alpha \in KB_{\mathcal{T}}} \pi(\alpha)$
$\pi(KB_{\mathcal{A}})$	$=$	$\bigwedge_{\alpha \in KB_{\mathcal{A}}} \pi(\alpha)$
$\pi(KB)$	$=$	$\pi(KB_{\mathcal{R}}) \wedge \pi(KB_{\mathcal{T}}) \wedge \pi(KB_{\mathcal{A}})$

Notes:
(i): X is a meta variable and is substituted by the actual variable,
(ii): π_x is defined as π_y by substituting x and x_i for all y and y_i, respectively.

Table 2.2: Direct Model-theoretic Semantics of $\mathcal{SHIQ}$

Interpreting Concepts		
$\top^I$	$=$	$\triangle^I$
$\bot^I$	$=$	$\emptyset$
$(\neg C)^I$	$=$	$\triangle^I \setminus C^I$
$(C \sqcap D)^I$	$=$	$C^I \cap D^I$
$(C \sqcup D)^I$	$=$	$C^I \cap D^I$
$(\forall R.C)^I$	$=$	$\{x \mid \forall y : (x,y) \in R^I \rightarrow y \in C^I\}$
$(\exists R.C)^I$	$=$	$\{x \mid \exists y : (x,y) \in R^I \wedge y \in C^I\}$
$(\leq n\, R.C)^I$	$=$	$\{x \mid \sharp\{y \mid (x,y) \in R^I \wedge y \in C^I\} \leq n\}$
$(\geq n\, R.C)^I$	$=$	$\{x \mid \sharp\{y \mid (x,y) \in R^I \wedge y \in C^I\} \geq n\}$
Semantics of Axioms		
$C \sqsubseteq D$		$C^I \subseteq D^I$
$C \equiv D$		$C^I = D^I$
$R \sqsubseteq S$		$R^I \subseteq S^I$
$\mathsf{Trans}(R)$		R is transitive
$C(a)$		$a^I \in C^I$
$R(a,b)$		$(a^I, b^I) \in R^I$
$\neg R(a,b)$		$(a^I, b^I) \notin R^I$
$a \circ b$		$a^I \circ b^I$ for $\circ \in \{\approx, \not\approx\}$

by Borgida in [21]. We now define several restrictions of $\mathcal{SHIQ}$, which we consider in our work.

Definition 2.7.3. *For a knowledge base KB, a role R is called* very simple *if no role S exists, such that $S \sqsubseteq R \in KB_{\mathcal{R}}$. Description logic $\mathcal{SHIQ}^-$ has the same syntax and semantics as $\mathcal{SHIQ}$, with the additional syntactical restriction that only very simple roles are allowed to occur in number restrictions $\leq n\, R.C$ and $\geq n\, R.C$.*

$\mathcal{ALCHIQ}$ ($\mathcal{ALCHIQ}^-$) is a fragment of $\mathcal{SHIQ}$ ($\mathcal{SHIQ}^-$), which does not allow transitivity axioms to occur in an RBox. $\mathcal{ALC}$ is the fragment of $\mathcal{ALCHIQ}$ which does not provide for role hierarchies, inverse roles and qualified number restrictions.

Notice that the relation $\sqsubseteq^*$ can be a cyclic relation in general. In [114] it has been shown that a $\mathcal{SHIQ}$ knowledge base KB with a cyclic role hierarchy can be reduced to a knowledge base KB' with an acyclic role hierarchy, as follows. First, we compute the set of maximal, strongly connected components (or maximal cycles) of the role inclusion relation $\sqsubseteq$ of KB. For each strongly connected component Γ, we select one representative role, denoted as $\mathsf{role}(\Gamma)$, such that, if $R \in \Gamma$, $\mathsf{Inv}(R) \in \Gamma'$ and $\mathsf{role}(\Gamma) = R$, then $\mathsf{role}(\Gamma') = \mathsf{Inv}(R)$. (Since we assume that $R \sqsubseteq S \in KB_{\mathcal{R}}$ implies $\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \in KB_{\mathcal{R}}$, we have that, if $R, S \in \Gamma$ and $\mathsf{Inv}(R) \in \Gamma'$, then $\mathsf{Inv}(S) \in \Gamma'$, so the definition of $\mathsf{role}(\Gamma)$ is correct.) Next, we form the new TBox $KB'_{\mathcal{T}}$ and ABox

$KB'_{\mathcal{A}}$ by replacing, in all axioms of $KB_{\mathcal{A}}$ and $KB_{\mathcal{T}}$, each role R with $\mathsf{role}(\Gamma)$, where Γ is the maximal, strongly connected component that R belongs to. Finally, we construct the new RBox $KB'_{\mathcal{R}}$ as follows: (i) for each pair of strongly connected components $\Gamma \neq \Gamma'$, we add the axiom $\mathsf{role}(\Gamma) \sqsubseteq \mathsf{role}(\Gamma')$ to $KB'_{\mathcal{R}}$ if there are roles $R \in \Gamma$ and $R' \in \Gamma'$ with $R \sqsubseteq R'$, and (ii) for each strongly connected component C, we add the axiom $\mathsf{Inv}(\mathsf{role}(\Gamma)) \sqsubseteq \mathsf{role}(\Gamma)$ to $KB'_{\mathcal{R}}$ if there is a role $R \in \Gamma$, such that also $\mathsf{Inv}(R) \in \Gamma$. Since the strongly connected components of $\sqsubseteq$ can be computed in time quadratic in the number of roles, this reduction can be performed in polynomial time. Hence, we can assume without loss of generality that RBoxes are acyclic.

A concept C is in *negation-normal form* if all negations in it occur in front of atomic concepts only. C can be transformed in time linear in the size of C into an equivalent concept in negation-normal form, denoted as $\mathsf{NNF}(C)$, by exhaustively applying the following rewrite rules to subconcepts of C:

$$\begin{array}{rcl@{\qquad}rcl} \neg\top & \rightsquigarrow & \bot & \neg\bot & \rightsquigarrow & \top \\ \neg(C_1 \sqcap C_2) & \rightsquigarrow & \neg C_1 \sqcup \neg C_2 & \neg(C_1 \sqcup C_2) & \rightsquigarrow & \neg C_1 \sqcap \neg C_2 \\ \neg(\exists R.C) & \rightsquigarrow & \forall R.\neg C & \neg(\forall R.C) & \rightsquigarrow & \exists R.\neg C \\ \neg(\geq (n+1)\, R.C) & \rightsquigarrow & \leq n\, R.C & \neg(\leq n\, R.C) & \rightsquigarrow & \geq (n+1)\, R.C \end{array}$$

With $|KB|$ we denote the *size of the knowledge base* assuming unary coding of numbers, which is computed recursively in the following way, for C and D concepts, A an atomic concept, and R and S roles:

- $|KB| = |KB_{\mathcal{R}}| + |KB_{\mathcal{T}}| + |KB_{\mathcal{A}}|$,
- $|KB_{\mathcal{R}}| = \sum_{\alpha \in KB_{\mathcal{R}}} |\alpha|$,
- $|KB_{\mathcal{T}}| = \sum_{\alpha \in KB_{\mathcal{T}}} |\alpha|$,
- $|KB_{\mathcal{A}}| = \sum_{\alpha \in KB_{\mathcal{A}}} |\alpha|$,
- $|R \sqsubseteq S| = 3$,
- $|\mathsf{Trans}(R)| = 2$,
- $|C \sqsubseteq D| = |C \equiv D| = |C| + |D| + 1$,
- $|R(a, b)| = 3$,
- $|C(a)| = |C| + 1$,
- $|\top| = |\bot| = 1$,
- $|A| = |\neg A| = 2$,
- $|C \sqcup D| = |C \sqcap D| = |C| + |D| + 1$,
- $|\exists R.C| = |\forall R.C| = 2 + |C|$,

- $|\geq n\, R.C| = |\leq n\, R.C| = n + 2 + |C|$.

Intuitively, $|KB|$ is the number of symbols needed to encode KB on the input tape of a Turing machine using the unary encoding of numbers. We use a single symbol for each atomic concept, role and individual. The n in the definition of the length of concepts $\geq n\, R.C$ and $\leq n\, R.C$ stems from the assumption on unary coding of numbers: a number n can be encoded in unary coding with n bits.

The notion of positions is extended to $\mathcal{SHIQ}$ concepts and axioms in the obvious way:

- $\alpha|_\epsilon = \alpha$ for α a concept or an axiom;
- $(\neg D)|_{1p} = D|_p$;
- $(D_1 \circ D_2)|_{ip} = D_i|_p$ for $\circ \in \{\sqcap, \sqcup, \sqsubseteq, \equiv\}$ and $i \in \{1, 2\}$;
- for $C = \bowtie R.D$ with $\bowtie \in \{\exists, \forall\}$, $C|_1 = R$ and $C|_{2p} = D|_p$;
- for $C = \bowtie n\, R.D$ with $\bowtie \in \{\leq, \geq\}$, $C|_1 = n$, $C|_2 = R$ and $C|_{3p} = D|_p$;
- $\mathsf{Trans}(R)|_1 = R$;
- for $\alpha = D(a)$, $\alpha|_{1p} = D|_p$ and $\alpha|_2 = a$;
- for $\alpha = R(a, b)$, $\alpha|_1 = R$, $\alpha|_2 = a$ and $\alpha|_3 = b$;
- $(a_1 \circ a_2)|_i = a_i$ for $\circ \in \{\approx, \not\approx\}$ and $i \in \{1, 2\}$.

For α a $\mathcal{SHIQ}$ concept or an axiom and p a position in α, replacement of the subobject $\alpha|_p$ with β is denoted as $\alpha[\beta]_p$ and is defined in the obvious way. Furthermore, if $\alpha|_p$ is a concept, then the polarity of p is defined to agree with the polarity of the corresponding position in translation of α into first-order logic, in the following way:

- $\mathsf{pol}(C, \epsilon) = 1$;
- $\mathsf{pol}(\neg C, 1p) = -\mathsf{pol}(C, p)$;
- $\mathsf{pol}(C_1 \circ C_2, ip) = \mathsf{pol}(C_i, p)$ for $\circ \in \{\sqcap, \sqcup\}$ and $i \in \{1, 2\}$;
- $\mathsf{pol}(\bowtie R.C, 2p) = \mathsf{pol}(C, p)$ for $\bowtie \in \{\exists, \forall\}$;
- $\mathsf{pol}(\leq n\, R.C, 3p) = -\mathsf{pol}(C, p)$;
- $\mathsf{pol}(\geq n\, R.C, 3p) = \mathsf{pol}(C, p)$;
- $\mathsf{pol}(C_1 \sqsubseteq C_2, 1p) = -\mathsf{pol}(C_1, p)$ and $\mathsf{pol}(C_1 \sqsubseteq C_2, 2p) = \mathsf{pol}(C_2, p)$;
- $\mathsf{pol}(C_1 \equiv C_2, ip) = 0$ for $i \in \{1, 2\}$;
- $\mathsf{pol}(C(a), 1p) = \mathsf{pol}(C, p)$.

2.8 Description Logics with Concrete Domains

Practical applications of description logics usually require representing *concrete* properties such as height, name, or age, having values from a fixed domain such as integers or strings, with *built-in* predicates. These requirements led to the extension of description logics with *concrete domains* [6]. Informally, a concrete domain consists of a set of predicates with a predefined interpretation. If a decision procedure for checking satisfiability of finite conjunctions over concrete domain predicates exists, many DLs can be combined with a concrete domain while retaining decidability. Unfortunately, in [75] it was shown that a logic with general inclusion axioms and concrete domains is undecidable. Therefore, in [59, 89, 53] several alternatives have been investigated. The cumulative results of this research influenced the design of the Ontology Web Language (OWL) [91], which, in its DL version, supports a basic form of concrete domains, so-called *datatypes*.

Before presenting the formal definition of description logics with concrete domains, we give an intuitive example. Let us introduce an atomic concept *Paper* to represent bibliographical information about papers. The concrete role *pages* represents the number of pages a paper has. We make this role functional by including the axiom $\top \sqsubseteq \leq 1\, \mathit{pages}$. To define long papers as papers with 25 pages or more, we use a unary concrete domain predicate $\geq_{25}$, which is interpreted as the set of all integers greater than 25: $\mathit{LongPaper} \equiv \mathit{Paper} \sqcap \exists \mathit{pages}.\ \geq_{25}$. Similarly, we define a short paper as $\mathit{ShortPaper} \equiv \mathit{Paper} \sqcap \exists \mathit{pages}.\ \leq_8$. From these definitions and the fact that the interpretations of the predicates $\geq_{25}$ and $\leq_8$ are disjoint, we may infer that the concept $\mathit{LongPaper} \sqcap \mathit{ShortPaper}$ is unsatisfiable, and thus prevent a potential modeling error.

2.8.1 Concrete Domains

To present our definitions in a concise way, we introduce the following notation: with $\mathbf{x}$ we denote a vector of variables $x_1, \ldots, x_n$, and for a function δ, with $\delta(\mathbf{x})$ we denote the application of δ to each element x_i of $\mathbf{x}$, i.e. $\delta(x_1), \ldots, \delta(x_n)$.

Definition 2.8.1. *A concrete domain* $\mathbf{D}$ *is a pair* $(\triangle_{\mathbf{D}}, \Phi_{\mathbf{D}})$*, where* $\triangle_{\mathbf{D}}$ *is a set, called the* domain *of* $\mathbf{D}$*, and* $\Phi_{\mathbf{D}}$ *is a finite set of* predicate names*. Each* $d \in \Phi_{\mathbf{D}}$ *is associated with an arity* n *and an* extension $d^{\mathbf{D}} \subseteq \triangle_{\mathbf{D}}^n$*. A concrete domain* $\mathbf{D}$ *is* admissible *if:*

- $\Phi_{\mathbf{D}}$ *is closed under negation, i.e. for each* $d \in \Phi_{\mathbf{D}}$*, there exists* $\overline{d} \in \Phi_{\mathbf{D}}$ *with* $\overline{d}^{\mathbf{D}} = \triangle_{\mathbf{D}}^n \setminus d^{\mathbf{D}}$*,*
- *it contains a unary predicate* $\top_{\mathbf{D}}$ *interpreted as* $\triangle_{\mathbf{D}}$*, and*
- $\mathbf{D}$*-satisfiability of finite conjunctions of the form* $\bigwedge_{i=1}^{n} d_i(\mathbf{x_i})$ *is decidable. The latter is the case if an assignment* δ *of variables to elements of* $\triangle_{\mathbf{D}}$ *exists, such that* $\delta(\mathbf{x_i}) \in d_i^{\mathbf{D}}$*, for each* $1 \leq i \leq n$*.*

Sometimes, we consider conjunctions over literals containing terms, rather than variables. Let $S = \{d_i(\mathbf{t_i})\}$ be a set of literals, where $\mathbf{t_i}$ is a vector of terms $t_{i1}, \ldots, t_{ik}$. With $\widehat{S}$ we denote a conjunction $C = \bigwedge d_i(\mathbf{x_i})$, obtained from S by replacing each occurrence of a term with the same variable, such that different terms are replaced with distinct variables. For two conjunctions C_1 and C_2, we write $C_1 \equiv C_2$ if they are equivalent up to variable renaming.

Extending first-order logic with a concrete domain is significantly simplified if the interpretation of concrete objects is separated from the interpretation of other objects. Hence, we assume that the set of sorts $\mathcal{S}$ of a signature Σ contains the sort c for the concrete objects, and the sort a for all other so-called *abstract* objects. Since the concrete domain does not provide an interpretation for function symbols, we do not allow nesting function symbols of sort c. Furthermore, all arguments of predicates from $\Phi_{\mathbf{D}}$ must be of sort c. To distinguish the sorts syntactically, we denote the variables (function symbols) of sort c as x^{c} (f^{c}).

Definition 2.8.2. *Let* $\mathbf{D}$ *be an admissible concrete domain, and* Σ *a signature such that* $\mathsf{c} \in \mathcal{S}$, $\Phi_{\mathbf{D}} \subseteq \mathcal{P}$, *the signature of each predicate from* $\Phi_{\mathbf{D}}$ *is* $\mathsf{c} \times \ldots \times \mathsf{c}$, *and for each function symbol* $f \in \mathcal{F}$, *no argument is of sort* c. *Let* φ *be a formula defined over* Σ *and* I *a "classic" multi-sorted model of* φ, *i.e.* $I \models \varphi$, *where concrete predicates are treated as normal predicates.* I *is a* $\mathbf{D}$-model *of* φ, *written* $I \models_{\mathbf{D}} \varphi$, *if and only if a valuation* $\delta : \mathcal{D}_{\mathsf{c}} \to \triangle_{\mathbf{D}}$ *exists, such that for each* $(\alpha_1, \ldots, \alpha_n) \in d^I$, $(\delta(\alpha_1), \ldots, \delta(\alpha_n)) \in d^{\mathbf{D}}$. *The usual notions of validity, satisfiability and entailment are generalized to* $\mathbf{D}$*-validity,* $\mathbf{D}$*-satisfiability and* $\mathbf{D}$*-entailment in the obvious way.*

The following simple lemma demonstrates that the $\mathbf{D}$-satisfiability can be considered with respect to Herbrand models only, without loss of generality.

Lemma 2.8.3. *A formula* φ *is* $\mathbf{D}$*-satisfiable if and only if* $\mathsf{sk}(\varphi)$ *is* $\mathbf{D}$*-satisfiable in a Herbrand model.*

Proof. The ($\Leftarrow$) direction follows immediately by definition of $\mathbf{D}$-satisfiability. For the ($\Rightarrow$) direction, if φ is $\mathbf{D}$-satisfiable in some model I with valuation δ, then there is a Herbrand model I' such that $I' \models \mathsf{sk}(\varphi)$, where each term t'_i from the Herbrand model corresponds to some α_i in I [42]. We then define the valuation δ' over the Herbrand universe of I' by setting $\delta'(t'_i) = \delta(\alpha_i)$. Obviously, $I' \models_{\mathbf{D}} \mathsf{sk}(\varphi)$. □

Notice that, since admissible concrete domains are closed under negation of predicates, we may assume that concrete domain predicates occur in all clauses positively. Also, when ambiguity does not arise, we do not stress $\mathbf{D}$ for satisfiability, equisatisfiability, entailment etc.

2.8.2 Description Logic $\mathcal{SHIQ}(\mathbf{D})$

We now present the formal definition of the $\mathcal{SHIQ}(\mathbf{D})$ description logic, which is obtained by combining $\mathcal{SHIQ}$ with an admissible concrete domain $\mathbf{D}$. The syntax

of $\mathcal{SHIQ}$ from Definition 2.7.1 can be extended to allow for concrete domains in the following way:

Definition 2.8.4. *Let N_{R_c} be the set of* concrete roles*. Additionally to $\mathcal{SHIQ}$ RBox axioms, a $\mathcal{SHIQ}(\mathbf{D})$ RBox $\mathcal{R}$ can contain a finite number concrete role inclusion axioms $T \sqsubseteq U$*[2]*.*

Let $\mathbf{D}$ be an admissible concrete domain. In addition to $\mathcal{SHIQ}$ concepts, the set of $\mathcal{SHIQ}(\mathbf{D})$ concepts contains $\exists T_1, \ldots, T_m.d$, $\forall T_1, \ldots, T_m.d$, $\leq n\, T$, $\geq n\, T$, for $T_{(i)}$ concrete roles, d an m-ary concrete domain predicate and n an integer. A $\mathcal{SHIQ}(\mathbf{D})$ TBox $\mathcal{T}$ is defined analogously to Definition 2.7.1, with the difference that concepts occurring in axioms are $\mathcal{SHIQ}(\mathbf{D})$ concepts.

Let N_{I_c} be a set of concrete individual names*. Additionally to $\mathcal{SHIQ}$ ABox axioms, a $\mathcal{SHIQ}(\mathbf{D})$ RBox $\mathcal{R}$ can contain a finite number of concrete role membership axioms $(\neg)T(a, b^{\mathsf{c}})$ and* (in)equality axioms *$a^{\mathsf{c}} \approx b^{\mathsf{c}}$ and $a^{\mathsf{c}} \not\approx b^{\mathsf{c}}$, where T is a concrete role, a an abstract individual, and a^{c} and b^{c} are concrete individuals.*

The semantics of $\mathcal{SHIQ}$ from Definition 2.7.2 can be extended to give semantics to $\mathcal{SHIQ}(\mathbf{D})$ in the following way:

Definition 2.8.5. *The semantics of a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB is given by extending the mapping π from Definition 2.7.2 to translate KB into a multi-sorted first-order formula, as presented in Table 2.3. Atomic concept predicates have the signature a, abstract roles have the signature $\mathsf{a} \times \mathsf{a}$, concrete roles have the signature $\mathsf{a} \times \mathsf{c}$, and n-ary concrete domain predicates have the signature $\mathsf{c} \times \ldots \times \mathsf{c}$.*

The basic inference problem for $\mathcal{SHIQ}(\mathbf{D})$ is checking satisfiability *of KB, that is, determining whether a $\mathbf{D}$-model of $\pi(KB)$ exists. The other inference problems are defined analogously as in Definition 2.7.2.*

Direct model-theoretic semantics can be easily extended to $\mathcal{SHIQ}(\mathbf{D})$, by assigning to each concrete individual a^{c} an element $(a^{\mathsf{c}})^I \in \triangle_{\mathbf{D}}$, and by assigning to each concrete role T a set $T^I \subseteq \triangle^I \times \triangle_{\mathbf{D}}$. The semantics of new concepts and axioms is given in Table 2.4, where $T_{(i)}$ and U are concrete roles, and d is a concrete predicate.

The $\mathcal{SHIQ}^-(\mathbf{D})$ and $\mathcal{ALCHIQ}^-(\mathbf{D})$ fragments of $\mathcal{SHIQ}(\mathbf{D})$ are defined as in Section 2.7. Below are the additional rewrite rules for reducing $\mathcal{SHIQ}(\mathbf{D})$ concepts to negation-normal form:

$$\neg(\geq (n+1)\, T) \rightsquigarrow \leq n\, T \qquad \neg(\leq n\, T) \rightsquigarrow \geq (n+1)\, T$$
$$\neg(\exists T_1, \ldots, T_n.d) \rightsquigarrow \forall T_1, \ldots, T_n.\overline{d} \qquad \neg(\forall T_1, \ldots, T_n.d) \rightsquigarrow \exists T_1, \ldots, T_n.\overline{d}$$

The size of a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB is obtained by extending the definition from Section 2.7 to handle new constructs in the following way:

[2] Inverse concrete roles do not make sense semantically, so we do not distinguish between concrete roles and concrete role names. Furthermore, transitive concrete roles also do not make sense semantically, so transitivity axioms are not allowed for concrete roles and all concrete roles are simple.

Table 2.3: Semantics of $\mathcal{SHIQ}(\mathbf{D})$ by Mapping to FOL

Mapping Concepts to FOL		
$\pi_y(\forall T_1, \ldots, T_m.d, X)$	$=$	$\forall y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}} : \bigwedge T_i(X, y_i^{\mathsf{c}}) \rightarrow d(y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}})$
$\pi_y(\exists T_1, \ldots, T_m.d, X)$	$=$	$\exists y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}} : \bigwedge T_i(X, y_i^{\mathsf{c}}) \wedge d(y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}})$
$\pi_y(\leq n\, T, X)$	$=$	$\forall y_1^{\mathsf{c}}, \ldots, y_{n+1}^{\mathsf{c}} : \bigwedge T(X, y_i^{\mathsf{c}}) \rightarrow \bigvee\, y_i^{\mathsf{c}} \approx y_j^{\mathsf{c}}$
$\pi_y(\geq n\, T, X)$	$=$	$\exists y_1^{\mathsf{c}}, \ldots, y_n^{\mathsf{c}} : \bigwedge T(X, y_i^{\mathsf{c}}) \wedge \bigwedge\, y_i^{\mathsf{c}} \not\approx y_j^{\mathsf{c}}$
Mapping Axioms to FOL		
$\pi(T \sqsubseteq U)$	$=$	$\forall x^{\mathsf{c}}, y^{\mathsf{c}} : T(x^{\mathsf{c}}, y^{\mathsf{c}}) \rightarrow U(x^{\mathsf{c}}, y^{\mathsf{c}})$
$\pi((\neg)T(a, b^{\mathsf{c}}))$	$=$	$(\neg)T(a, b^{\mathsf{c}})$
$\pi(a^{\mathsf{c}} \circ b^{\mathsf{c}})$	$=$	$a^{\mathsf{c}} \circ b^{\mathsf{c}}$ for $\circ \in \{\approx, \not\approx\}$

Table 2.4: Direct Model-theoretic Semantics of $\mathcal{SHIQ}(\mathbf{D})$

Interpreting Concepts		
$(\forall T_1, \ldots, T_m.d)^I$	$=$	$\{x \mid \forall y_1, \ldots, y_m : (x, y_1) \in T_1^I \wedge \ldots \wedge (x, y_m) \in T_m^I \rightarrow (y_1, \ldots, y_m) \in d^{\mathbf{D}}\}$
$(\exists T_1, \ldots, T_m.d)^I$	$=$	$\{x \mid \exists y_1, \ldots, y_m : (x, y_1) \in T_1^I \wedge \ldots \wedge (x, y_m) \in T_m^I \wedge (y_1, \ldots, y_m) \in d^{\mathbf{D}}\}$
$(\leq n\, T)^I$	$=$	$\{x \mid \sharp\{y \mid (x, y) \in T^I\} \leq n\}$
$(\geq n\, T)^I$	$=$	$\{x \mid \sharp\{y \mid (x, y) \in T^I\} \geq n\}$
Semantics of Axioms		
$T \sqsubseteq U$		$T^I \subseteq U^I$
$T(a, b^{\mathsf{c}})$		$(a^I, (b^{\mathsf{c}})^I) \in T^I$
$\neg T(a, b^{\mathsf{c}})$		$(a^I, (b^{\mathsf{c}})^I) \notin T^I$
$a^{\mathsf{c}} \circ b^{\mathsf{c}}$		$(a^{\mathsf{c}})^I \circ (b^{\mathsf{c}})^I$ for $\circ \in \{\approx, \not\approx\}$

- $|\exists T_1, \ldots, T_m.d| = |\forall T_1, \ldots, T_m.d| = 2 + m$,
- $|\geq n\, T| = |\leq n\, T| = n + 2$.

The notion of positions is defined for the new $\mathcal{SHIQ}(\mathbf{D})$ constructs as follows:

- for $C = \bowtie T_1, \ldots, T_m.d$ with $\bowtie \in \{\exists, \forall\}$, $C|_i = T_i$ for $1 \leq i \leq m$, and $C|_{m+1} = d$;
- for $C = \bowtie n\, T$ with $\bowtie \in \{\leq, \geq\}$, $C|_1 = n$ and $C|_2 = T$.

Since the new constructs do not contain nested concepts, the notion of polarity carries over from $\mathcal{SHIQ}$ to $\mathcal{SHIQ}(\mathbf{D})$ without change.

Chapter 3

Deciding $\mathcal{SHIQ}$ by Basic Superposition

For a $\mathcal{SHIQ}$ knowledge base KB, our ultimate goal is to derive a disjunctive datalog program $\mathsf{DD}(KB)$ which is satisfiable if and only if KB is. The intuitive principle by which the equisatisfiability of KB and $\mathsf{DD}(KB)$ is achieved is relatively simple. Namely, if KB is unsatisfiable, this can be demonstrated by a refutation in some sound and complete calculus $\mathcal{I}$. If it is possible to simulate inferences of $\mathcal{I}$ in $\mathsf{DD}(KB)$, a refutation in KB by $\mathcal{I}$ can be reduced to a refutation in $\mathsf{DD}(KB)$. Conversely, if $\mathsf{DD}(KB)$ is unsatisfiable, there is a refutation in $\mathsf{DD}(KB)$. If it is possible to simulate inferences in $\mathsf{DD}(KB)$ by $\mathcal{I}$, then a refutation in $\mathsf{DD}(KB)$ can be reduced to a refutation in KB by $\mathcal{I}$. To summarize, if the simulation of inferences can be performed in both directions, $\mathsf{DD}(KB)$ and KB are equisatisfiable.

To obtain a sound, complete and terminating algorithm from the high-level idea outlined above, it is necessary to select the appropriate calculus $\mathcal{I}$, capable of effectively deciding satisfiability of KB. Positive disjunctive datalog is strongly related to clausal first-order logic. Intuitively, simulation of inferences using disjunctive datalog will be easier if $\mathcal{I}$ is a clausal refutation calculus. Hence, in the rest of this chapter we present a procedure for deciding satisfiability of $\mathcal{SHIQ}$ knowledge bases by basic superposition. The algorithm presented in this chapter is then used in Chapter 5 to obtain the desired reduction.

In Section 3.6 we give an overview of existing decision procedures for various decidable fragments of first-order logic based on clausal calculi. However, $\mathcal{SHIQ}$ cannot be directly embedded into any of these fragments. An exception is the two-variable fragment with counting quantifiers, but we find it difficult to use the decision procedure from [95] to simulate inference steps as sketched above. Hence, we designed a new, worst-case optimal decision procedure for $\mathcal{SHIQ}$, which itself has several novel aspects. It is well-known that the combination of inverse roles and counting quantifiers is difficult to handle algorithmically. On the model-theoretic side, it makes the logic lose the finite model property, and on the proof-theoretic side, tableau decision

procedures for such logics require sophisticated pair-wise blocking techniques to ensure termination [61]. It turns out that this combination makes a resolution-based decision procedure more complicated as well: contrary to most existing decision procedures (such as [43] or [105]), to decide $\mathcal{SHIQ}$ it is necessary to consider clauses containing terms of depth two. To block certain undesirable inferences with such clauses, a stronger calculus for equality than superposition [10] is needed. The solution we adopt is to use *basic superposition* [15].

3.1 Decision Procedure Overview

Before delving into the details, in this section we present a high-level overview of the principles we base our decision procedure upon.

Basic superposition alone unfortunately does not decide $\mathcal{SHIQ}$. A minor problem are transitivity axioms, which in a clausal representation require clauses without so-called *covering literals* — literals containing all variables of a clause [64]. As shown in [70], termination of resolution on such clauses is very difficult to achieve. To address this, in Section 3.2 we show how to eliminate transitivity axioms by polynomially encoding a $\mathcal{SHIQ}$ knowledge base KB into an equisatisfiable $\mathcal{ALCHIQ}$ knowledge base $\Omega(KB)$. After this initial transformation step, we focus on deciding satisfiability of $\mathcal{ALCHIQ}$ knowledge bases only.

A significantly more complex problem is that basic superposition alone decides only $\mathcal{ALCHIQ}^-$, where number restrictions are allowed only on roles not having subroles. Namely, the combination of role hierarchies, inverse roles and counting quantifiers may produce clauses which, during saturation by basic superposition, produce terms of ever increasing depth, so the saturation does not necessarily terminate. We address this problem in two stages.

In Section 3.3 we present a decision procedure for deciding satisfiability of an $\mathcal{ALCHIQ}^-$ knowledge base KB. First, we preprocess KB into a clausal representation as explained in Subsection 3.3.1. Let us denote the resulting set of closures with $\Xi(KB)$. It is not difficult to see that $\Xi(KB)$ will contain only closures of a certain syntactic form, as shown in Table 3.1. We then saturate $\Xi(KB)$ under $\mathcal{BS}_{DL}$ with eager application of redundancy elimination rules, where $\mathcal{BS}_{DL}$ is the $\mathcal{BS}$ calculus, parameterized as described in Definition 3.3.3. We denote the saturated set of closures by $\mathsf{Sat}(\Xi(KB))$. Since $\mathcal{BS}_{DL}$ is sound and complete [15], $\mathsf{Sat}(\Xi(KB))$ will contain the empty closure if and only if $\Xi(KB)$ is unsatisfiable. In order to obtain a decision procedure, we show that saturation always terminates. This is done in a proof-theoretic way as follows:

- We generalize the types of closures from Table 3.1 to so called $\mathcal{ALCHIQ}^-$-closures, which are presented in Table 3.2. In Lemma 3.3.4, we show that each closure occurring in $\Xi(KB)$ is an $\mathcal{ALCHIQ}^-$-closure.
- In Lemma 3.3.5, we show that in any $\mathcal{BS}_{DL}$-derivation starting from $\Xi(KB)$, each inference produces either an $\mathcal{ALCHIQ}^-$-closure, or a closure which is redundant (and may be deleted).

- In Lemma 3.3.8, we show that, for some finite knowledge base, the set of possible $\mathcal{ALCHIQ}^-$-closures occurring in any $\mathcal{BS}_{DL}$-derivation is finite.
- Termination is now a simple consequence of these two lemmata: in the worst case, one will derive all possible $\mathcal{ALCHIQ}^-$-closures, after which all further inferences are redundant. The bound on the size of the set of $\mathcal{ALCHIQ}^-$-closures gives us the complexity of the decision procedure, as demonstrated in Theorem 3.3.9.

To handle $\mathcal{ALCHIQ}$, in Section 3.4 we extend the basic superposition calculus with a new *decomposition* inference rule. This rule decomposes certain closures with undesirable terms into several simpler closures. We show that decomposition is sound and complete, and that it guarantees the termination of basic superposition for $\mathcal{ALCHIQ}$. It turns out that the decomposition rule is quite general and versatile; in Section 3.5 we use it to decide a slightly stronger logic $\mathcal{ALCHIQ}b$, which allows certain *safe* role expressions.

3.2 Eliminating Transitivity Axioms

In this section we show how to eliminate transitivity axioms from a $\mathcal{SHIQ}$ knowledge base KB, by transforming it polynomially into an $\mathcal{ALCHIQ}$ knowledge base $\Omega(KB)$. Since $\Omega(KB)$ is satisfiable exactly when KB is, without loss of generality we restrict our attention in the remaining sections to $\mathcal{ALCHIQ}$ knowledge bases. As one can easily see from Definition 3.2.2, for a $\mathcal{SHIQ}^-$ knowledge base KB, $\Omega(KB)$ is an $\mathcal{ALCHIQ}^-$ knowledge base. Therefore, we do not consider very simple roles explicitly in the definition of Ω.

The transformation presented here is similar to the one found in [114], where an algorithm for transforming $\mathcal{SHIQ}$ concepts to concepts in $\mathcal{ALCIQ}b$ logic was presented ($\mathcal{ALCIQ}b$ does not provide role hierarchy, but allows certain types of boolean operations on roles). Another similar transformation has been presented in [105], where it is demonstrated, among others, how $\mathrm{K4}_m$ (i.e. the multi-modal logic with transitive modalities) formulae can be encoded into K_m (i.e. the multi-modal logic without transitive modalities) formulae.

Definition 3.2.1. *For a $\mathcal{SHIQ}$ knowledge base KB, let* $\mathsf{clos}(KB)$ *denote the* concept closure *of KB, defined as the smallest set of concepts satisfying the following conditions:*

- *If* $C \sqsubseteq D \in KB_{\mathcal{T}}$, *then* $\mathsf{NNF}(\neg C \sqcup D) \in \mathsf{clos}(KB)$.
- *If* $C \equiv D \in KB_{\mathcal{T}}$, *then* $\mathsf{NNF}(\neg C \sqcup D) \in \mathsf{clos}(KB)$ *and* $\mathsf{NNF}(\neg D \sqcup C) \in \mathsf{clos}(KB)$.
- *If* $C(a) \in KB_{\mathcal{A}}$, *then* $\mathsf{NNF}(C) \in \mathsf{clos}(KB)$.
- *If* $C \in \mathsf{clos}(KB)$ *and* D *occurs in* C, *then* $D \in \mathsf{clos}(KB)$.
- *If* $\leq n\, R.C \in \mathsf{clos}(KB)$, *then* $\mathsf{NNF}(\neg C) \in \mathsf{clos}(KB)$.

- *If* $\forall R.C \in \mathsf{clos}(KB)$, $S \sqsubseteq^* R$, *and* $\mathsf{Trans}(S) \in KB_{\mathcal{R}}$, *then* $\forall S.C \in \mathsf{clos}(KB)$.

Notice that all concepts in $\mathsf{clos}(KB)$ are in NNF. We define next the operator Ω which encodes any $\mathcal{SHIQ}$ knowledge base KB into an equisatisfiable $\mathcal{ALCHIQ}$ knowledge base $\Omega(KB)$.

Definition 3.2.2. *For a* $\mathcal{SHIQ}$ *knowledge base* KB, *let* $\Omega(KB)$ *denote the following* $\mathcal{ALCHIQ}$ *knowledge base:*

- $\Omega(KB)_{\mathcal{R}}$ *is obtained from* $KB_{\mathcal{R}}$ *by removing all axioms* $\mathsf{Trans}(R)$,
- $\Omega(KB)_{\mathcal{T}}$ *is obtained by adding to* $KB_{\mathcal{T}}$ *the axiom* $\forall R.C \sqsubseteq \forall S.(\forall S.C)$ *for each concept* $\forall R.C \in \mathsf{clos}(KB)$ *and role* S *such that* $S \sqsubseteq^* R$ *and* $\mathsf{Trans}(S) \in KB_{\mathcal{R}}$,
- $\Omega(KB)_{\mathcal{A}} = KB_{\mathcal{A}}$.

Observe that, for any concept C, the number of subconcepts in $\mathsf{clos}(KB)$ is bounded by the number of subexpressions of C. Furthermore, for each concept from $\mathsf{clos}(KB)$, we may generate at most $|N_R|$ axioms in $\Omega(KB)_{\mathcal{T}}$. Hence, the encoding is polynomial in $|KB|$. Furthermore, it does not affect satisfiability, as we show next.

Theorem 3.2.3. *KB is satisfiable if and only if* $\Omega(KB)$ *is satisfiable.*

Proof. Out of convenience, we use the model-theoretic semantics of $\mathcal{SHIQ}$ in this proof.

($\Rightarrow$) Assume that I is a model of KB, but that I is not a model of $\Omega(KB)$, i.e. some axiom from $\Omega(KB)$ is not satisfied in I. Since $\Omega(KB)_{\mathcal{R}} \subseteq KB_{\mathcal{R}}$ and $KB_{\mathcal{T}} \subseteq \Omega(KB)_{\mathcal{T}}$, such an axiom must have been added in the second point of Definition 3.2.2. Hence, there is a domain element α such that $\alpha \in (\forall R.C)^I$, but $\alpha \notin (\forall S.(\forall S.C))^I$. There are two possibilities:

- There is no domain element β for which $(\alpha, \beta) \in S^I$. Then $\alpha \in (\forall S.X)^I$, regardless of X. Hence, $\alpha \in (\forall S.(\forall S.C))^I$ holds, which is a contradiction.
- There is a domain element β such that $(\alpha, \beta) \in S^I$. There are two possibilities:
  - If no domain element γ exists such that $(\beta, \gamma) \in S^I$, then $\beta \in (\forall S.C)^I$. Since this holds for any β, we have $\alpha \in (\forall S.(\forall S.C))^I$, which is a contradiction.
  - If there is γ such that $(\beta, \gamma) \in S^I$, by transitivity of S we have $(\alpha, \gamma) \in S^I$. Since $S^I \subseteq R^I$ and $\alpha \in (\forall R.C)^I$, we have $\gamma \in C^I$. Since this holds for any γ, we have $\beta \in (\forall S.C)^I$. Since this holds for any β, we have that $\alpha \in (\forall S.(\forall S.C))^I$, which is a contradiction.

Hence, I is a model of $\Omega(KB)$.

($\Leftarrow$) As explained in Section 2.7, without loss of generality we may focus only on knowledge bases with acyclic RBoxes. Let I be a model of $\Omega(KB)$, and I' an interpretation constructed from I as follows:

- $\triangle^{I'} = \triangle^{I}$.
- For each individual a, $a^{I'} = a^{I}$.
- For each atomic concept $A \in \mathsf{clos}(KB)$, $A^{I'} = A^{I}$.
- If $\mathsf{Trans}(R) \in KB_{\mathcal{R}}$, $R^{I'} = (R^{I})^{+}$.
- If $\mathsf{Trans}(R) \notin KB_{\mathcal{R}}$, $R^{I'} = R^{I} \cup \bigcup_{S \sqsubseteq^{*} R, S \neq R} S^{I'}$.

Since we may assume that $KB_{\mathcal{R}}$ is acyclic, the above induction is well-defined. We will now show that I' satisfies every RBox axiom of KB. By construction I' satisfies all transitivity axioms in $KB_{\mathcal{R}}$. Furthermore, I' satisfies each inclusion axiom in $KB_{\mathcal{R}}$: if R is not transitive, this is obvious from the construction; otherwise, this follows from the fact that $A^{+} \cup B^{+} \subseteq (A \cup B)^{+}$ for any A and B. Furthermore, for any role R we have $R^{I} \subseteq R^{I'}$, and, if R is simple, then $R^{I'} = R^{I}$.

For concepts C and D from $\mathsf{clos}(KB)$, let $C \lessdot D$ if and only if C or $\mathsf{NNF}(\neg C)$ occurs in D. We now show by induction on $\lessdot$ that, for each $D \in \mathsf{clos}(KB)$, $D^{I} \subseteq D^{I'}$. The relation $\lessdot$ is obviously acyclic, so the induction is well-founded. For the base case when D is an atomic concept A or a negation of an atomic concept $\neg A$, the claim follows immediately from the definition of I'. For the induction step we examine each possible form D might have:

- $D = C_1 \sqcap C_2$. Assume for some α we have $\alpha \in (C_1 \sqcap C_2)^{I}$. Then $\alpha \in C_1^{I}$ and $\alpha \in C_2^{I}$. By the induction hypothesis, $\alpha \in C_1^{I'}$ and $\alpha \in C_2^{I'}$, so $\alpha \in (C_1 \sqcap C_2)^{I'}$.
- $D = C_1 \sqcup C_2$. Assume for some α we have $\alpha \in (C_1 \sqcup C_2)^{I}$. Then either $\alpha \in C_1^{I}$, so by the induction hypothesis $\alpha \in C_1^{I'}$, or $\alpha \in C_2^{I}$, so by the induction hypothesis $\alpha \in C_2^{I'}$. Either way, $\alpha \in (C_1 \sqcup C_2)^{I'}$.
- $D = \exists R.C$. Assume for some α we have $\alpha \in (\exists R.C)^{I}$. Then there is a β such that $(\alpha, \beta) \in R^{I}$ and $\beta \in C^{I}$, so by the induction hypothesis $\beta \in C^{I'}$. Since $R^{I} \subseteq R^{I'}$, we have $(\alpha, \beta) \in R^{I'}$, so $\alpha \in (\exists R.C)^{I'}$.
- $D = \forall R.C$. Assume that $\alpha \in (\forall R.C)^{I}$. If there is no object β such that $(\alpha, \beta) \in R^{I'}$, then $\alpha \in (\forall R.C)^{I'}$. Otherwise, assume there is such a β. There are two possibilities:
  - $(\alpha, \beta) \in R^{I}$. Then $\beta \in C^{I}$, so by the induction hypothesis $\beta \in C^{I'}$.

– $(\alpha, \beta) \notin R^I$. Then there must be a role $S \sqsubseteq^* R$ with $\mathsf{Trans}(S) \in KB_{\mathcal{R}}$, and a path $(\alpha, \gamma_1) \in S^I$, $(\gamma_1, \gamma_2) \in S^I$, ..., $(\gamma_{n-1}, \beta) \in S^I$, $n > 1$. But then $\forall R.C \sqsubseteq \forall S.(\forall S.C) \in \Omega(KB)_{\mathcal{T}}$, so $\alpha \in (\forall S.(\forall S.C))^I$ and $\gamma_1 \in (\forall S.C)^I$. Furthermore, $\forall S.C \sqsubseteq \forall S.(\forall S.C) \in \Omega(KB)_{\mathcal{T}}$, so $\gamma_i \in (\forall S.C)^I$ for $1 < i < n$. For $n - 1$ we have $\beta \in C^I$, so by the induction hypothesis $\beta \in C^{I'}$.

Since for any β, in both cases we have that $\beta \in C^{I'}$, we have $\alpha \in (\forall R.C)^{I'}$.

- $D = \geq n\, R.C$. Assume that $\alpha \in (\geq n\, R.C)^I$. Then there are at least n distinct domain elements β_i such that $(\alpha, \beta_i) \in R^I$ and $\beta_i \in C^I$, so by the induction hypothesis $\beta_i \in C^{I'}$. Since $R^I \subseteq R^{I'}$, we have $\alpha \in (\geq n\, R.C)^{I'}$.

- $D = \leq n\, R.C$. Since R is simple, $R^I = R^{I'}$. Let $E = \mathsf{NNF}(\neg C)$. Assume now that $\alpha \in (\leq n\, R.C)^I$, but $\alpha \notin (\leq n\, R.C)^{I'}$. Then a β must exist such that $(\alpha, \beta) \in R^I$, $\beta \notin C^I$, $\beta \in C^{I'}$, $\beta \in E^I$ and $\beta \notin E^{I'}$. However, since $E \in \mathsf{clos}(KB)$, by induction hypothesis we have $\beta \in E^{I'}$, which is a contradiction. Hence, $\alpha \in (\leq n\, R.C)^{I'}$.

Now it is obvious that any ABox axiom of the form $C(a)$ from KB is satisfied in I': since I is a model of $\Omega(KB)$, we have $a^I \in C^I$, but since $C^I \subseteq C^{I'}$, we have $a^I \in C^{I'}$. Also, any ABox axiom of the form $R(a, b)$ from KB is satisfied in I': since I is a model of $\Omega(KB)$, we have $(a^I, b^I) \in R^I$, but since $R^I \subseteq R^{I'}$, we have $(a^I, b^I) \in R^{I'}$. For an ABox axiom of the form $\neg S(a, b)$, S must be a simple role so $S^I = S^{I'}$ and $(a^I, b^I) \notin S^I$ implies $(a^I, b^I) \notin S^{I'}$. Finally, any TBox axiom of the form $C \subseteq D$ from KB is satisfied in I': since I is a model of $\Omega(KB)$, we have that $\triangle^I \subseteq (\neg C \sqcup D)^I$, but since $(\neg C \sqcup D)^I \subseteq (\neg C \sqcup D)^{I'}$, we have $\triangle^{I'} \subseteq (\neg C \sqcup D)^{I'}$. Similar arguments hold for any TBox axiom of the form $C \equiv D$. □

Notice that for α of the form $(\neg)A(a)$ with A an atomic concept, or of the form $(\neg)R(a, b)$ with R a simple role, $\Omega(KB \cup \{\alpha\}) = \Omega(KB) \cup \{\alpha\}$, so $KB \models \alpha$ if and only if $\Omega(KB) \models \alpha$. Hence, transitivity axioms can be eliminated once, and thus obtained knowledge base can be used for query answering. Unfortunately, the models of KB and $\Omega(KB)$ may differ in the interpretation of complex roles. Therefore, $\Omega(KB)$ cannot be used to answer such ground queries where R is a complex role. This is why we restrict negative ground role atoms in Definition 2.7.1 of $\mathcal{SHIQ}$ knowledge bases to simple roles only.

3.3 Deciding $\mathcal{ALCHIQ}^-$

In this section we present in detail each step of the algorithm for deciding satisfiability of an $\mathcal{ALCHIQ}^-$ knowledge base KB, which we outlined in Section 3.1.

3.3.1 Preprocessing

The first step in deciding satisfiability of KB is to transform it into clausal form. Straightforward transformation of $\pi(KB)$ into disjunctive normal form has two significant drawbacks. Firstly, the structure of formulae would be destroyed. Secondly, the usual transformation into clausal normal form might increase the size of the closure set exponentially. To overcome these drawbacks, we first apply the *structural transformation* [93, 86, 9], also known as *renaming*. Intuitively, for some first-order formula φ, the structural transformation introduces a new name for each subformula of φ. Thus, the original formula structure is preserved and, since the number of subformulae of φ is linear in the size of φ, the exponential blowup is avoided.

In the rest, we assume without loss of generality that all ABox concept membership axioms in KB are expressed using atomic concepts: for each membership axiom $C(a)$ where C is not atomic, one may introduce a new atomic concept A_C, add the axiom $A_C \sqsubseteq C$ to the TBox and replace $C(a)$ with $A_C(a)$. Such a transformation is obviously polynomial, and we call such knowledge bases *extensionally reduced*.

Definition 3.3.1. *Let C be a concept, and $\Lambda(C)$ the set of positions $p \neq \epsilon$ of subconcepts of C such that $C|_p$ is neither an atomic concept, nor a negation of an atomic concept, and for all positions q below p, $C|_q$ is either an atomic concept or a negation of an atomic concept.* $\mathsf{Def}(C)$ *is defined recursively as follows:*

$$\mathsf{Def}(C) = \begin{cases} \{C\} & \text{if } \Lambda(C) = \emptyset \\ \{\neg Q \sqcup C|_p\} \cup \mathsf{Def}(C[Q]_p) & \text{if } p \in \Lambda(C) \text{ and } \mathsf{pol}(C,p) = 1 \\ \{Q \sqcup \neg C|_p\} \cup \mathsf{Def}(C[Q]_p) & \text{if } p \in \Lambda(C) \text{ and } \mathsf{pol}(C,p) = -1 \end{cases}$$

where Q is a new globally unique atomic concept. Furthermore, let

$$\mathsf{Cls}(\mathsf{Def}(C)) = \bigcup_{D \in \mathsf{Def}(C)} \mathsf{Cls}(\forall x : \pi_y(D, x))$$

For an $\mathcal{ALCHIQ}$ knowledge base KB, $\Xi(KB)$ is the smallest set of closures satisfying the following conditions:

- *For each abstract role name $R \in N_{R_a}$, $\mathsf{Cls}(\pi(R)) \subseteq \Xi(KB)$.*
- *For each RBox or ABox axiom α in KB, $\mathsf{Cls}(\pi(\alpha)) \subseteq \Xi(KB)$.*
- *For each TBox axiom $C \sqsubseteq D$ in KB, $\mathsf{Cls}(\mathsf{Def}(\neg C \sqcup D)) \subseteq \Xi(KB)$.*
- *For each TBox axiom $C \equiv D$ in KB, $\mathsf{Cls}(\mathsf{Def}(\neg C \sqcup D)) \subseteq \Xi(KB)$ and $\mathsf{Cls}(\mathsf{Def}(\neg D \sqcup C)) \subseteq \Xi(KB)$.*

In the above definition, for $C = C_1 \sqcup \ldots \sqcup C_n$, one can safely omit the positions $1, \ldots, n$ in $\Lambda(C)$, since this reduces the number of closures generated. For example, the negation normal form of the axiom $\neg C \sqcap \neg D \sqsubseteq \exists R.\top$ is $C \sqcup D \sqcup \exists R.\top$, which can be readily transformed into a closure $C(x) \vee D(x) \vee R(x, f(x))$, without introducing a new name for the subconcept $\exists R.\top$.

Table 3.1: Closure Types after Preprocessing

1	$\neg R(x,y) \vee \mathsf{Inv}(R)(y,x)$
2	$\neg R(x,y) \vee S(x,y)$
3	$\bigvee(\neg)C_i(x) \vee R(x,f(x))$
4	$\bigvee(\neg)C_i(x) \vee (\neg)D(f(x))$
5	$\bigvee(\neg)C_i(x) \vee f_i(x) \not\approx f_j(x)$
6	$\bigvee(\neg)C_i(x)$
7	$\bigvee(\neg)C_i(x) \vee \bigvee_{i=1}^{n} \neg R(x,y_i) \vee \bigvee_{i=1}^{n} D(y_i) \vee \bigvee_{i,j=1;j>i}^{n} y_i \approx y_j$
8	$(\neg)C(a)$
8	$(\neg)R(a,b)$
10	$a \approx b$
11	$a \not\approx b$

Lemma 3.3.2. *Let KB be an $\mathcal{ALCHIQ}$ knowledge base. Then KB is satisfiable if and only if $\Xi(KB)$ is satisfiable. Furthermore, $\Xi(KB)$ can be computed in time polynomial in $|KB|$ for unary coding of numbers in input.*

Proof. Let $\psi_C = \bigwedge_{D \in \mathsf{Def}(C)} \forall x : \pi_y(C,x)$. It is easy to see that ψ_C is actually the definitional normal form of $\varphi = \forall x : \pi_y(C,x)$ with respect to the set of positions of all non-atomic subformulae of φ. By the definition of π and Cls, and since transformation into definitional normal form does not affect satisfiability, it is easy to see that KB and $\Xi(KB)$ are equisatisfiable. The inductive step of $\mathsf{Def}(C)$ is applied at most once for each subconcept of C, so the number of new concepts Q introduced by Def is polynomial in $|C|$, and $\mathsf{Def}(C)$ can be computed in polynomial time. For each $D \in \mathsf{Def}(C)$, the number of function symbols f introduced by the skolemization of $\forall x : \pi_y(D,x)$ is bounded by the maximal number occurring in a number restriction in D. For unary coding of numbers, f is linear in $|D|$, so $\mathsf{Cls}(D)$ can obviously be computed in time polynomial in $|D|$, thus implying the claim of the lemma. □

Using binary coding, a number n can be represented with $\lceil \log_2 n \rceil$ bits. In this case, the number of function symbols introduced by skolemization is exponential in the size of the input, so translation into first-order logic would incur exponential blowup.

By definition of π from Table 2.1, it is easy to see that all closures obtained by this transformation share some common syntactic properties. Table 3.1 lists the types of closures that $\Xi(KB)$ may contain.

3.3.2 Parameters for Basic Superposition

For the following definition it is necessary to keep in mind that literals $A(t_1,\ldots,t_n)$ are encoded as $A(t_1,\ldots,t_n) \approx \top$, as discussed in Section 2.4. Due to this encoding,

predicate symbols become function symbols, so atoms can be compared using a term ordering given below.

Definition 3.3.3. *Let $\mathcal{BS}_{DL}$ denote the $\mathcal{BS}$ calculus parameterized as follows:*

- *The term ordering $\succ$ is a lexicographic path ordering induced over a total precedence $>_P$ over function, constant and predicate symbols, such that, for any function symbol f, constant symbol c, and predicate symbol p, $f >_P c >_P p >_P \top$.*
- *The selection function selects in each closure $C \cdot \sigma$ every negative binary literal.*

In $\mathcal{BS}_{DL}$, we need to compare terms and literals only in closures of types 3–6 and 8 from Table 3.2. It is easy to see that, since LPOs are total on ground terms, and terms in closures of type 3–6 and 8 have at most one variable, any LPO is total on non-ground terms from these closures. In this case, one can use a more direct definition of the literal ordering. We associate with each literal $L = s \circ t$, $\circ \in \{\approx, \not\approx\}$, the triple $c_L = (\max(s,t), p_L, \min(s,t))$, where $\max(s,t)$ is the bigger of the two terms, p_L is 1 if $\circ$ is $\not\approx$, and 0 otherwise, and $\min(s,t)$ is the smaller of the two terms. Then $L_1 \succ L_2$ if and only if $c_{L_1} \succ c_{L_2}$, where c_{L_i} are compared lexicographically. An LPO is used to compare the first and the third position of c_L, where for the second position we take $1 \succ 0$. It is easy to see that, since $\succ$ is total on terms, this definition is equivalent to the one based on two-fold multiset extension, given in Section 2.4.

Ordering and selection constraints in $\mathcal{BS}$ are checked *a posteriori*, that is, after computing the unifier. This is more general, since some terms may be comparable only after unification. For example, $s = f(x)$ and $t = y$ are not comparable using an LPO. However, for $\sigma = \{x \mapsto a, y \mapsto g(f(a))\}$, we have $t\sigma \succ s\sigma$. The drawback is that the unifier is often computed in vain, just to determine that constraints are not satisfied. However, LPOs are total on terms from closures 3–6 and 8, so we may check ordering and selection constraints *a priori*, that is, before computing the unifier. Namely, if s and t are two terms to be compared, they are either both ground or both have the same, single free variable, so they can always be compared before computing σ. Also, if $s \succ t$, then $s\sigma \succ t\sigma$ for any substitution σ.

3.3.3 Closure of $\mathcal{ALCHIQ}^-$-closures under Inferences

We now generalize types of closures from Table 3.1 to so-called $\mathcal{ALCHIQ}^-$-closures presented in Table 3.2. For a term t, with $\mathbf{P}(t)$ we denote a possibly empty disjunction of the form $(\neg)P_1(t) \vee \ldots \vee (\neg)P_n(t)$. With $\mathbf{P}(\mathbf{f}(x))$ we denote a possibly empty disjunction of the form $\mathbf{P_1}(f_1(x)) \vee \ldots \vee \mathbf{P_n}(f_m(x))$. Notice that this definition allows each $\mathbf{P_i}(f_i(x))$ to contain positive and negative literals. With $\langle t \rangle$ we denote that the term t may, but need not be marked. In all closure types, some of the disjuncts may be empty. Finally, with $\approx/\not\approx$ we denote a positive or a negative equality literal.

Lemma 3.3.4. *Each closure from $\Xi(KB)$ is of exactly one of the types from Table 3.2. Furthermore, for each function symbol f occurring in $\Xi(KB)$, there is exactly one*

Table 3.2: Types of $\mathcal{ALCHIQ}^-$-closures

1	$\neg R(x,y) \vee \mathsf{Inv}(R)(y,x)$
2	$\neg R(x,y) \vee S(x,y)$
3	$\mathbf{P^f}(x) \vee R(x, \langle f(x) \rangle)$
4	$\mathbf{P^f}(x) \vee R([f(x)], x)$
5	$\mathbf{P_1}(x) \vee \mathbf{P_2}(\langle \mathbf{f}(x) \rangle) \vee \bigvee \langle f_i(x) \rangle \approx / \not\approx \langle f_j(x) \rangle$
6	$\mathbf{P_1}(x) \vee \mathbf{P_2}([g(x)]) \vee \mathbf{P_3}(\langle \mathbf{f}([g(x)]) \rangle) \vee \bigvee \langle t_i \rangle \approx / \not\approx \langle t_j \rangle$ where t_i and t_j are either of the form $f([g(x)])$ or of the form x
7	$\mathbf{P_1}(x) \vee \bigvee \neg R(x, y_i) \vee \mathbf{P_2}(\mathbf{y}) \vee \bigvee y_i \approx y_j$
8	$\mathbf{R}(\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle) \vee \mathbf{P}(\langle \mathbf{t} \rangle) \vee \bigvee \langle t_i \rangle \approx / \not\approx \langle t_j \rangle$ where t, t_i and t_j are either some constant b or a functional term $f_i([a])$

Conditions:

- (i): In any term $f(t)$, the inner term t occurs marked.
- (ii): In all positive equality literals with at least one function symbol, both sides are marked.
- (iii): Any closure containing a term $f(t)$ contains $\mathbf{P^f}(t)$ as well.
- (iv): In a literal $[f_i(t)] \approx [f_j(t)]$, $\mathsf{role}(f_i) = \mathsf{role}(f_j)$.
- (v): In a literal $[f(g(x))] \approx x$, $\mathsf{role}(f) = \mathsf{Inv}(\mathsf{role}(g))$.
- (vi): For each $[f_i(a)] \approx [b]$ a *witness* closure of the form $R(\langle a \rangle, \langle b \rangle) \vee D$ exists, with $\mathsf{role}(f_i) = R$, D does not contain functional terms or negative binary literals, and is contained in this closure.

*closure of type 3 containing $f(x)$ unmarked; this closure is called the R^f-*generator*, the disjunction $\mathbf{P^f}(x)$ is called the f-*support*, and R is called the* designated role *for f and is denoted as $\mathsf{role}(f)$.*

Proof. The first claim follows trivially from the definition of $\Xi(KB)$. For the second claim, observe that each closure of type 3 is generated by skolemizing an existentially quantified subformula. Since each skolemization introduces a fresh function symbol, this symbol will be associated with exactly one closure of type 3. □

We now prove the core result that our decision procedure is based upon.

Lemma 3.3.5. *Let $\Xi(KB) = N_0, \ldots, N_i \cup \{C\}$ be a $\mathcal{BS}_{DL}$-derivation, where C is the conclusion derived from premises in N_i. Then C is either an $\mathcal{ALCHIQ}^-$-closure or it is redundant in N_i.*

Proof. We first prove the property (max), determining which literals can be maximal in closures of types 3, 4, 5, 6 and 8 under ordering and selection function of $\mathcal{BS}_{DL}$:

- In a closure of type 3, the literal $R(x, \langle f(x) \rangle)$ is always maximal.

- In a closure of type 4, the literal $R([f(x)], x)$ is always maximal.
- In a closure of type 5, the literal of the form $(\neg)P(x)$ can be maximal only if the closure does not contain a term of the form $f(x)$.
- In a closure of type 6, only a literal containing a term of the form $f([g(x)])$ can be maximal.
- In a closure of type 8, a literal of the form $(\neg)R(a, b)$, $(\neg)P(a)$, $a \approx b$, or $a \not\approx b$ can be maximal only if the closure does not contain a function symbol.

For closures of type 3 and 4, the claims follow directly from the properties of the precedence $>_P$ from Definition 3.3.3 and the definition of LPO. Furthermore, for any term t, function symbol f, and predicate symbol P, since $f >_P P$ and $f(t) \succ t$, we have $f(t) \succ P(t)$. For a closure of type 5, we have $P'(f(x)) \succ f(x) \succ P(x)$, so $P(x)$ may be maximal only if the closure does not contain a term of the form $f(x)$. For a closure of type 6, we have $P''(f(g(x))) \succ f(g(x)) \succ P'(g(x)) \succ g(x) \succ P(x)$, so only a literal containing a term of the form $f(g(x))$ may be maximal. Finally, for any function symbol f, constants a, b, and c, unary predicate symbol P and a binary predicate symbol R, by the properties of $>_P$ we have $f(a) \succ P(b)$, $f(a) \succ R(b, c)$ and $f(a) \succ b$. Hence, a literal not containing function symbols can be maximal only if a closure it occurs in does not contain a function symbol.

We now prove the lemma by induction on the derivation length. By Lemma 3.3.4, N_0 contains only $\mathcal{ALCHIQ}^-$-closures, so the induction base holds. For the induction step, we examine all possible applications of inference rules of $\mathcal{BS}_{DL}$ to closures in N_i. We show first that the conclusion has the structure of an $\mathcal{ALCHIQ}^-$-closure, and later show that conditions (i)–(vi) hold for it.

Inferences with closures of types 1 and 2. Since negative binary literals are always selected and superposition into variables is not necessary, closures of type 1 and 2 can participate only as negative premises in resolution inferences with closures of types 3, 4 and 8. Obviously, the unifier binds the variables x and y to corresponding terms in the positive premise, and the result is of type 3, 4 or 8.

Inferences between closures of types 5, 6 and 8. Consider any inference between closures of types 5 or 6 with free variables x and x', respectively. Since the term $g(x)$ in some $f([g(x)])$ is always marked, terms can be unified only at their root position. The following pairs of terms from premises can be unifiable:

- x and x'; $f(x)$ and $f(x')$; or $f([g(x)])$ and $f([g(x')])$. The unifier is $\sigma = \{x \mapsto x'\}$, and the conclusion is obviously a closure of type 5 or 6. Notice that superposition from $f(g(x)) \approx x$ into $f(g(x')) \not\approx x'$ results in $x' \not\approx x'$, which can eagerly be eliminated by reflexivity resolution.

- x and $g(x')$; or $f(x)$ and $f([g(x')])$. The unifier is $\sigma = \{x \mapsto g(x')\}$, and the conclusion is also a closure of type 5 or 6.

Inferences between closures of type 6 and 8 are not possible since a term of the form $f(g(x))$ does not unify with a term of the form a or $f(a)$. For closures of type 5 and 8, the unifier may be $\sigma = \{x \mapsto a\}$ or $\sigma = \{x \mapsto f(a)\}$, and the conclusion is of type 8. Inferences between closures of type 8 have an empty unifier and the conclusion is trivially of type 8.

Inferences with a closure of type 7. Since all binary literals are always selected, a closure of type 7 can participate only in a hyperresolution inference as the main premise C. The side premises can have the maximal literals of the form $R(x_i, \langle f_i(x_i)\rangle)$, $R([g_i(x_i)], x_i)$ or $R(\langle a\rangle, \langle b_i\rangle)$. These combinations are possible:

- Assume there are two (or more) side premises with the maximal literal of the form $R([g_i(x_i)], x_i)$. Without loss of generality these premises may be assigned indices 1 and 2. Since $g_1(x_1)$ and $g_2(x_2)$ must be unified with x, σ contains mappings $y_1 \mapsto x_1, y_2 \mapsto x_1$ and $x_2 \mapsto x_1$. Since C contains a literal $y_i \approx y_j$ for each pair of i and j, the conclusion contains $x_1 \approx x_1$ and is a tautology.

- Assume the first side premise has the maximal literal of the form $R([g(x')], x')$. Since $g(x')$ does not unify with a constant, no side premise may be of type 8. The unifier σ is of the form $x \mapsto g(x'), x_i \mapsto g(x'), y_1 \mapsto x', y_i \mapsto f_i(g(x')), 2 \leq i \leq n$. If $n = 1$, the conclusion is of type 5; otherwise it is of type 6.

- Assume all side premises have the maximal literals of the form $R(x_i, \langle f_i(x_i)\rangle)$. The unifier σ is of the form $x_i \mapsto x, y_i \mapsto f_i(x)$ and the conclusion is of type 5.

- Assume some side premises have the maximal literal of the form $R(x_i, \langle f_i(x_i)\rangle)$ for $1 \leq i \leq k$, and $R(\langle a\rangle, \langle b_i\rangle)$ for $k < i \leq n$ (all ground premises must have the same first argument in the maximal literal since all these arguments should unify with x). The unifier σ contains mappings of the form $x \mapsto a, x_i \mapsto a, y_i \mapsto f_i(a)$ for $1 \leq i \leq k$ and $y_i \mapsto b_i$ for $k + 1 \leq i \leq n$. The conclusion is of type 8.

Superposition into a closure of type 3. The only remaining possible inference is superposition into a closure of type 3 with the free variable x'. By condition (max), $(w \approx v) \cdot \rho$ can only be the literal $R(x', f(x'))$ with R being the designated role for f. There are these possibilities:

- $(C \vee s \approx t) \cdot \rho$ is a closure of type 5, 6 or 8 with $(s \approx t) \cdot \rho$ of the form $[f(\iota)] \approx [g(\iota)]$. The unifier σ is $\{x' \mapsto \iota\}$, so the conclusion is $S = \mathbf{P^f}([\iota]) \vee R([\iota], [g(\iota)]) \vee C \cdot \rho$, where $\mathbf{P^g}([\iota]) \subseteq C \cdot \rho$. By Condition (iv), the R^g-generator $P^g(y) \vee R(y, g(y))$ exists, so it subsumes S by the substitution $\{y \mapsto \iota\}$.

- $(C \vee s \approx t) \cdot \rho$ is a closure of type 6 with $(s \approx t) \cdot \rho$ of the form $[f(g(x))] \approx x$. The unifier σ is $\{x' \mapsto g(x)\}$, so the conclusion is $S = \mathbf{P^f}([g(x)]) \vee R([g(x)], x) \vee C \cdot \rho$, where $\mathbf{P^g}(x) \subseteq C \cdot \rho$. By Condition (v), the closure $\mathbf{P^g}(y) \vee R([g(y)], y)$ exists, so it subsumes S by the substitution $\{y \mapsto x\}$.

- $(C \vee s \approx t) \cdot \rho$ is a closure of type 8 with $(s \approx t) \cdot \rho$ of the form $[f(a)] \approx [b]$. The unifier σ is $\{x' \mapsto a\}$, so the conclusion is $S = \mathbf{P^f}([a]) \vee R([a], [b]) \vee C \cdot \rho$. By Condition (vi), a witness of the form $R(\langle a \rangle, \langle b \rangle) \vee D$, where $D \subseteq C \cdot \rho$, exists, and it subsumes S by the empty substitution. If the witness has been subsumed by some other closure, then, since the subsumption relation is transitive, this other closure subsumes the conclusion.

In all cases, the superposition conclusion is subsumed by an existing closure, so a superposition into a closure of type 3 is always redundant.

Equality factoring. Ordering constraints allow optimizing the application of equality factoring. Only closures of types 5, 6 or 8 are candidates for equality factoring. The premise has the form $(C \vee s \approx t \vee s' \approx t') \cdot \rho$, where $s\rho \approx t\rho$ is maximal with respect to $C\rho \vee s'\rho \approx t'\rho$, $t\rho \not\succeq s\rho$, and $t'\rho \not\succeq s'\rho$. The unifier σ is always empty. If we assume that simplification by duplicate literal elimination is applied eagerly, we safely conclude that $(s \approx t) \cdot \rho$ is strictly maximal, so t' and t cannot be $\top$. Hence, terms $s\rho$, $t\rho$, $s'\rho$ and $t'\rho$ are either all ground or all contain the same free variable x. The ordering $\succ$ is total on such terms, so we may rewrite ordering constraints as $t\rho \prec s\rho$ and $t'\rho \prec s'\rho$. By the fact that $s\rho \approx t\rho$ is strictly maximal, we conclude that $t\rho \succ t'\rho$.

Consider now the case where all equalities involved in the inference are marked, so s, t, s' and t' are variables. This is the case for all closures of type 5 and 6, and some closures of type 8. The conclusion has then the form $(C \vee t \not\approx t' \vee s' \approx t') \cdot \rho$, where t is a variable and $t\rho \succ t'\rho$. Thus, the conclusion is a basic tautology and is redundant.

Hence, provided that duplicate literal elimination is applied eagerly, equality factoring is redundant for all closures, apart from closures of type 8, where it can be applied to equalities of the form $\langle a \rangle \approx \langle b \rangle$ with at least one non-marked term. Depending on the marking, equality factoring either yields a basic tautology which is redundant or a closure of type 8. Notice that a closure of type 8 might contain disjunctions of the form $R([a], b) \vee R(a, [b])$, to which duplicate literal removal does not apply directly due to incompatible markers. However, we assume that in such a case the markers are eagerly retracted, i.e. the disjunction is converted into $R(a, b) \vee R(a, b)$ which is then collapsed into $R(a, b)$.

Reflexivity resolution. Reflexivity resolution can only be applied to a closure of type 5, 6 and 8 with the empty unifier, so it produces a closure of type 5, 6 or 8. Since the unifier is always empty, the conclusion subsumes the premise, so this inference should be applied eagerly.

Conditions. From the above case analysis, one may observe that the following property (*uni*) hold: closures of type 3, 4 and 5 with a free variable x participate in inferences only with a unifier containing mappings $x \mapsto x'$, $x \mapsto g(x')$, $x \mapsto f(g(x'))$ or $x \mapsto a$. We now show that all conditions from Table 3.2 hold for each non-redundant inference conclusion.

Condition (*i*): If a closure C satisfies Condition (*i*), by the property (*uni*) the unifier σ may only instantiate x. Hence, $C\sigma$ will satisfy (*i*) as well. Since no inference removes markers from functional terms, Condition (*i*) holds for any conclusion.

Condition (*ii*): All positive equality literals with at least one function symbol are generated by hyperresolution with a closure of type 7, so all terms in positive equalities in the conclusion are marked. Since no inference removes markers from the roots of the terms occurring in equalities, Condition (*ii*) holds for any conclusion.

Condition (*iii*): If a closure C contains $f([t])$ and satisfies Condition (*iii*), by the property (*max*) literals from $\mathbf{P^f}([t])$ cannot participate in an inference. Furthermore, by the property (*uni*), any variable x is instantiated simultaneously in $f([t])$ and $P^f([t])$. Since no inference adds new functional terms in the conclusion, Condition (*iii*) holds for any conclusion.

Conditions (*iv*) and (*v*): All equality literals with functional terms are generated by hyperresolution with the nucleus C of type 7. Since the role R occurring in C is very simple, a closure of type 3 or 4 cannot be resolved with a closure of type 2. Hence, for all electrons of the form $\mathbf{P^f}(x) \vee R(x, \langle f(x)\rangle)$ we have $\mathsf{role}(f) = R$, and for an electron of the form $\mathbf{P^g}(x) \vee R([g(x)], x)$ we have $\mathsf{role}(g) = \mathsf{Inv}(R)$. Hence, Conditions (*iv*) and (*v*) are satisfied for each conclusion of a hyperresolution inference. Furthermore, by Condition (*ii*) superposition into equality literals with functional terms is not possible, and in any literal $[f_i(x)] \approx [f_j(x)]$ the variable x is instantiated simultaneously. Hence, Conditions (*iv*) and (*v*) hold for each conclusion of any inference.

Condition (*vi*): All literals of the form $[f(a)] \approx [b]$ are generated by hyperresolution involving an electron E_1 of type 8 with the maximal literal $R(\langle a\rangle, \langle b\rangle)$ and an electron E_2 of type $\mathbf{P^f}(x) \vee R(x, \langle f(x)\rangle)$. Since R occurring in such C must be very simple, a closure of type 8 cannot be resolved with a closure of type 2, so $\mathsf{role}(f) = R$. Since the literal $R(\langle a\rangle, \langle b\rangle)$ is maximal in E_1, by the property (*max*) no literal from E_1 contains a functional term, and E_2 does not contain a negative binary literal. Hence, the conclusion satisfies Condition (*vi*). Assume now that Condition (*vi*) holds for some closure C, where D is a witness of $[f(a)] \approx [b]$. Since no literal from D is functional or is a negative binary literal, by the property (*max*) no literal from C occurring in D may participate in an inference, so all literals are present in the conclusion. Finally, both sides of the all equality literals containing functional terms are marked, so each equality literal with functional terms derived in an inference must always occur in some of the premises. Hence, Condition (*vi*) holds for each conclusion. □

The following corollary can be easily shown by inspecting the proof of Lemma 3.3.5:

Corollary 3.3.6. *If a closure of type 8 participates in a $\mathcal{BS}_{DL}$ inference in a derivation from Lemma 3.3.5, the unifier σ contains only ground mappings of the form $x \mapsto a$*

and $x \mapsto f(b)$, and the conclusion is a closure of type 8. Furthermore, a closure of type 8 cannot participate in an inference with a closure of type 4 or 6.

The following corollary is useful for optimizing certain algorithms we present later.

Corollary 3.3.7. *Let KB be an $\mathcal{ALCHIQ}^-$ knowledge base, containing neither at-most number restrictions occurring under positive, nor at-least number restrictions occurring under negative polarity. Then, in a saturation of $\Xi(KB)$ by $\mathcal{BS}_{DL}$, closures of type 8 do not contain functional terms. Furthermore, a closure of type 8 can participate in an inference only with closures not containing functional terms.*

Proof. For a KB as in the corollary, with (*) we denote the fact that each closure of type 7 in $\Xi(KB)$ contains exactly one literal $\neg R(x, y)$ and does not contain equality literals. Now the claim of the corollary can be shown by induction on the derivation length. The base case is obvious, since all ABox closures in $\Xi(KB) = N_0$ are of type from Table 3.1, so they do not contain functional terms. For the induction step, we consider all inferences generating a closure of type 8 in N_{i+1}. A closure with an equality literal containing a functional term might be generated only by a hyperresolution with a closure of type 7 containing equality literals, but this is not possible by (*). A literal $(\neg)C([g(a)])$ might be derived by hyperresolving a closure of type 7 with an electron of type 3 and of type 8, but this is again not possible by (*). The only remaining possibility to derive a literal $(\neg)C([g(a)])$ is by superposition from $[f(a)] \approx [g(a)]$ into $(\neg)C(f(x))$, but this is not possible, since N_i does not contain equality literals with functional terms. Finally, since a closure of type 8 does not contain a functional term, it can participate in an inference only with a literal not containing a functional term. Since such a literal must be maximal, it may occur only in a closure not containing a functional term. □

3.3.4 Termination and Complexity Analysis

We show that the number of $\mathcal{ALCHIQ}^-$-closures is finite for a finite signature. This, in combination with Lemma 3.3.5 and the soundness and completeness of $\mathcal{BS}_{DL}$ shows that $\mathcal{BS}_{DL}$, with eager application of redundancy elimination rules, is a decision procedure for checking satisfiability of $\mathcal{ALCHIQ}^-$ knowledge bases.

Lemma 3.3.8. *Let N_i be any closure set obtained in a derivation as defined in Lemma 3.3.5. If C is a closure in N_i, then the number of literals in C is at most polynomial in $|KB|$, for unary coding of numbers in input. Furthermore, $|N_i|$ is at most exponential in $|KB|$, for unary coding of numbers in input.*

Proof. By Lemma 3.3.5, N_i can contain only $\mathcal{ALCHIQ}^-$-closures. Since redundancy elimination is applied eagerly, N_i cannot contain closures with duplicate literals or closures identical up to variable renaming. Let r denote the number of role predicates, a the number of atomic concept predicates, c the number of constants and f the number of function symbols occurring in the signature of $\Xi(KB)$. By definition of $|KB|$, r and

c are obviously linear in $|KB|$. Furthermore, a is also linear in $|KB|$, since the number of new atomic concept predicates introduced during preprocessing is bounded by the number of subconcepts of each concept, which is linear in $|KB|$. The number f is bounded by the sum of all numbers n in $\geq n\, R.C$ and $\leq n\, R.C$ plus one for each $\exists R.C$ and $\forall R.C$ occurring in KB. Since numbers are coded in unary, f is linear in $|KB|$. Let n denote the maximal number occurring in any number restriction. For unary coding of numbers, n is linear in $|KB|$.

Consider now the maximal number of literals in a closure of type 5. The maximal number of literals for $\mathbf{P_1}(x)$ is $2a$ (factor 2 allows for each atomic concept predicate to occur positively or negatively), for $\mathbf{P_2}(\langle \mathbf{f}(x) \rangle)$ it is $2a \cdot 2f$ (f is multiplied by 2 since each term may or may not be marked), for equalities it is f^2 (both terms are always marked), and for inequalities it is $4f^2$ (factor 4 allows for each side of the equality to be marked or not). Hence, the maximal number of literals is $2a+4af+f^2+4f^2$. For a closure of type 6, the maximal number of literals is $2a+2a+4af+(f^2+f)+(4f^2+2f)$: possible choices for g do not contribute to the closure length, and the expressions in parenthesis take into account that each term in an equality or an inequality can be $f_i(g(x))$ or x. For a closure of type 8, the maximal number of literals is $2r \cdot 2c \cdot 2c + 2a \cdot 2c + 2a \cdot 2f \cdot c + 2 \cdot (4c^2 + c \cdot f^2 + cf \cdot 2c)$: the factor 2 in front of the parenthesis takes into account that equalities and inequalities may have the same form, and the expression in the parenthesis counts all possible forms these literals may have. The maximal number of literals of closures of type 1 and 2 is obviously 2, and for closures of type 3 and 4 it is $a+1$. For a closure of type 7, the number of variables y_i is bounded by n: each such closure in $\Xi(KB)$ contains at most n variables, and no inference steps increases the number of variables. The maximal number of literals in a closure of type 7 is $n+4c+n^2$, since the choice for R does not contribute to the closure length. Hence, the maximal number of literals in any closure is polynomial in $|KB|$, for unary coding of numbers.

The maximal number of closures of type 1–6 and 8 in N_i is now easily obtained as follows: if C_ℓ is the closure with maximal number of literals ℓ for some closure type, then there are 2^ℓ subsets of literals of C_ℓ. To obtain the total number of closures, one must multiply 2^ℓ with the number of closure-wide choices. For closures of type 6, the function symbol g can be chosen in f ways. For closures of type 3 and 4, one can choose R and f in rf ways. For closures of type 1, one can choose R in r ways. For closures of type 2, one can choose R and S in r^2 ways. Since all these factors are polynomial in $|KB|$ for unary coding of numbers, we obtain an exponential bound on the number of closures of types 1–6 and 8. Finally, no inference derives a new closure of type 7, so N_i contains only those closures of type 7 which are contained in $\Xi(KB)$. □

Theorem 3.3.9. *For an $\mathcal{ALCHIQ}^-$ knowledge base KB, saturation of $\Xi(KB)$ by $\mathcal{BS}_{DL}$ with eager application of redundancy elimination rules decides satisfiability of KB and runs in time exponential in $|KB|$, for unary coding of numbers in input.*

Proof. Translation of KB into $\Xi(KB)$ can be performed in time polynomial in $|KB|$ by Lemma 3.3.2, and contains only $\mathcal{ALCHIQ}^-$-closures by Lemma 3.3.4. Let ℓ denote the

maximal number of closures occurring in the closure set in a derivation as specified in Lemma 3.3.5, and let l denote the maximal number of literals in a closure. By Lemma 3.3.8, ℓ is exponential, and l polynomial in $|KB|$, for unary coding of numbers. Since all terms are unary, ordering constraints can be checked in time polynomial in l. In [46], a subsumption decision algorithm was presented, running in time exponential in l. Furthermore, the subsumption check is performed at most for each pair of closures, so it takes at most exponential time. Apart from ordered hyperresolution with a closure of type 7, each of the four inference rules can potentially be applied to any closure pair. Since for closures other than of type 7 exactly one literal is maximal/selected, this gives rise to at most $4\ell^2$ inferences. For a hyperresolutiuon inference with a closure of type 7, n side premises can be chosen in ℓ^n ways. Hence, the number of applications of inference rules of $\mathcal{BS}_{DL}$ is bounded by $4\ell^2 + \ell^n$, which is exponential in $|KB|$ for unary coding of numbers in input. Now it is obvious that, after at most an exponential number of steps, the set of closures will be saturated, and the procedure will terminate. Since $\mathcal{BS}_{DL}$ is sound and complete with eager application of redundancy elimination rules, the claim of the theorem follows. □

In the proof of Theorem 3.3.9, we assumed an exponential algorithm for checking subsumption. In practice, it is known that modern theorem provers spend up to 90% of their time in subsumption checking, so an algorithm with a better worst case complexity on $\mathcal{ALCHIQ}^-$-closures is useful in practice.

Lemma 3.3.10. *Subsumption checks for $\mathcal{ALCHIQ}^-$-closures may be performed in polynomial time.*

Proof. In [46] it was shown that subsumption between closures having at most one variable can be checked in polynomial time. This algorithm can be easily extended to additionally check η-reducibility, so subsumption checking for closures of type 3, 4, 5, 6 and 8 can be performed in polynomial time. Checking whether a closure of type 1 or 2 subsumes some other closure can be performed by matching the negative literal first, and then checking whether the positive literal matches as well, which can be performed in quadratic time.

Let C and C' be two closures of type 7. For C to subsume C', the only possibility is that σ contains mappings $y_i \mapsto y'_i$ and $x \mapsto x'$. Hence, C subsumes C' only if the number of variables y_i in C is smaller than in C', both closures contain the same role R, and the predicates occurring in $\mathbf{P_1}(x)$ and $\mathbf{P_2}(\mathbf{y})$ of C are a subset of the predicates occurring in C', respectively. These checks can obviously be performed in polynomial time. □

3.4 Removing the Restriction to Very Simple Roles

In this section we show how to remove the restriction to very simple roles and thus obtain an algorithm for deciding satisfiability of an $\mathcal{ALCHIQ}$ knowledge base KB. Our main problem is that by saturating $\Xi(KB)$, we may obtain closures whose structure

corresponds to the Table 3.2, but for which conditions (*iii*)–(*vi*) do not hold; we call such closures $\mathcal{ALCHIQ}$-closures.

Consider the following knowledge base KB and its translation into closures $\Xi(KB)$:

$$R \sqsubseteq T \qquad \neg R(x,y) \vee T(x,y) \tag{3.1}$$
$$S \sqsubseteq T \qquad \neg S(x,y) \vee T(x,y) \tag{3.2}$$
$$C \sqsubseteq \exists R.\top \qquad \neg C(x) \vee R(x, f(x)) \tag{3.3}$$
$$\top \sqsubseteq \exists S^{-}.\top \qquad S^{-}(x, g(x)) \tag{3.4}$$
$$\top \sqsubseteq \leq 1\, T \quad \rightsquigarrow \quad \neg T(x,y_1) \vee \neg T(x,y_2) \vee y_1 \approx y_2 \tag{3.5}$$
$$\exists S.\top \sqsubseteq D \qquad \neg S(x,y) \vee D(x) \tag{3.6}$$
$$\exists R.\top \sqsubseteq \neg D \qquad \neg R(x,y) \vee \neg D(x) \tag{3.7}$$
$$\top \sqsubseteq C \qquad C(x) \tag{3.8}$$
$$\neg S^{-}(x,y) \vee S(y,x) \tag{3.9}$$

Consider a saturation of $\Xi(KB)$ by $\mathcal{BS}_{DL}$. Resolving (3.4) with (3.9) yields (3.10). Furthermore, (3.3) and (3.10) are resolved with (3.1) and (3.2) to produce (3.11) and (3.12), respectively. These can then be resolved with (3.5) to produce (3.13):

$$S([g(x)], x) \tag{3.10}$$
$$\neg C(x) \vee T(x, [f(x)]) \tag{3.11}$$
$$T([g(x)], x) \tag{3.12}$$
$$\neg C([g(x)]) \vee [f(g(x))] \approx x \tag{3.13}$$

For (3.13), Condition (*v*) is not satisfied: $\mathsf{role}(f) = R \neq \mathsf{Inv}(\mathsf{role}(g)) = \mathsf{Inv}(S^{-}) = S$. Obviously, this is because in (3.5), a number restriction was stated on a role that is not very simple. Now (3.13) can be superposed into (3.3), resulting in (3.14):

$$\neg C([g(x)]) \vee R([g(x)], x) \tag{3.14}$$

Since condition (*v*) is not satisfied for (3.13), we cannot assume that there is a closure that subsumes (3.14), as we did in the proof of Lemma 3.3.5. Hence, we must keep (3.14), which is obviously not an $\mathcal{ALCHIQ}$-closure. This might cause termination problems, as, in general, (3.14) might be resolved with some closure of type 6 of the form $C([g(h(x))])$, producing a closure of the form $R([g(h(x))], [h(x)])$. The term depth in the binary literal is now two, and it is not difficult to see that, by resolving it with some closure of type 7, it is possible to derive closures with ever deeper terms. Hence, Lemma 3.3.8, stating that the number of closures that can be derived is finite, does not hold any more, so saturation does not necessarily terminate.

A careful analysis of the problem reveals that various refinements of the ordering and the selection function will not help. Furthermore, (3.14) is necessary for completeness. Namely, KB is unsatisfiable, and the empty clause may be derived only by the following deduction, which involves (3.14):

$$D([g(x)]) \tag{3.15}$$
$$\neg D([g(x)]) \vee \neg C([g(x)]) \tag{3.16}$$
$$\neg C([g(x)]) \tag{3.17}$$
$$\square \tag{3.18}$$

3.4.1 Transformation by Decomposition

To remedy the problems outlined above, we introduce *decomposition* — an additional transformation which may be applied to the result of any $\mathcal{BS}$ inference. This transformation is generally applicable and is not limited to $\mathcal{ALCHIQ}$ or description logic. We show that decomposition can be combined with basic superposition, but in a similar way one can show that decomposition can be combined with any clausal calculus compatible with the general notion of redundancy [14]. In Subsection 3.4.2 we extend $\mathcal{BS}_{DL}$ with decomposition to obtain a decision procedure for $\mathcal{ALCHIQ}$.

In the following, for $\mathbf{x}$ a vector of distinct variables $x_1, \ldots, x_n$, and $\mathbf{t}$ a vector of (not necessarily distinct) terms $t_1, \ldots, t_n$, let $\{\mathbf{x} \mapsto \mathbf{t}\}$ denote the substitution $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, and let $Q([\mathbf{t}])$ denote $Q([t_1], \ldots, [t_n])$.

Definition 3.4.1. *Let $C \cdot \rho$ be a closure and N a set of closures. A* decomposition *of $C \cdot \rho$ w.r.t. N is a pair of closures $C_1 \cdot \rho \vee Q([\mathbf{t}])$ and $C_2 \cdot \theta \vee \neg Q(\mathbf{x})$ where $\mathbf{t}$ is a vector of n terms and $\mathbf{x}$ is a vector of n distinct variables, $n \geq 0$, such that (*i*) $C = C_1 \cup C_2$, (*ii*) $\rho = \theta\{\mathbf{x} \mapsto \mathbf{t}\}$, (*iii*) the set of free variables of $C_2\theta$ is equal to $\mathbf{x}$ and, (*iv*) if $C_2 \cdot \theta \vee \neg Q'(\mathbf{x}) \in N$, then $Q = Q'$, otherwise Q is a new predicate not occurring in N. The closure $C_2 \cdot \theta$ is called the* fixed *part, the closure $C_1 \cdot \rho$ is called the* variable *part, and the predicate Q is called the* definition predicate*. An application of decomposition is often written as*

$$C \cdot \rho \rightsquigarrow \begin{array}{l} C_1 \cdot \rho \vee Q([\mathbf{t}]) \\ C_2 \cdot \theta \vee \neg Q(\mathbf{x}) \end{array}$$

*With $\mathcal{BS}^+$ we denote the $\mathcal{BS}$ calculus where (*a*) the conclusion of any inference performed on a non-definition predicate can be replaced by its decomposition w.r.t. the set of closures derived thus far, (*b*) in each saturation only a finite number of definition predicates is introduced, and (*c*) definition predicates are smaller than non-definition predicates.*

As an example, consider superposition from a closure $[f(g(x))] \approx [h(g(x))]$ into a closure $C(x) \vee R(x, f(x))$, resulting in a closure $C([g(x)]) \vee R([g(x)], [h(g(x))])$. The conclusion is obviously not an $\mathcal{ALCHIQ}$-closure, so performing further inferences with it might lead to non-termination. However, the closure can be decomposed into closures $C([g(x)]) \vee Q_{R,f}([g(x)])$ and $\neg Q_{R,f}(x) \vee R(x, [h(x)])$, which are both $\mathcal{ALCHIQ}$-closures, and do not cause termination problems. We first show that $\mathcal{BS}^+$ is sound.

Lemma 3.4.2. *Let $N_0, \ldots, N_i$ be a $\mathcal{BS}^+$-derivation, and let I_0 be a model of N_0. Then for $i > 1$, N_i has a model I_i such that, if the inference deriving N_i from N_{i-1} involves a decomposition step as specified in Definition 3.4.1 introducing a new predicate Q, then $I_i = I_{i-1} \cup \{Q(\mathbf{s}) \mid \mathbf{s}$ is a vector of ground terms such that $C_2\theta\{\mathbf{x} \mapsto \mathbf{s}\}$ is true in $I_{i-1}\}$; otherwise $I_i = I_{i-1}$.*

Proof. The proof proceeds by induction on the length of the derivation $N_0, \ldots, N_i$. The base case is trivial, since I_0 is a model of N_0 by the assumption. For the induction step, we assume that N_{i-1} has a model I_{i-1} satisfying the conditions of the lemma and consider possible inferences deriving N_i. For inferences without a decomposition step, the claim is trivial. For example, consider a positive superposition inference from $(s \approx t \vee C) \cdot \rho$ into $(w \approx v \vee D) \cdot \rho$ with unifier σ resulting in $(C \vee D \vee w[t]_p \approx v) \cdot \theta$, where $\theta = \rho\sigma$, and let τ be a ground substitution. If $(s \approx t) \cdot \theta\tau$ is false in I_{i-1}, then $C \cdot \theta\tau$ is true in I_{i-1}; if $(w \approx v) \cdot \theta\tau$ is false in I_{i-1}, then $D \cdot \theta\tau$ is true in I_{i-1}; and if both $(s \approx t) \cdot \theta\tau$ and $(w \approx v) \cdot \theta\tau$ are true in I_{i-1}, then $(w[t]_p \approx v) \cdot \theta\tau$ is true in I_{i-1}. The cases for other inference rules are similar.

If the inference deriving N_i from N_{i-1} involves a decomposition step, then two cases are possible. If the predicate Q is new, then I_{i-1} can be extended to I_i by adding those ground literals $Q(\mathbf{s})$ for which $C_2\theta\{\mathbf{x} \mapsto \mathbf{s}\}$ is true in I_{i-1}. Hence, for any ground substitution τ, in $C_2 \cdot \theta\tau \vee \neg Q(\mathbf{x})\tau$ either $C_2 \cdot \theta\tau$ or $\neg Q(\mathbf{x})\tau$ is true in I_i. Furthermore, if $C \cdot \rho\tau$ is true in I_{i-1}, then $C_1 \cdot \rho\tau \vee Q([\mathbf{t}])\tau$ is obviously true in I_i. If the predicate Q is not new, then $I_i = I_{i-1}$. Then, by the induction hypothesis $Q(\mathbf{s})$ is true if and only if $C_2\theta\{\mathbf{x} \mapsto \mathbf{s}\}$ is true, and $C_1 \cdot \rho\tau \vee Q([\mathbf{t}])\tau$ is true in I_i as in the previous case. □

We now show that, if the result of an inference is decomposed in a way that makes the literals involving Q minimal, then the fixed and the variable part together make the inference redundant. This is the key step in showing completeness of $\mathcal{BS}^+$.

Lemma 3.4.3. *Let ξ be a $\mathcal{BS}$ inference applied to premises $D_1 \cdot \sigma$ and $D_2 \cdot \sigma$ from a closure set N on a literal not containing a definition predicate, resulting in a closure $C \cdot \rho$. If $C \cdot \rho$ can be decomposed into closures $C_1 \cdot \rho \vee Q([\mathbf{t}])$ and $C_2 \cdot \theta \vee \neg Q(\mathbf{x})$ which are both redundant in N, then the inference ξ is redundant in N.*

Proof. Let ξ be an inference with premises $D_1 \cdot \sigma$ and $D_2 \cdot \sigma$ resulting in a closure $C \cdot \rho$, R a rewrite system, and τ a ground substitution such that $\xi\tau$ is a variable irreducible ground instance of ξ w.r.t. R. Let $E_1 = (C_1 \cdot \rho \vee Q([\mathbf{t}]))\tau$ and $E_2 = (C_2 \cdot \theta \vee \neg Q(\mathbf{x}))\tau$. Finally, let $D = \max(D_1 \cdot \sigma\tau, D_2 \cdot \sigma\tau)$.

It is not hard to see that, for any conclusion $C \cdot \rho\tau$ of a ground $\mathcal{BS}$ inference $\xi\tau$, we have $C \cdot \rho\tau \prec D$. Namely, the side premise participates in an inference always on the maximal literal L_s which is, by the ordering constraints of $\mathcal{BS}$ inference rules, always smaller than the corresponding literal L_m of the main premise. Furthermore, superposition inferences are allowed only from the maximal side of the equality, so the literal L'_m produced by superposition of L_s into L_m is always smaller than L_m. Similar observations hold for non-superposition inferences. Hence, $L'_m \preceq C \cdot \rho\tau \prec L_m$.

By Definition 3.4.1, all definition predicates are smaller than non-definition predicates, so $Q(\mathbf{x})\tau \preceq L'_m$ and $Q([\mathbf{t}])\tau \preceq L'_m$. Hence, $E_1 \preceq L'_m \prec L_m \preceq D$ and $E_2 \preceq L'_m \prec L_m \preceq D$. Notice that, unless definition predicates are required to be smallest, decomposition might introduce a literal which is bigger than the literals in the premises; this is the reason for restriction (c) of Definition 3.4.1. Furthermore, if ξ is performed on a literal containing a decomposition predicate Q', then decomposition might introduce a predicate Q which is larger than Q'; this is the reason for restriction (a) of Definition 3.4.1.

Now the vector of terms $\mathbf{t}$ is "extracted" from the substitution part of $C \cdot \rho$. Hence, if a term t occurs in E_1 and E_2 at a substitution position, then t occurs in $C \cdot \rho\tau$ also at a substitution position. Therefore, if $C \cdot \rho\tau$ is variable irreducible w.r.t. R, so are E_1 and E_2.

To summarize, for all rewrite systems R and all ground substitutions τ such that $\xi\tau$ is a variable irreducible ground instance of ξ w.r.t. R, the closures E_1 and E_2 are $\prec$-smaller than D, and they are variable irreducible w.r.t R if $C \cdot \rho\tau$ is variable irreducible w.r.t R. The closure $C_1 \cdot \rho \vee Q([\mathbf{t}])$ is redundant in N by assumption, so $R \cup \mathsf{irred}_R(N)^{\preceq E_1} \models E_1$ but, since $E_1 \prec D$, we have $R \cup \mathsf{irred}_R(N)^{\prec D} \models E_1$. Similarly $R \cup \mathsf{irred}_R(N)^{\prec D} \models E_2$. Since $\{E_1, E_2\} \models C \cdot \rho\tau$, we have $R \cup \mathsf{irred}_R(N)^{\prec D} \models C \cdot \rho\tau$, so the claim of the lemma holds. □

We are now ready to show that $\mathcal{BS}^+$ is a sound and complete refutation calculus.

Theorem 3.4.4. *For N_0 a set of closures of the form $C \cdot \{\}$, let N be a set of closures obtained by saturating N_0 under $\mathcal{BS}^+$. Then N_0 is satisfiable if and only if N does not contain the empty closure.*

Proof. The $(\Rightarrow)$ direction follows immediately from Lemma 3.4.2. For the $(\Leftarrow)$ direction, assume that N is saturated under $\mathcal{BS}^+$ and contains only a finite number of definition predicates. Then, by Lemma 3.4.3, N is saturated under $\mathcal{BS}$ as well, so using the model generation method, we can build a rewrite system R, such that $R^* \models \mathsf{irred}_R(N)$. Unlike for basic superposition without decomposition, the set of closures N does not need to be well-constrained, so we cannot immediately assume that R^* is a model of N. However, we may conclude that $R^* \models \mathsf{irred}_R(N_0)$: consider a closure $C \in N_0$ and its variable irreducible ground instance $C\tau$. If $C \in N$, then R^* is obviously a model of $C\tau$. Furthermore, $C \notin N$ only if it is redundant in N, but then, for any τ, there are variable irreducible ground closures $D_i\tau \in \mathsf{irred}_R(N)$ such that $D_1\tau, \ldots, D_n\tau \models C\tau$. Hence, since $R^* \models D_i\tau$ by assumption, we have $R^* \models C\tau$ as well.

Now consider a closure $C \in N_0$ and its (not necessarily variable irreducible) ground instance $C\eta$. Let η' be a substitution obtained from η by replacing each mapping $x \mapsto t$ with $x \mapsto \mathsf{nf}_R(t)$. Since the substitution part of C is empty, $C\eta' \in \mathsf{irred}_R(N_0)$ and, since $R^* \models C\eta'$, we have $R^* \models C\eta$. Hence, $R^* \models N_0$, and by Lemma 3.4.2, there is a model of N. □

We explain the intuition behind the result of Theorem 3.4.4. Decomposition is essentially the structural transformation applied in the course of the theorem proving process. Since the formulae obtained by the structural transformation are equisatisfiable with the original formula, the application of the structural transformation does not affect soundness of completeness of the calculus. The only potential problem might be that decomposition somehow interferes with markers of basic superposition. This does not occur since, for any rewrite system R, decomposing $C \cdot \rho$ into $C_1 \cdot \rho \vee Q([\mathbf{t}])$ and $C_2 \cdot \theta \vee \neg Q(\mathbf{x})$ actually decomposes any variable irreducible ground instance of the premise into corresponding variable irreducible ground instances of the conclusions. In this way, we do not loose any variable irreducible ground instance "relevant" for detecting potential inconsistency of the closure set. It is worth mentioning that in a rewrite system R, closures $C_1 \cdot \rho \vee Q([\mathbf{t}])$ and $C_2 \cdot \theta \vee \neg Q(\mathbf{x})$ can have variable irreducible ground instances which do not correspond to a variable irreducible ground instance of $C \cdot \rho$. However, this "excessive" variable irreducible ground instances do not cause problems, since decomposition is a sound inference. The restriction to introducing a finite number of definition predicates ensures that we do not get an infinite signature of the closure set, which might invalidate the basic assumptions of resolution-based theorem proving.

For any other sound clausal calculus, Lemma 3.4.2 applies identically. Furthermore, for any calculus compatible with the standard notion of redundancy [14], Lemma 3.4.3 can be proved in a similar manner with minor differences.

3.4.2 Deciding $\mathcal{ALCHIQ}$ by Decomposition

We now show how to apply the decomposition rule from Subsection 3.4.1 to obtain a decision procedure for checking satisfiability of $\mathcal{ALCHIQ}$ knowledge bases.

Definition 3.4.5. *Let $\mathcal{BS}^{+}_{DL}$ be the $\mathcal{BS}_{DL}$ calculus where conclusions, whenever possible, are decomposed as follows, for an arbitrary term t:*

$$D \cdot \rho \vee R([t], [f(t)]) \rightsquigarrow \begin{array}{r} D \cdot \rho \vee Q_{R,f}([t]) \\ \neg Q_{R,f}(x) \vee R(x, [f(x)]) \end{array}$$

$$D \cdot \rho \vee R([f(x)], x) \rightsquigarrow \begin{array}{r} D \cdot \rho \vee Q_{\mathsf{Inv}(R),f}(x) \\ \neg Q_{\mathsf{Inv}(R),f}(x) \vee R([f(x)], x) \end{array}$$

The precedence of the LPO is $f >_P c >_P p >_P Q_{S,f} >_P \top$, for any function symbol f, constant symbol c, non-definition predicate p and definition predicate $Q_{S,f}$.

By Definition 3.4.1, for a (possibly inverse) role S and a function symbol f, the definition predicate $Q_{S,f}$ is unique. Furthermore, a strict application of Definition 3.4.1 would require introducing a distinct definition predicate $Q'_{R,f}$ for $R([f(x)], x)$. However, due to the translation operator π, $R([f(x)], x)$ and $\mathsf{Inv}(R)(x, [f(x)])$ are logically equivalent. Therefore, the predicate $Q_{\mathsf{Inv}(R),f}$ may be used instead of $Q'_{R,f}$

as the definition predicate for $R([f(x)], x)$, thus avoiding the need to introduce an additional definition predicate in the second form of decomposition in Definition 3.4.5.

Intuitively, $\mathcal{BS}^+_{DL}$ decides satisfiability of $\mathcal{ALCHIQ}$ knowledge bases because decomposition replaces a non-$\mathcal{ALCHIQ}$-closure with two $\mathcal{ALCHIQ}$-closures. Furthermore, since the definition predicate $Q_{R,f}$ is unique for a pair of role and function symbols R and f, at most a polynomial number of definition predicates may be introduced during saturation, so the result of Theorem 3.4.4 applies. Since the number of $\mathcal{ALCHIQ}$-closures is finite according to Lemma 3.3.8, $\mathcal{BS}^+_{DL}$ terminates.

Theorem 3.4.6. *For an $\mathcal{ALCHIQ}$ knowledge base KB, saturation of $\Xi(KB)$ by $\mathcal{BS}^+_{DL}$ decides satisfiability of KB, and runs in time exponential in $|KB|$, for unary coding of numbers in input.*

Proof. The proof of Lemma 3.3.5 obviously applies even if conditions (*iii*)–(*vi*) from Table 3.2 do not hold; the only exception is a superposition into a generator closure $\mathbf{P^f}(x) \vee R(x, f(x))$. For the latter, there are these three possibilities, depending on the structure of the premise that superposition is performed from:

- Superposition from a closure of type 5, 6 or 8 of the form $[f(t)] \approx [g(t)] \vee D \cdot \rho$, where t is either a variable x', a term $g(x)$ or a constant a, results in a closure of the form $\mathbf{P^f}([t]) \vee R([t], [g(t)]) \vee D \cdot \rho$, which is decomposed into a closure of type 3 and a closure of type 5, 6 or 8.
- Superposition from a closure of type 6 of the form $[f(g(x'))] \approx x' \vee D \cdot \rho$ results in a closure of the form $\mathbf{P^f}([g(x')]) \vee R([g(x')], x') \vee D \cdot \rho$. This closure is decomposed into a closure of type 4 and a closure of type 5 or 6. Since $R([g(x')], x')$ and $\mathsf{Inv}(R)(x', [g(x')])$ are logically equivalent due to the translation operator π, the predicate $Q_{\mathsf{Inv}(R),f}$ can be used as the definition predicate for $R([g(x')], x')$.
- Superposition from a closure of type 8 of the form $[f(a)] \approx [b] \vee D \cdot \rho$ results in a closure of the form $\mathbf{P^f}(a) \vee R([a], [b]) \vee D \cdot \rho$, which is of type 8.

Hence, by decomposition, the conclusion of each inference of $\mathcal{BS}^+_{DL}$ is transformed into an $\mathcal{ALCHIQ}$-closure. Let r be a number of roles and f the number of function symbols occurring in $\Xi(KB)$; as in Lemma 3.3.8, both r and f are linear in $|KB|$ for unary coding of numbers. The number of definition predicates $Q_{R,f}$ introduced by decomposition is then bounded by $r \cdot f$, which is quadratic in $|KB|$, so the number of different predicates is polynomial in $|KB|$. Hence, Lemma 3.3.8 applies in this case as well, so the maximal set of closures derived in a saturation is at most exponential in $|KB|$ for unary coding of numbers. After deriving this set, all inferences of $\mathcal{BS}^+_{DL}$ are redundant and the saturation terminates. Since only a finite number of definition predicates are introduced, by Theorem 3.4.4, $\mathcal{BS}^+_{DL}$ is sound and complete, so the claim of this theorem follows. □

Although this result supersedes the Theorem 3.3.9, in practice it is useful to know that, for roles not having superroles, superposition into a generator is not necessary. In

this way a practical implementation can be optimized not to perform inferences whose conclusions are immediately subsumed.

We note that for most knowledge bases, not all predicates $Q_{R,f}$ will occur in a derivation. Rather, only predicates from the set $\mathsf{dec}(KB)$, defined next, may be introduced by decomposition:

Definition 3.4.7. *For an $\mathcal{ALCHIQ}$ knowledge base KB, $\mathsf{dec}(KB)$ is the set of those predicates $Q_{R,f}$, for which $R \neq \mathsf{role}(f)$ and there exists a role S such that:*

- *S occurs in an at-least number restriction under positive, or in an at-most number restriction under negative polarity in KB,*
- *$R \sqsubseteq^* S$ or $\mathsf{Inv}(R) \sqsubseteq^* S$ and*
- *$\mathsf{role}(f) \sqsubseteq^* S$ or $\mathsf{Inv}(\mathsf{role}(f)) \sqsubseteq^* S$.*

Furthermore, let $\mathsf{gen}(KB) = \{\neg Q_{R,f} \vee R(x, [f(x)]) \mid Q_{R,f} \in \mathsf{dec}(KB)\}$.

Lemma 3.4.8. *In any saturation of $\Xi(KB)$ by $\mathcal{BS}^+_{DL}$, only predicates from $\mathsf{dec}(KB)$ can be introduced by the decomposition rule.*

Proof. A predicate $Q_{R,f}$ is introduced by decomposing the conclusion of a superposition into a generator $\mathbf{P}^{\mathbf{g}}(x) \vee R(x, g(x))$ from a literal of the form $[g(f(x))] \approx x$ where $\mathsf{role}(g) \neq \mathsf{Inv}(\mathsf{role}(f))$, or $[g(t)] \approx [f(t)]$ where $\mathsf{role}(g) \neq \mathsf{role}(f)$. However, such a literal can only be generated by a resolution with a closure of type 7, obtained by translating an at-least restriction on some role S into clausal form. Unless $R \sqsubseteq^* S$ or $\mathsf{Inv}(R) \sqsubseteq^* S$, the function symbol g cannot occur in any literal of the form $[g(f(x))] \approx x$ or $[g(t)] \approx [f(t)]$. Similarly, unless $\mathsf{role}(f) \sqsubseteq^* S$ or $\mathsf{Inv}(\mathsf{role}(f)) \sqsubseteq^* S$, the function symbol f cannot occur in any such literal either. □

The sets $\mathsf{dec}(KB)$ and $\mathsf{gen}(KB)$ are used in the algorithm for reducing an $\mathcal{ALCHIQ}$ knowledge base KB to a disjunctive datalog program presented in Chapter 5.

3.5 Safe Role Expressions

The decomposition rule introduced in Subsection 3.4.1 is actually very versatile, and can be used to decide a stronger logic $\mathcal{ALCHIQ}b$, which additionally allows a certain type of *safe* role expressions.

Definition 3.5.1. *A* role expression *is a finite expression built over the set of abstract roles using the connectives $\sqcup$, $\sqcap$ and $\neg$ in the usual way. Let safe^+ and safe^- be functions defined on the set of all role expressions as specified below, where R is an abstract role, and E and E_i are role expressions:*

$$\begin{aligned} \mathsf{safe}^+(R) &= \top & \mathsf{safe}^-(R) &= \bot \\ \mathsf{safe}^+(\neg E) &= \neg\mathsf{safe}^-(E) & \mathsf{safe}^-(\neg E) &= \neg\mathsf{safe}^+(E) \\ \mathsf{safe}^+(\sqcup E_i) &= \bigwedge \mathsf{safe}^+(E_i) & \mathsf{safe}^-(\sqcup E_i) &= \bigvee \mathsf{safe}^+(E_i) \\ \mathsf{safe}^+(\sqcap E_i) &= \bigvee \mathsf{safe}^+(E_i) & \mathsf{safe}^-(\sqcap E_i) &= \bigwedge \mathsf{safe}^+(E_i) \end{aligned}$$

A role expression E is safe *if* $\mathsf{safe}^+(E) = \top$. *The description logic $\mathcal{ALCHIQ}b$ is obtained from $\mathcal{ALCHIQ}$ by allowing concepts $\exists E.C$, $\forall E.C$, $\geq n\, E.C$ and $\leq n\, E.C$, inclusion axioms $E \sqsubseteq F$ and ABox axioms $E(a, b)$, where E is a safe role expression, and F is any role expression. The semantics of $\mathcal{ALCHIQ}b$ is obtained by extending the translation operator π as specified in Table 3.3.*

Safe expressions allow stating negative or disjunctive knowledge for roles. Roughly speaking, "relativized" statements such as

$$\forall x, y : \mathit{isParentOf}(x, y) \rightarrow \mathit{isMotherOf}(x, y) \vee \mathit{isFatherOf}(x, y)$$

are allowed, but "fully negated" statements such as the one below are not allowed:

$$\forall x, y : \neg \mathit{isMotherOf}(x, y) \rightarrow \mathit{isFatherOf}(x, y)$$

In [114], a similar logic $\mathcal{ALCIQ}b$ was considered. There, the safety condition for role expressions required that, if the expression is transformed into disjunctive normal form, then each disjunct contains at least one non-negated conjunct. It is straightforward to see that these two definitions coincide. We use the above definition because transformation into disjunctive normal form can introduce exponential blowup, which we avoid by using the structural transformation. The above definition makes it easier to identify those role expressions, for which the structural transformation produces closures with syntactic properties needed to show termination.

To decide satisfiability of an $\mathcal{ALCHIQ}b$ knowledge base KB, we extend the preprocessing of KB to transform each role expression into negation-normal form, and to introduce a new name for each non-atomic role expression. The set of closures obtained by preprocessing is also denoted with $\Xi(KB)$. From [86] it is known that $\Xi(KB)$ can be computed in polynomial time.

Theorem 3.5.2. *For an $\mathcal{ALCHIQ}b$ knowledge base KB, saturation of $\Xi(KB)$ by $\mathcal{BS}^+_{DL}$ decides satisfiability of KB in time exponential in $|KB|$, for unary coding of numbers in input.*

Proof. In addition to $\mathcal{ALCHIQ}$ closures, $\Xi(KB)$ can contain closures obtained by translating role expressions. For a role expression E occurring in concepts $\exists E.C$ and $\geq n\, E.C$ under positive polarity, or in concepts $\forall E.C$ and $\leq n\, E.C$ under negative

Table 3.3: Semantics of Role Expressions

$\pi(R, X, Y)$	$=$	$R(X, Y)$
$\pi(\neg R, X, Y)$	$=$	$\neg R(X, Y)$
$\pi(\sqcap E_i, X, Y)$	$=$	$\bigwedge E_i(X, Y)$
$\pi(\sqcup E_i, X, Y)$	$=$	$\bigvee E_i(X, Y)$

polarity, structural transformation introduces a formula of the following form, where Q is a new predicate:

$$\forall x, y : Q(x, y) \rightarrow \pi(E, x, y) \tag{3.19}$$

For a role expression E occurring in concepts $\exists E.C$ and $\geq n\, E.C$ under negative polarity, or in concepts $\forall E.C$ and $\leq n\, E.C$ under positive polarity, the structural transformation introduces a formula of the following form, where Q is a predicate:

$$\forall x, y : \pi(E, x, y) \rightarrow Q(x, y) \tag{3.20}$$

Finally, for an inclusion axiom $E \sqsubseteq F$, the structural transformation introduces formulae of the form (3.19) and (3.20). Regardless of whether E is safe in (3.19) or not, the structural transformation and clausification of (3.19) produces closures containing the literal $\neg Q(x, y)$. Furthermore, in (3.20), E is safe, which means that, in the disjunctive normal form of E, each conjunct contains at least one positive literal. Hence, each disjunction in the conjunctive normal form of $\neg\pi(E, x, y)$ contains at least one negative literal, so the structural transformation and clausification of (3.20) also produces a closure with at least one negative literal. Hence, all closures produced by role expressions have the form (3.21), where $n > 0$ and $m \geq 0$:

$$\neg R_1(x, y) \vee \ldots \vee \neg R_n(x, y) \vee S_1(x, y) \vee \ldots \vee S_m(x, y) \tag{3.21}$$

Since such a closure always contains at least one negative literal, the only possible inference with it is hyperresolution on all negative literals. If one of the side premises is a ground closure, no side premise can have the maximal literal of the form $R(x, \langle f(x)\rangle)$ or $R([f(x)], x)$ (since $f(x)$ does not unify with a constant). Hence, the resolvent is a ground closure of type 8. Furthermore, if one side premise has a maximal literal of the form $R(x, \langle f(x)\rangle)$, then no side premise can have a maximal literal of the form $R'([g(x')], x')$, because this would require unifying x with $g(x')$ and $f(x)$ with x' simultaneously, which is not possible due to the occurs-check in unification [8]. If all side premises have the maximal literal the form $R_i(x_i, \langle f(x_i)\rangle)$, the resolvent has the form (3.22), for $\mathbf{S}(s, t) = S_1(s, t) \vee \ldots \vee S_m(s, t)$:

$$\mathbf{P}(x) \vee \mathbf{S}(x, [f(x)]) \tag{3.22}$$

This closure can be decomposed into (3.23)–(3.25):

$$\mathbf{P}(x) \vee Q_{S_1,f}(x) \vee \ldots \vee Q_{S_m,f}(x) \tag{3.23}$$

$$\neg Q_{S_1,f}(x) \vee S_1(x, [f(x)]) \tag{3.24}$$

$$\vdots$$

$$\neg Q_{S_m,f}(x) \vee S_m(x, [f(x)]) \tag{3.25}$$

Finally, if all side premises have the maximal literal of the form $R_i([f(x_i)], x_i)$, the resolvent has the form (3.26):

$$\mathbf{P}(x) \vee \mathbf{S}([f(x)], x) \tag{3.26}$$

This closure can further be decomposed into (3.27)–(3.29). Since $S_i([f(x)], x)$ and $\mathsf{Inv}(S_i)(x, [f(x)])$ are logically equivalent due to the translation operator π, the predicate $Q_{\mathsf{Inv}(S_i),f}$ can be used as the definition predicate for $S_i([f(x)], x)$, for each i, $1 \leq i \leq m$.

$$\mathbf{P}(x) \vee Q_{\mathsf{Inv}(S_1),f}(x) \vee \ldots \vee Q_{\mathsf{Inv}(S_m),f}(x) \tag{3.27}$$

$$\neg Q_{\mathsf{Inv}(S_1),f}(x) \vee S_1([f(x)], x) \tag{3.28}$$

$$\vdots$$

$$\neg Q_{\mathsf{Inv}(S_m),f}(x) \vee S_m([f(x)], x) \tag{3.29}$$

The closures we obtain match the ones from Table 3.2, so the proof of Lemma 3.3.5 applies to this case as well. Furthermore, the number of literals in a closure of type (3.21) is linear in the size of the role expression, so the claim of this theorem holds in the same way as for Theorem 3.4.6. □

Definition 3.4.7, specifying the set $\mathsf{dec}(KB)$ of possible predicates $Q_{R,f}$ that can occur in the saturation, can be simply extended to require that R or $\mathsf{Inv}(R)$ occur in the same role expression as S or $\mathsf{Inv}(S)$.

We briefly comment why role safety is important for decidability. Namely, due to safety, closures of type (3.21) always contain a negative literal which is selected. If this were not the case, a closure of the form $R(x, y)$ might participate in resolution with a closure $\mathbf{P_1}(x) \vee \neg R(x, y) \vee \mathbf{P_2}(x)$, producing a closure $\mathbf{P_1}(x) \vee \mathbf{P_2}(y)$. This closure does not match any closure from Table 3.2. Since $\mathbf{P_1}(x)$ and $\mathbf{P_2}(y)$ do not have variables in common, one may don't-know non-deterministically assume that either one is true, and thus reduce this closure to a closure of type from Table 3.2. This obviously increases the complexity of reasoning from ExpTime to NExpTime. In fact, in [76] it was shown that reasoning in a description logic with non-safe role expressions is NExpTime-complete.

3.6 Related Work

Decision procedures for various logics were at the focus of the automated theorem proving research from its early days. Three such procedures were already implemented by Wang in 1960: a procedure capable of deciding validity in propositional logic, a procedure for deriving theorems in propositional logic, and a procedure for deciding validity in the so-called **AE**-fragment of first-order logic, consisting of first-order formulae with a quantifier prefix of the form $\forall x_1 \ldots \forall x_m \exists y_1 \ldots \exists y_n$.

At the beginning of the sixties, Robinson introduced the resolution principle [100] for first-order logic consisting of a single inference rule. An automated theorem prover based on resolution does not need to make a choice as to which rule to apply next, so inference rule can be implemented in a straightforward manner. Soon after the initial work of Robinson, various refinements of resolution were developed, such as hyperresolution [101], ordered resolution [97] or lock resolution [22], to name just a few. The common goal of all of these refinements is to reduce the number of consequences generated in the theorem proving process without loosing completeness. A good overview of resolution and related refinements is given in the classical text-book by Chang and Lee [25].

Soon after the introduction of the resolution principle and its refinements, attempts were made to use these principles to obtain efficient decision procedures for various classes of first-order logic. The first such procedure was presented by Kallick [69] for the class of formulae with the quantification prefix $\forall x_1 \forall x_2 \exists y$. This decision procedure is based on a refinement of resolution which is incomplete for first-order logic and is therefore difficult to extend.

In [68] Joyner established the basic principles of resolution-based decision procedures. He observed that, if clauses derivable in a saturation by a resolution refinement have a bounded term depth and clause length, then saturation necessarily terminates. By choosing appropriate refinements, he presented decision procedures for the Ackermann class (where the formulae are restricted to the quantification prefix $\exists^* \forall \exists^*$), the Monadic class (where only unary predicates are allowed) and the Maslov class (where formulae are restricted to quantification prefix $\exists^* \forall^* \exists^*$ and the matrix is a conjunction of binary disjunctions).

In the years to follow, the approach by Joyner was applied to numerous other decidable classes, such as the $\mathbf{E}^+$ class [112], the $\mathcal{PVD}$ class [71], and the $\mathcal{PVD}^g_=$ class [92], to name just a few. An overview of these results is given in a monograph by Fermüller, Leitsch, Tammet and Zamov [41].

Decidability of description logics in the resolution framework has been studied extensively in [85, 67, 64]. There, the description logic $\mathcal{ALB}$ is embedded in the DL* clausal class, which is then decided using the resolution framework by Bachmair and Ganzinger [14]. The main advantage of using this framework lies in its effective redundancy elimination methods, which have proven themselves essential for the practical applicability of resolution calculi. $\mathcal{ALB}$ is a very expressive logic and allows for unsafe role expressions, but does not provide counting quantifiers.

In [44] a decision procedure for the modal logic with a single transitive modality K4 was presented. To deal with transitivity, the algorithm is based on the ordered chaining calculus [12]. This calculus consists of inference rules aimed at optimizing theorem proving with chains of binary roles. Unfortunately, our attempts to decide $\mathcal{SHIQ}$ using ordered chaining proved unsuccessful, mainly due to certain negative chaining inferences, which produced undesirable equality literals. Therefore, we adopted the approach for eliminating transitivity from Section 3.2.

The guarded fragment was introduced in [4] to explain and generalize the good properties of modal and description logic, such as decidability. A resolution decision procedure based on a non-liftable ordering was given in [32], and was later modified to handle the (loosely) guarded fragment with equality [43] by basing the algorithm on superposition [10]. Since the basic description logic $\mathcal{ALC}$ is actually a syntactic variant of the multi-modal logic K_m [103], it can be embedded into the guarded fragment and decided by [43]. Using the approach from [105], certain extensions of $\mathcal{ALC}$, such as role transitivity, can be encoded into $\mathcal{ALC}$ knowledge bases, so the algorithm from [43] can decide these extensions as well. However, the (loosely) guarded fragment is not capable of expressing $\mathcal{SHIQ}$, because of the counting quantifiers: equality is available in the logic, but each two pairs of free variables of a guarded formula must occur in a guard atom. In fact, in [55] it was shown that the guarded fragment has the finite model property, which is known not to hold for $\mathcal{SHIQ}$ [5, Chapter 2], thus hinting that other mechanisms are necessary for handling $\mathcal{SHIQ}$.

$\mathcal{SHIQ}$ can easily be embedded into the two-variable fragment of first-order logic with counting quantifiers $\mathcal{C}^2$. This fragment was shown to be decidable in [47] and a decision procedure based on a combination of resolution and integer programming was given in [95]. However, deciding satisfiability of $\mathcal{C}^2$ is NExpTime-complete, and $\mathcal{SHIQ}$ is an ExpTime-complete [114] logic. Thus, the decision procedure from [95] introduces an unnecessary overhead for $\mathcal{SHIQ}$. Furthermore, we do not see how to derive the desired reduction to disjunctive datalog based on this procedure.

The decomposition rule from Section 3.4 is closely related to the structural transformation [86]. However, structural transformation is usually applied as a preprocessing step and not in the theorem proving process. In [33] and [99] *splitting by propositional symbols* was considered, which allows splitting variable-disjoint subsets of a clause and connecting them by a propositional symbol. Finally, a so-called *separation* rule, similar to decomposition, was used to decide fluted logic in [104]. It was shown that resolution remains complete if the separation rule is applied a finite number of times during saturation. Our approach differs in that we demonstrate compatibility of the decomposition rule with the standard redundancy notion for basic superposition. Furthermore, contrary to all approaches cited above, our rule allows decomposing a complex term into simpler terms.

Chapter 4

Reasoning with Concrete Domains

Reasoning for description logics with a concrete domain has been studied predominately in the context of tableaux and automata algorithms. Hence, extending the results from Chapter 3 to handle concrete domains is not trivial: our algorithm is based on basic superposition, which is a clausal refutation calculus. Since clauses are disjunctions of literals, it is not straightforward to combine basic superposition with concrete domain constraint checking, which works with conjunctions of literals.

We extend our algorithm in two stages. Firstly, in Section 4.1 we present a general approach for combining concrete domain reasoning with clausal calculi whose completeness proof is based on the model generation method. In short, we introduce so-called *concrete domain resolution* inference rule, for which we show soundness and completeness. Since the model generation method is the standard technique for proving completeness of many state-of-the-art calculi, such as ordered resolution [14], basic superposition [15] or ordered chaining [12], concrete domain resolution can readily be used with any of them.

Secondly, in Section 4.2 we apply the concrete domain resolution rule to extend the decision procedure from Chapter 3 to handle $\mathcal{SHIQ}(\mathbf{D})$. We show that, assuming a bound on the arity of concrete predicates, extending the logic with a concrete domain does not increase the reasoning complexity, i.e. it remains in ExpTime.

4.1 Resolution with a Concrete Domain

We now present a general approach for reasoning with a concrete domain in the resolution framework. As usual for various resolution calculi, in Subsection 4.1.1 we consider first the resolution with a concrete domain on ground clauses, and in Subsection 4.1.3 we lift the ground calculus to general clauses.

4.1.1 Concrete Domain Resolution with Ground Clauses

We now present the *ground concrete domain resolution* calculus, $\mathcal{G}^{\mathbf{D}}$ for short, for checking **D**-satisfiability of a set of clauses, where **D** is an admissible concrete domain. In order not to make the presentation too technical, we add the *concrete domain resolution* rule to the ordered resolution calculus [14] only, and argue later that the rule can be combined with other calculi as well. As for ordinary resolution, $\mathcal{G}^{\mathbf{D}}$ is parameterized with an admissible ordering $\succ$ on literals, which is a reduction ordering total on ground literals such that $\neg A \succ A$, for any atom A. A literal L is (strictly) maximal with respect to a clause C if there is no literal $L' \in C$, such that $L' \succ L$ ($L' \succeq L$). A literal $L \in C$ is (strictly) maximal in C if and only if L is (strictly) maximal with respect to $C \setminus L$. We extend the literal ordering $\succ$ to clauses by identifying each clause with a multiset of literals, and compare clauses by the multiset extension of the literal ordering; we denote the clause ordering ambiguously with $\succ$. Since the literal ordering is total and well-founded on ground literals, the clause ordering is total and well-founded on ground clauses. Since $\Phi_{\mathbf{D}}$ is closed w.r.t. negation, we assume without loss of generality that all concrete domain predicates occur in all clauses positively.

Definition 4.1.1. *A set $S = \{d_i(\mathbf{t_i})\}$ of positive concrete literals is a* **D**-constraint *if $\widehat{S}$ is* not **D***-satisfiable. A* **D***-constraint S is* minimal *if $\widehat{S'}$ is* **D***-satisfiable for each $S' \subsetneq S$; S is* connected *if it cannot be decomposed into two disjoint non-empty subsets S_1 and S_2 not sharing a common term (S_1 and S_2 do not share a common term if for all $d_i(\mathbf{t_i}) \in S_1$ and $d_j(\mathbf{t_j}) \in S_2$, we have $\mathbf{t_i} \cap \mathbf{t_j} = \emptyset$).*

Lemma 4.1.2. *Each minimal* **D***-constraint S is connected.*

Proof. Assume that S is a **D**-constraint, but it is not connected. Hence, S can be decomposed into subsets S_1 and S_2 not sharing a common term. Since S is a minimal **D**-constraint, $\widehat{S_1}$ and $\widehat{S_2}$ are **D**-satisfiable. However, since $\widehat{S_1}$ and $\widehat{S_2}$ do not have a common variable, $\widehat{S_1} \wedge \widehat{S_2} = \widehat{S}$ is **D**-satisfiable as well, which is a contradiction. □

Note that in its contrapositive form, Lemma 4.1.2 states that if S is not connected, then it is not a minimal **D**-constraint. We now present the inference rules of $\mathcal{G}^{\mathbf{D}}$.

Positive factoring:
$$\frac{C \vee A \vee \ldots \vee A}{C \vee A}$$

where (i) A is strictly maximal with respect to C.

Ordered resolution:
$$\frac{C \vee A \quad D \vee \neg A}{C \vee D}$$

where (i) A is strictly maximal with respect to C, (ii) $\neg A$ is maximal with respect to D.

Concrete domain resolution: $$\frac{C_1 \vee d_1(\mathbf{t_1}) \quad \ldots \quad C_n \vee d_n(\mathbf{t_n})}{C_1 \vee \ldots \vee C_n}$$

where (i) $d_i(\mathbf{t_i})$ are strictly maximal with respect to C_i, (ii) the set $S = \{d_i(\mathbf{t_i})\}$ is a minimal $\mathbf{D}$-constraint.

In $\mathcal{G}^{\mathbf{D}}$, the clauses $C \vee A \vee \ldots \vee A$ and $D \vee \neg A$ are called the *main premises*, whereas the clauses $C \vee A$ and $C_i \vee d_1(\mathbf{t_1})$ are called the *side premises*. Notice that, under this definition, the concrete domain resolution rule does not have a main premise.

It is well-known that effective redundancy elimination criteria are necessary for theorem proving to be applicable in practice. A powerful *standard notion of redundancy* was introduced in [14]. We adapt this notion slightly to take into account the fact that the concrete domain resolution rule does not have a main premise.

Definition 4.1.3. *Let N be a set of ground clauses. A ground clause C is* redundant *in N if clauses $D_i \in N$ exist, such that $C \succ D_i$ and $D_1, \ldots, D_m \models C$. A ground inference ξ with side premises C_i and a conclusion D is* redundant *in N if clauses $D_i \in N$ exist, such that $C_1, \ldots, C_n, D_1, \ldots, D_m \models D$; if ξ has a main premise C, then additionally $C \succ D_i$.*

We now prove the soundness and completeness of $\mathcal{G}^{\mathbf{D}}$ under the standard notion of redundancy.

Lemma 4.1.4 (Soundness)**.** *Let N be a set of ground clauses, I a $\mathbf{D}$-model of N, and $N' = N \cup \{C\}$, where C is the conclusion of an inference by $\mathcal{G}^{\mathbf{D}}$ with premises from N. Then I is a $\mathbf{D}$-model of N'.*

Proof. For an inference by positive factoring or ordered resolution, soundness is trivial and is shown in the same way as in [14]. Let C be obtained by the concrete domain resolution rule, with S being as in the rule definition. Since S is a $\mathbf{D}$-constraint, I is a $\mathbf{D}$-model of N only if there exists a literal $d_i(\mathbf{t_i}) \in S$, such that $d_i(\mathbf{t_i}) \notin I$. Since $C_i \vee d_i(\mathbf{t_i})$ is by assumption true in I, some literal from C_i is true in I. Since $C_i \subseteq C$, C is true in I as well. □

Lemma 4.1.5 (Completeness)**.** *Let N be a set of ground clauses such that each inference by $\mathcal{G}^{\mathbf{D}}$ from premises in N is redundant in N. If N does not contain the empty clause, then N is $\mathbf{D}$-satisfiable.*

Proof. We extend the model building method from [14] to handle the concrete domain resolution rule. For a set of ground clauses N, we define an interpretation I by induction on the clause ordering $\succ$ as follows: for some clause C, we set $I_C = \bigcup_{C \succ D} \varepsilon_D$, where $\varepsilon_D = \{A\}$ if (i) $D \in N$, (ii) D is of the form $D' \vee A$, such that A is strictly maximal with respect to D', and (iii) D is false in I_D; otherwise, $\varepsilon_D = \emptyset$. Let $I = \bigcup_{D \in N} \varepsilon_D$. A clause D such that $\varepsilon_D = \{A\}$ is called *productive*, and it is said to *produce* the atom A in I. Before proving the lemma, we show the following three properties.

Invariant (*): if C is false in $I_C \cup \varepsilon_C$, then C is false in I. Since C is false in I_C, all negative literals from C are false in I_C, and since $I_C \subseteq I$, all negative literals

from C are false in I as well. Furthermore, since $\neg A \succ A$, any atom produced by a clause $D \succ C$ is larger than any literal occurring in C, so no clause greater than C can produce an atom that will make C true.

Invariant (**): if C is true in $I_C \cup \varepsilon_C$, then C is true in I. If C is true in $I_C \cup \varepsilon_C$ because some positive literal is true in $I_C \cup \varepsilon_C$, since $I_C \cup \varepsilon_C \subseteq I$, C is true in I as well. Otherwise, C can be true in $I_C \cup \varepsilon_C$ because some negative literal $\neg A$ is true in $I_C \cup \varepsilon_C$. Since $\neg A \succ A$, any atom produced by a clause $D \succ C$ is larger than any literal occurring in C. Hence, no such clause D can produce A, so $\neg A$ is true in I as well. We often use this invariant in its contrapositive form: if C is false in I, it is false in $I_C \cup \varepsilon_C$.

Property (***): if an inference ξ with a main premise C, side premises C_i and a conclusion D is redundant in N, then there are clauses $D_i \in N$ which are not redundant in N, such that $D_1, \ldots, D_m, C_1, \ldots, C_n \models D$ and $C \succ D_i$. If ξ is redundant, by definition of the standard notion of redundancy, there are clauses $D'_j \in N$ such that $D'_1, \ldots, D'_m, C_1, \ldots, C_n \models D$ and $C \succ D'_i$. Namely, if some D'_j is redundant, then there are clauses D''_k such that $D''_1, \ldots, D''_{m'} \models D'_j$ and $D'_j \succ D''_k$. Since $\succ$ is well-founded, this process can be continued recursively until we obtain the set of smallest non-redundant clauses for which (***) obviously holds. This property holds analogously if ξ does not have a main premise.

We now show that, if all inferences by $\mathcal{G}^{\mathbf{D}}$ from premises in N are redundant in N, then I is a $\mathbf{D}$-model of N. The proof is by contradiction: let us assume that I is not a $\mathbf{D}$-model of N. There may be two causes for that:

- There is a clause $C \in N$ which is false in I; such a clause is called a *counterexample* for I. Since $\succ$ is well-founded and total, we may assume w.l.o.g. that C is the smallest counterexample. C is obviously not productive, since all productive clauses are true in I. C can be non-productive and false in I if it has one of the following two forms:

  - $C = C' \vee A \vee \ldots \vee A$. Then, C is not productive since A is not strictly maximal in C. Since C is false in I, by (**) it is false in $I_C \cup \varepsilon_C$. Since $\varepsilon_C = \emptyset$, C' is false in I_C, and by (*) C' is false in I. Since N is saturated, the inference by positive factoring resulting in $D = C' \vee A$ is redundant in N. D is obviously false in I. By the fact that N is saturated and by (***), a set of clauses $D_i \in N$ exists, such that $C \succ D_i$ and $D_1, \ldots, D_n \models D$. Since C is the smallest counterexample, all D_i are true in I, but then D is true in I as well, which is a contradiction.
  - $C = C' \vee \neg A$. Since C is false in I, it must hold that $A \in I$, which is produced by some smaller clause $D = D' \vee A$. Similarly as in the previous case, C' is false in I. Since D is productive, D' is false in I_D, and since A is strictly maximal with respect to D', by (*) D' is false in I. Since N is saturated, the inference by ordered resolution resulting in $E = C' \vee D'$ is redundant in

N. E is obviously false in I. By the fact that N is saturated and by (***), a set of clauses $D_i \in N$ exists, such that $C \succ D_i$ and $D_1, \ldots, D_n, D' \vee A \models E$. Since C is the smallest counterexample, all D_i are true in I, and since $D' \vee A$ is productive, it is also true in I. Hence, E is true in I, which is a contradiction.

- All clauses from N are true in I, but I contains a minimal set of concrete domain literals $S = \{d_i(\mathbf{t_i})\}$ such that $\widehat{S}$ is $\mathbf{D}$-unsatisfiable. The literals from S must have been produced by clauses $E_i = C_i \vee d_i(\mathbf{t_i}) \in N$, where E_i is false in I_{E_i}. For any i, since $d_i(\mathbf{t_i})$ is strictly maximal with respect to C_i, C_i is false in $I_{E_i} \cup \varepsilon_{E_i}$, and by (*) C_i is false in I. Since N is saturated, the inference by concrete domain resolution resulting in $D = C_1 \vee \ldots \vee C_n$ is redundant in N. Obviously, D is false in I. Since the inference is redundant, by property (***), non-redundant clauses $D_i \in N$ exist, such that $D_1, \ldots, D_m, C_1 \vee d_1(\mathbf{t_1}), \ldots, C_n \vee d_n(\mathbf{t_n}) \models D$. All D_i and $C_i \vee d_i(\mathbf{t_i})$ are by assumption true in I, implying that D is true in I, which is a contradiction.

Hence, I is a $\mathbf{D}$-model of N, so N is $\mathbf{D}$-satisfiable. □

A *derivation* by $\mathcal{G}^{\mathbf{D}}$ from a set of clauses N_0 is a sequence of clause sets $N_0, N_1, \ldots$ where $N_i = N_{i-1} \cup \{C\}$ and C is a consequence of some inference rule of $\mathcal{G}^{\mathbf{D}}$ from premises in N_{i-1}, or $N_i = N_{i-1} \setminus \{C\}$ where C is redundant in N_{i-1}. A derivation is *fair* with *limit* $N_\infty = \bigcup_j \bigcap_{k \geq j} N_k$, if each clause C that can be deduced from non-redundant premises in N_∞ is contained in some set N_j. In [14] it was shown that under the standard notion of redundancy, each inference from premises in N_∞ is redundant in N_∞. From this and lemmata 4.1.4 and 4.1.5, we get the following result:

Theorem 4.1.6. *The set of ground clauses N is $\mathbf{D}$-unsatisfiable if and only if the limit N_∞ of a fair derivation by $\mathcal{G}^{\mathbf{D}}$ contains the empty clause.*

4.1.2 Most General Partitioning Unifiers

Lifting the concrete domain resolution rule to general clauses is not trivial, since unification can only partly guide the rule application. Consider, for example, the set of clauses $N = \{d_1(x_1, y_1), d_2(x_2, y_2)\}$. N has ground instances $d_1(a_1, b_1)$ and $d_2(a_2, b_2)$, which do not share a common term. Hence, a conjunction consisting of these literals is not connected, so by Lemma 4.1.2 it cannot be minimal and the conditions of the concrete domain resolution are not satisfied. However, clauses $d_1(a, b_1)$ and $d_2(a, b_2)$ are also ground instances of N, but they do share common terms. Hence, a conjunction of these literals is connected, so the conditions of the concrete domain resolution rule should be checked. This is the consequence of the fact that it is possible to unify x_1 and x_2 from $d_1(x_1, y_1)$ and $d_2(x_2, y_2)$. Another possibility is to unify e.g. x_2 and y_1. Even for a set consisting of a single clause, such as $M = \{d(x, y)\}$, it is possible to obtain a $\mathbf{D}$-constraint $d(x, x)$ by unifying x and y. These examples show that, to check

all potential **D**-constraints at the ground level, one has to consider all possible substitutions which produce a minimal **D**-constraint at the non-ground level. We formalize this idea by the following definition.

Definition 4.1.7. *Let $S = \{d_i(\mathbf{t_i})\}$ be a multiset of positive concrete domain literals. A substitution σ is a* partitioning unifier *of S if the set $S\sigma$ is connected. Furthermore, σ is a* most general partitioning unifier *if, for any partitioning unifier θ such that $\widehat{S\sigma} \equiv \widehat{S\theta}$, a substitution η exists such that $\theta = \sigma\eta$. With $\mathsf{MGPU}(S)$ we denote the set of all most general partitioning unifiers of S.*

Note that in Definition 4.1.7 we assume S is a multiset, so it can contain repeated literals. If no terms from literals in S are unifiable, a most general partitioning unifier of S does not exist. Furthermore, the previous example shows that several most general partitioning unifiers of S may exist. However, for a given $\widehat{S\sigma}$, all most general partitioning unifiers are identical up to variable renaming, as demonstrated next.

Let $S = \{d_i(\mathbf{t_i})\}$ be a multiset of positive concrete domain literals, and C a conjunction over literals in S, obtained by replacing terms of each $d_i(\mathbf{t_i})$ with arbitrary variables (it is not necessary to use different variables in C for different terms from S, but the same term in S should always be replaced with the same variable in C). For such a C, let m be the number of distinct variables in C; for each such variable x_i, $1 \leq i \leq m$, let T_{x_i} denote the set of all terms occurring in a literal in S at a position corresponding to an occurrence of x_i in C; finally, let $n = \max |T_{x_i}|$. With S_C we denote the set of terms $t_j = f(s_j^1, \ldots, s_j^m)$, where s_j^i is the j-th term of T_{x_i} if $j \leq |T_{x_i}|$, and a fresh variable if $j > |T_{x_i}|$, for $1 \leq j \leq n$. Most general partitioning unifiers of S and most general unifiers of S_C are closely related, as demonstrated next.

Lemma 4.1.8. *Let θ be a partitioning unifier of a multiset of concrete domain literals $S = \{d_i(\mathbf{t_i})\}$. Then the substitution $\sigma = \mathsf{MGU}(S_C)$, where $C = \widehat{S\theta}$, is the most general partitioning unifier of S, unique up to variable renaming, such that $\widehat{S\sigma} \equiv \widehat{S\theta}$.*

Proof. To obtain the variable x_i in C, θ must be such that $s_1^i\theta = \ldots = s_n^i\theta = T_{x_i}\theta$. Furthermore, $x_i \neq x_j$ for $i \neq j$, so $T_{x_i}\theta \neq T_{x_j}\theta$. Because of the first property, θ is obviously a unifier of S_C, so it is well-known [8] that $\sigma = \mathsf{MGU}(S_C)$ exists and is unique up to variable remaining. Furthermore, it is obvious that $s_1^i\sigma = \ldots = s_n^i\sigma = T_{x_i}\sigma$ and $T_{x_i}\sigma \neq T_{x_j}\sigma$ for $i \neq j$. Namely, since σ is the most general unifier of S_C, then a substitution η exists, such that $\theta = \sigma\eta$. Hence, it is impossible that $T_{x_i}\sigma = T_{x_j}\sigma$ and $T_{x_i}\sigma\eta \neq T_{x_j}\sigma\eta$. Hence, $\widehat{S\sigma} \equiv \widehat{S\theta}$, so the claim of the lemma follows. □

Lemma 4.1.9. *For a multiset of concrete domain literals $S = \{d_i(\mathbf{t_i})\}$, $\mathsf{MGPU}(S)$ consists exactly of all $\mathsf{MGU}(S_C)$, where C is a conjunction over literals of S.*

Proof. For $\sigma \in \mathsf{MGPU}(S)$, $C = \widehat{S\sigma}$ is obviously a conjunction over literals of S satisfying conditions of Lemma 4.1.8, so σ is equivalent to $\mathsf{MGU}(S_C)$ up to variable renaming. Conversely, let C be a conjunction over literals of S. Now $\sigma = \mathsf{MGU}(S_C)$ is obviously a partitioning unifier of S. Let $C' = \widehat{S\sigma}$. Observe that $C \equiv C'$ does not necessarily

hold: it is possible that $T_{x_i}\sigma = T_{x_j}\sigma, i \neq j$. However, C' satisfies conditions of Lemma 4.1.8, so $\mathsf{MGU}(S_{C'})$ exists, and is a most general partitioning unifier of S. □

Hence, Lemma 4.1.9 gives a brute-force algorithm for computing $\mathsf{MGPU}(S)$: one should systematically examine all connected conjunctions C over literals in S. The main performance drawback of this algorithm is that one must examine all such conjunctions C. For n literals in S of maximal arity m, the number of possible assignments of variables in C is bounded by $(nm)^{nm}$, which is obviously exponential. However, for certain logics, it is possible to construct a specialized, but much more efficient algorithm. In Chapter 4, we present such an algorithm applicable to $\mathcal{SHIQ}(\mathbf{D})$.

4.1.3 Concrete Domain Resolution with General Clauses

As usual in a resolution setting, we assume that all clauses involved in an inference rule do not share common variables. The inference rules of the *concrete domain resolution* calculus, $\mathcal{R}^{\mathbf{D}}$ for short, are presented below.

Positive factoring:
$$\frac{C \vee A \vee B}{C\sigma \vee A\sigma}$$

where (i) $\sigma = \mathsf{MGU}(A, B)$, (ii) $A\sigma$ is strictly maximal with respect to $C\sigma$.

Ordered resolution:
$$\frac{C \vee A \quad D \vee \neg B}{C\sigma \vee D\sigma}$$

where (i) $\sigma = \mathsf{MGU}(A, B)$, (ii) $A\sigma$ is strictly maximal with respect to $C\sigma$, (iii) $\neg B\sigma$ is maximal with respect to $D\sigma$.

Concrete domain resolution:
$$\frac{C_1 \vee d_1(\mathbf{t_1}) \quad \ldots \quad C_n \vee d_n(\mathbf{t_n})}{C_1\sigma \vee \ldots \vee C_n\sigma}$$

where (i) clauses $C_i \vee d_i(\mathbf{t_i})$ are not necessarily unique, (ii) for the set $S = \{d_i(\mathbf{t_i})\}$, $\sigma \in \mathsf{MGPU}(S)$, (iii) $d_i(\mathbf{t_i})\sigma$ are strictly maximal with respect to $C_i\sigma$, (iv) the set $S\sigma$ is a minimal $\mathbf{D}$-constraint.

We briefly comment on the constraint (i) of the concrete domain resolution rule. Consider a clause $d(x, y)$. It is possible that, for the set $S = \{d(a, b), d(b, c)\}$ of ground instances of $d(x, y)$, the conjunction $\widehat{S}$ is $\mathbf{D}$-unsatisfiable. This is detected by the concrete domain resolution rule only if several "copies" of the clause $d(x, y)$ are considered simultaneously. Without any assumptions on the nature of the predicate d, there is no upper bound on the number of "copies" that should be considered simultaneously. This property of the calculus obviously leads to undecidability in the general case. To obtain a decision procedure for $\mathcal{SHIQ}(\mathbf{D})$, in Section 4.2 we show that the number of such "copies" that must be considered is bounded.

To prove the completeness of $\mathcal{R}^{\mathbf{D}}$, we show now that for each ground derivation, there is a corresponding non-ground derivation.

Lemma 4.1.10 (Lifting)**.** *Let N be a set of clauses with the set of ground instances N^G. For each ground inference ξ^G by $\mathcal{G}^{\mathbf{D}}$ applicable to premises $C_i^G \in N^G$, there is an inference ξ by $\mathcal{R}^{\mathbf{D}}$ applicable to premises $C_i \in N$, where ξ^G is an instance of ξ.*

Proof. Let C_i^G be ground premises from N^G participating in ξ^G, resulting in a ground clause D^G. The ground inference ξ^G of $\mathcal{G}^{\mathbf{D}}$ can be simulated by a corresponding non-ground inference ξ, where for each ground premise C_i^G we take the corresponding non-ground premise C_i that C_i^G is an instance of. Since all C_i are variable-disjoint, there is a ground substitution τ such that $C_i^G = C_i\tau$. Let us denote with D the result of ξ on C_i. We now show that ξ is an inference of $\mathcal{R}^{\mathbf{D}}$.

Let ξ^G be an inference by positive factoring on ground literals A_i^G of C^G. Substitution τ is obviously a unifier of corresponding non-ground literals A_i of C. Since any unifier is an instance of the most general unifier σ of A_i, a substitution η exists such that $\tau = \sigma\eta$. Furthermore, if A_i^G is strictly maximal with respect to C^G, since $\succ$ is a reduction ordering, corresponding A_i is strictly maximal with respect to $C\sigma$, so $D^G = D\eta$. Hence, ξ is an inference of $\mathcal{R}^{\mathbf{D}}$. Similar reasoning applies in the case of ordered resolution.

Let ξ^G be an inference by concrete domain resolution, and let $S = \{d_i(\mathbf{t_i})\}$ be the set of corresponding non-ground literals. Since $S\tau$ is a minimal $\mathbf{D}$-constraint, by Lemma 4.1.2, $S\tau$ is connected, so τ is obviously a partitioning unifier of S. Then, by Lemma 4.1.8, there exists some most general partitioning unifier σ such that $\tau = \sigma\eta$ for some η, and $\widehat{S\sigma} \equiv \widehat{S\tau}$. Obviously, if $S\tau$ is a minimal $\mathbf{D}$-constraint, so is $S\sigma$. Furthermore, if literals from $S\tau$ are strictly maximal with respect to C_i^G, since $\succ$ is a reduction ordering, corresponding literals from $S\sigma$ are strictly maximal with respect to $C_i\sigma$, so $D^G = D\eta$. Hence, ξ is an inference of $\mathcal{R}^{\mathbf{D}}$. □

The notion of redundancy is lifted to the non-ground case as usual [14]: a clause C (an inference ξ) is redundant in a set of clauses N if all ground instances of C (ξ) are redundant in N^G. This is enough for soundness and completeness of $\mathcal{R}^{\mathbf{D}}$.

Theorem 4.1.11. *The set of clauses N is $\mathbf{D}$-unsatisfiable if and only if the limit N_∞ of a fair derivation by $\mathcal{R}^{\mathbf{D}}$ contains the empty clause.*

Proof. The set N is unsatisfiable if and only if the set of its ground instances N^G is unsatisfiable. By Theorem 4.1.6, N^G is unsatisfiable if and only if there is a ground derivation $N^G = N_0^G, N_1^G, \ldots, N_\infty^G$ where the limit N_∞^G contains the empty clause. By Lemma 4.1.10 and the definition of the redundancy for non-ground clauses, for each ground derivation, a non-ground derivation $N = N_0, N_1, \ldots, N_\infty$ exists, where each N_i^G is a subset of ground instances of N_i. Hence, N_∞^G contains the empty clause if and only if N_∞ contains the empty clause, so the claim of the theorem follows. □

4.1.4 Combining Concrete Domains with Other Resolution Calculi

In order not to make the presentation too technical, we extended only the ordered resolution calculus with the concrete domain resolution rule. However, from the proof

of Lemma 4.1.5, one may see that the concrete domain resolution rule is largely independent from the actual calculus, and may be combined with other calculi whose completeness proof is based on the model generation method [14]. For example, it is straightforward to extend $\mathcal{R}^{\mathbf{D}}$ with selection of negative literals. Similarly, concrete domain can be combined with an equational theorem proving calculus, such as $\mathcal{BS}$; we denote thus obtained calculus with $\mathcal{BS}^{\mathbf{D}}$.

To apply the concrete domain resolution rule to other calculi, the premises of the concrete domain resolution rule must be those clauses which can have at least one productive ground instance. For the ordered resolution with selection, this means that premises are not allowed to contain selected literals (since such clauses do not have productive ground instances). The usual arguments for the calculus at hand show that, if N_∞ does not contain the empty clause, one may generate an interpretation I using the model generation method, such that all clauses from N_∞ are true in I. Furthermore, the argument from the second part of Lemma 4.1.5 shows independently that, if all inferences by the concrete domain resolution rule are redundant in N_∞, then I is a $\mathbf{D}$-model of N.

4.2 Deciding $\mathcal{SHIQ}(\mathbf{D})$

In this section we combine the concrete domain resolution rule from Section 4.1 with the algorithm from Chapter 3 to obtain a decision procedure for checking satisfiability of $\mathcal{SHIQ}(\mathbf{D})$ knowledge bases. We also show that this extension does not increase the complexity of reasoning, assuming a bound on the arity of concrete predicates.

It is easy to see that the operator Ω from Section 3.2 can be used to eliminate transitivity axioms from a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB by encoding in into an equisatisfiable knowledge base $\Omega(KB)$. Namely, concrete roles cannot be transitive, so Theorem 3.2.3 applies without changes. Hence, without loss of generality, in the rest of this section we focus on deciding satisfiability of $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge bases. This we achieve in two steps.

We first derive a decision procedure for $\mathcal{ALCHIQ}^-(\mathbf{D})$. With $\mathcal{BS}^{\mathbf{D}}_{DL}$ we denote the $\mathcal{BS}$ calculus extended with the concrete domain resolution rule, parameterized as specified in Definition 3.3.3. To obtain a decision procedure for $\mathcal{ALCHIQ}^-(\mathbf{D})$, we show that the results of Lemma 3.3.5 remain valid when $\mathcal{ALCHIQ}^-(\mathbf{D})$-closures are saturated under $\mathcal{BS}^{\mathbf{D}}_{DL}$. In particular, we show that the application of the concrete domain resolution rule does not lead to generation of terms of arbitrary depth. Furthermore, we show that the maximal length of a $\mathbf{D}$-constraint to be considered is polynomial in $|KB|$, assuming a limit on the arity of concrete predicates, so the complexity of reasoning does not increase.

Next, to obtain a decision procedure for full $\mathcal{ALCHIQ}(\mathbf{D})$, in Subsection 4.2.4 we simply observe that the decomposition rule from Section 3.4 applies without change.

Table 4.1: Additional Closures after Preprocessing

12	$\neg T(x, y^{\mathsf{c}}) \vee U(x, y^{\mathsf{c}})$
13	$\bigvee(\neg)C_i(x) \vee T(x, f^{\mathsf{c}}(x))$
14	$\bigvee(\neg)C_i(x) \vee d(f_1^{\mathsf{c}}(x), \ldots, f_m^{\mathsf{c}}(x))$
15	$\bigvee(\neg)C_i(x) \vee f_i^{\mathsf{c}}(x) \not\approx f_j^{\mathsf{c}}(x)$
16	$\bigvee(\neg)C_i(x) \vee \bigvee \neg T_i(x, y_i^{\mathsf{c}}) \vee d(y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}})$
17	$\bigvee(\neg)C_i(x) \vee \bigvee_{i=1}^{n} \neg T(x, y_i^{\mathsf{c}}) \vee \bigvee_{i,j=1;j>i}^{n} y_i^{\mathsf{c}} \approx y_j^{\mathsf{c}}$
18	$(\neg)T(a, b^{\mathsf{c}})$
19	$a^{\mathsf{c}} \approx b^{\mathsf{c}}$
20	$a^{\mathsf{c}} \not\approx b^{\mathsf{c}}$

4.2.1 Closures with Concrete Predicates

As before, with $\Xi(KB)$ we denote the set of closures obtained from KB by structural transformation, as explained in Definition 3.3.1. By definition of π from Table 2.1 and Table 2.3, it is easy to see that $\Xi(KB)$ may contain closures with structure as in Table 3.1, with all variables, function symbols and predicate arguments being of the sort a, and additionally closures with structure as in Table 4.1.

We now generalize the closures from Table 4.1 to include closure types produced in a saturation of $\Xi(KB)$ by $\mathcal{BS}_{DL}^{\mathbf{D}}$. So called $\mathcal{ALCHIQ}^{-}(\mathbf{D})$-closures include those with structure as in Table 3.2 and in Table 4.2. By the definition of Ξ, it is obvious that Lemma 3.3.2 and Lemma 3.3.4 hold for $\mathcal{ALCHIQ}^{-}(\mathbf{D})$-closures as well. To make a distinction with generators of type 3, we call closures of type 10 where $f^{\mathsf{c}}(x)$ occurs unmarked c-*generators*.

Table 4.2: Types of $\mathcal{ALCHIQ}^{-}(\mathbf{D})$-closures

8	in addition to literals from Table 3.2, a closure can additionally contain $\ldots \vee \mathbf{T}(\langle \mathbf{a} \rangle, \langle \mathbf{b}^{\mathsf{c}} \rangle) \vee \mathbf{d}(\langle \mathbf{t_1^c} \rangle, \ldots, \langle \mathbf{t_n^c} \rangle) \vee \bigvee \langle t_i^{\mathsf{c}} \rangle \approx / \not\approx \langle {t^{\mathsf{c}}}_j \rangle$ where t_i^{c} and t_j^{c} are either some constant b^{c} or a functional term $f_i^{\mathsf{c}}([a])$
9	$\neg T(x, y^{\mathsf{c}}) \vee U(x, y^{\mathsf{c}})$
10	$\mathbf{P}^{\mathbf{f^c}}(x) \vee T(x, \langle f^{\mathsf{c}}(x) \rangle)$
11	$\mathbf{P}(x) \vee \mathbf{d}(\langle \mathbf{f_1^c}(x) \rangle, \ldots, \langle \mathbf{f_n^c}(x) \rangle) \vee \bigvee \langle f_i^{\mathsf{c}}(x) \rangle \approx / \not\approx \left\langle {f^{\mathsf{c}}}_j(x) \right\rangle$
12	$\mathbf{P}(x) \vee \bigvee \neg T(x, y_i^{\mathsf{c}}) \vee \bigvee y_i^{\mathsf{c}} \approx y_j^{\mathsf{c}}$
13	$\mathbf{P}(x) \vee \bigvee \neg T_i(x, y_i^{\mathsf{c}}) \vee d(y_1^{\mathsf{c}}, \ldots, y_n^{\mathsf{c}})$
Note: Conditions from Table 3.2 apply analogously.	

4.2.2 Closure of $\mathcal{ALCHIQ}^-(\mathbf{D})$-closures under Inferences

We now extend Lemma 3.3.5 to handle $\mathcal{ALCHIQ}^-(\mathbf{D})$-closures.

Lemma 4.2.1. *Let $\Xi(KB) = N_0, \ldots, N_i \cup \{C\}$ be a $\mathcal{BS}^{\mathbf{D}}_{DL}$-derivation, where C is the conclusion derived from premises in N_i. Then C is either an $\mathcal{ALCHIQ}^-(\mathbf{D})$-closure or it is redundant in N_i.*

Proof. Since names and sorts of the abstract and concrete domain predicates are disjoint, and since in closures of type 11 literals with concrete predicates or concrete equalities are always maximal, an inference between closures of types 1–7 and 9–13 is not possible. Furthermore, a closure of type 8 can participate in an inference with a closure of type 1–7 only on abstract, and with a closure of type 9–13 only on concrete predicates. Hence, the proof of Lemma 3.3.5 remains valid for closures of types 1–8.

Similarly as in the proof of Lemma 3.3.5, one can show that a superposition into a c-generator is redundant. This follows from the fact that by Condition (i), each $\mathcal{ALCHIQ}^-(\mathbf{D})$-closure containing $f^{\mathsf{c}}(x)$ contains $\mathbf{P}^{\mathbf{f^c}}(x)$ as well, and that for each equality $[f_i^{\mathsf{c}}(x)] \approx \left[{f^{\mathsf{c}}}_j(x)\right]$, $\mathsf{role}(f_i^{\mathsf{c}}) = \mathsf{role}(f_j^{\mathsf{c}})$, so each superposition into a generator is subsumed by some other generator or a witness closure. For other basic superposition inferences, the claim of the lemma can be trivially demonstrated.

The most important difference to the proof of Lemma 3.3.5 lies in the application of the concrete domain resolution rule, with n side premises of the form $C_i \vee d_i(\langle \mathbf{t_i} \rangle)$. For side premises of type 11, we denote their free variables with x_i. Let $S = \{d_i(\mathbf{t_i})\}$. For any most general partitioning unifier σ of S, $S\sigma$ must be connected. Hence, there are two possibilities:

- If all side premises are of type 11, σ is of the form $\{x_2 \mapsto x_1, \ldots, x_n \mapsto x_1\}$. The result is obviously a closure of type 11.
- Assume there is a side premise of type 8 and that $S\sigma$ is connected. If $S\sigma$ would contain a variable x_i, then the literal $d_i(\mathbf{f^c_i}(x_i))$ would not contain a ground term, so $S\sigma$ would not be connected. Hence, all literals from $S\sigma$ are ground, and σ is of the form $\{x_1 \mapsto c_1, \ldots, x_n \mapsto c_n\}$. The result is a closure of type 8.

Hence, all non-redundant inferences of $\mathcal{BS}^{\mathbf{D}}_{DL}$ applied to $\mathcal{ALCHIQ}^-(\mathbf{D})$-closures produce an $\mathcal{ALCHIQ}^-(\mathbf{D})$-closure. □

By inspecting the proof of Lemma 4.2.1, one may easily extend the Corollary 3.3.6 to include $\mathcal{ALCHIQ}^-(\mathbf{D})$ closures:

Corollary 4.2.2. *If a closure of type 8 participates in a $\mathcal{BS}^{\mathbf{D}}_{DL}$ inference in a derivation from Lemma 4.2.1, the unifier σ contains only ground mappings and the conclusion is a closure of type 8. Furthermore, a closure of type 8 cannot participate in an inference with a closure of type 4 or 6.*

4.2.3 Termination and Complexity Analysis

Establishing termination and determining the complexity is slightly more difficult. Let m denote the maximal arity of a concrete domain predicate, d the number of concrete domain predicates, and f the number of function symbols. As in Lemma 3.3.8, f is linear in $|KB|$ for unary coding of numbers, since by skolemizing $\exists T_1, \ldots, T_m.d$ we introduce m function symbols. In addition to literals from $\mathcal{ALCHIQ}^-$-closures, $\mathcal{ALCHIQ}^-(\mathbf{D})$-closures may contain literals of the form $d(\langle f_1(x)\rangle, \ldots, \langle f_m(x)\rangle)$. The maximal non-ground closure will contain all such literals, of which there can be $d(2f)^m$ many (the factor 2 takes into account that each functional term can be marked or not). Hence, there are $2^{d(2f)^m}$ different combinations of concrete domain literals. This presents us with a problem: in general, m is linear in $|KB|$, so the number of closures becomes doubly-exponential, thus invalidating the results of Lemma 3.3.8.

A possible solution to this problem is to assume a bound on the arity of concrete predicates. This is justifiable from a practical point of view: it is hard to imagine a practically useful concrete domain with predicates of unbounded arity as well as a sequence of concrete domains with increasing arities, but where the arity of each domain is bounded. In this case, m becomes a constant, and does not depend on $|KB|$. The maximal length of a closure is then polynomial in $|KB|$, and the number of closures is exponential in $|KB|$. Hence, Lemma 3.3.8 can be easily extended to include $\mathcal{ALCHIQ}^-(\mathbf{D})$-closures. The following lemma shows the last step in determining the complexity of the decision algorithm.

Lemma 4.2.3. *The maximal number of side premises participating in a concrete domain resolution rule in a $\mathcal{BS}^{\mathbf{D}}_{DL}$ derivation from Lemma 4.2.1 is at most polynomial in $|KB|$, assuming a limit on the arity of concrete predicates.*

Proof. Let $S = \{d_i(\mathbf{t_i})\}$ be the multiset of maximal concrete domain literals of n side premises $C_i \vee d_i(\mathbf{t_i})$, and let σ be a most general partitioning unifier of S. Obviously, $S\sigma$ should not contain repeated literals $d_i(\mathbf{t_i})\sigma$; otherwise $\widehat{S\sigma}$ contains repeated conjuncts and is not minimal. Hence, the longest set $S\sigma$ is the one where each $d_i(\mathbf{t_i})\sigma$ is distinct.

Let m be the maximal arity of a concrete domain predicate, f the number of function symbols, d the number of concrete domain predicates, and c the number of constants in the signature of $\Xi(KB)$. Then there are at most $\ell_{ng} = df^m$ distinct non-ground concrete domain literals, and at most $\ell_g = d(c+cf)^m$ distinct ground concrete domain literals. Assuming a bound on m and for unary coding of numbers, both ℓ_{ng} and ℓ_g are polynomial in $|KB|$.

If all side premises are non-ground, since $S\sigma$ should be connected, the only possible form σ can take is $\{x_2 \mapsto x_1, \ldots, x_n \mapsto x_1\}$. Hence, $S\sigma$ contains only one variable x_1, so the maximal number of distinct literals in $S\sigma$ is ℓ_{ng}. Thus, the maximal number of non-ground side premises n to be considered is bounded by ℓ_{ng}, and all side premises are unique up to variable renaming.

If there is a ground side premise, since $S\sigma$ should be connected, σ can be of the form $\{x_1 \mapsto c_1, \ldots, x_n \mapsto c_n\}$. All literals in $S\sigma$ are ground, so the maximum number

of distinct literals in $S\sigma$ is ℓ_g. Thus, the maximal number of side premises n (by counting each "copy" of a premise separately) to be considered is bounded by ℓ_g. □

Theorem 4.2.4. *Let KB be an $\mathcal{ALCHIQ}^-(\mathbf{D})$ knowledge base, defined over an admissible concrete domain $\mathbf{D}$, for which $\mathbf{D}$-satisfiability of finite conjunctions over $\Phi_{\mathbf{D}}$ can be decided in deterministic exponential time. Then saturation of $\Xi(KB)$ by $\mathcal{BS}^{\mathbf{D}}_{DL}$ with eager application of redundancy elimination rules decides $\mathbf{D}$-satisfiability of KB and runs in time exponential in $|KB|$, for unary coding of numbers and assuming a bound on the arity of concrete predicates.*

Proof. As already explained, a polynomial bound on the length, and an exponential bound on the number of $\mathcal{ALCHIQ}^-(\mathbf{D})$-closures follows in the same way as in Lemma 3.3.8. By substituting Lemma 3.3.5 for Lemma 4.2.1, the proof of the Theorem 3.3.9 can straightforwardly be extended to the case of $\mathcal{ALCHIQ}^-(\mathbf{D})$ knowledge bases, for all inferences apart from the concrete domain resolution rule. The only remaining problem is to show that, in applying the concrete domain resolution rule, the number of satisfiability checks of conjunctions over $\mathbf{D}$ is exponential in $|KB|$, and that the length of each conjunction is polynomial in $|KB|$.

To apply the concrete domain resolution rule to a set of closures N, a subset $N' \subseteq N$ must be selected, for which maximal concrete domain literals are unique up to variable renaming. By Lemma 4.2.3, $\ell = |N'|$ is polynomial in $|KB|$, and N' contains at most ℓ variables. If all closures from N' are non-ground, there is exactly one substitution σ which unifies all of these variables. If there is at least one ground closure, then each one of ℓ variables can be assigned to one of c constants, producing c^ℓ combinations, which is exponential in $|KB|$. Closures in N' are chosen from the maximal set of all closures which is exponential in $|KB|$, and since $|N'|$ is polynomial in $|KB|$, the number of different sets N' is exponential in $|KB|$. Hence, the maximal number of $\mathbf{D}$-constraints that should be examined by the concrete domain resolution rule is exponential in $|KB|$, where the length of each $\mathbf{D}$-constraint is polynomial in $|KB|$. Under the assumption that the satisfiability of each $\mathbf{D}$-constraint can be checked in deterministic exponential time, all inferences by the concrete domain resolution rule can be performed in exponential time, so the claim of the theorem follows. □

The proof of Lemma 4.2.3 demonstrates how to implement the concrete domain resolution rule in practice. There are two cases:

- For concrete domain resolution on non-ground closures, the most general partitioning unifier is of the form $\sigma = \{x_2 \mapsto x_1, \ldots, x_n \mapsto x_1\}$. Hence, one should consider only closures with maximal literals unique up to variable renaming, so several "copies" of a closure need not be considered.

- For concrete domain resolution where at least one closure is ground, one first chooses a connected set of ground closures Δ. Let Δ_c be the set of constants, and Δ_f the set of function symbols occurring in a ground functional term $f(c)$ in a closure from Δ. The most general partitioning unifier σ may contain only

mappings of the form $x_i \mapsto c$, where $c \in \Delta_c$. Hence, one selects the set Δ_{ng} of non-ground closures containing a function symbol from Δ_f. To apply the concrete domain resolution rule, one considers at most $|\Delta_c|$ "copies" of a closure from Δ_{ng}, since σ might assign a distinct value from Δ_c to each such closure.

4.2.4 Deciding $\mathcal{ALCHIQ}(\mathbf{D})$ and $\mathcal{ALCHIQ}b(\mathbf{D})$

It is obvious that decomposition rule from Subsection 3.4.1 applies in the case of concrete domains as well. Hence, let $\mathcal{BS}^{\mathbf{D},+}_{DL}$ be the $\mathcal{BS}^{\mathbf{D}}_{DL}$ calculus extended with the decomposition rule from Definition 3.4.1. For an $\mathcal{ALCHIQ}(\mathbf{D})$ ($\mathcal{ALCHIQ}b(\mathbf{D})$) knowledge base, saturation of $\Xi(KB)$ by $\mathcal{BS}^{\mathbf{D},+}_{DL}$ decides $\mathbf{D}$-satisfiability of KB. Assuming a bound on the arity of concrete predicates, for unary coding numbers and if $\mathbf{D}$-satisfiability of conjunctions of concrete domain literals can be decided in exponential time, saturation runs in time exponential in $|KB|$.

4.3 Related Work

The need to represent and reason with concrete data was recognized in description logic systems early on. The early systems such as MESON [37] and CLASSIC [20] provided such features, mostly by means of built-in predicates.

The first rigorous treatment of reasoning with concrete data in description logics was given by Baader and Hanschke in [6]. The authors introduce the notion of a concrete domain and consider reasoning in $\mathcal{ALC}(\mathcal{D})$, a logic allowing feature chains (chains of functional roles) to occur in existential and universal restrictions. The authors show that adding a concrete domain does not increase the complexity of checking concept satisfiability, i.e. it remains in PSpace. Sound and complete reasoning can be performed by extending the standard tableaux algorithm for $\mathcal{ALC}$ with an additional branch closure check, which detects potential unsatisfiability of concrete domain constraints on a branch. Furthermore, if transitive closure of roles is allowed, the authors show that checking concept satisfiability becomes undecidable. The approach of Baader and Hanschke was implemented in the TAXON system [1].

Contrary to modern description logic systems, the approach from [6] does not allow TBoxes. In [75] it was shown that allowing cyclic TBoxes makes reasoning with a concrete domain undecidable in general. More importantly, undecidability holds for all numeric concrete domains, which are probably the most practically relevant ones. Still, in [74] it was shown that reasoning with a temporal concrete domain is decidable even with cyclic TBoxes, thus providing for an expressive description logic with features for temporal modeling.

It turned out that even without cyclic TBoxes, extending the logic is difficult. In [75] it was shown that extending $\mathcal{ALC}(\mathcal{D})$ with either acyclic TBoxes, inverse roles or role-forming concrete domain constructor makes reasoning NExpTime-hard. The main reason for this is the presence of feature chains, which may be used to force equivalence of objects at the end of two distinct feature chains. Hence, feature chains

with concrete domains destroy the tree-model property, which was identified in [115] as the main explanation for the good properties of modal and description logics.

Since obtaining expressive logics with concrete domains turned out to be difficult, another path to extending the logic was taken in [53], by prohibiting feature chains. In this way, concrete domain reasoning is restricted only to immediate successors of an abstract individual, so the tree-model property remains preserved. A decision procedure for an expressive $\mathcal{ALCN}\mathcal{H}_{R^+}$ logic extended with concrete domains without features is presented in [53], obtained by extending the tableaux algorithm with concrete domain constraint checking. In [59] the term *datatypes* was introduced for a concrete domain without feature chains, and a tableaux decision procedure for $\mathcal{SHOQ}(\mathbf{D})$ (a logic providing nominals and datatypes) was presented. This approach was extended to allow for n-ary concrete roles in [89]. The results of this research influenced significantly the development of the Semantic Web ontology language OWL [91], which, in its DL version, supports datatypes.

Apart from fundamental issues, such as decidability and complexity or reasoning, other issues were considered to make concrete domains practically applicable. In practice, usually several different datatypes are needed. For example, an application might use the string datatype to represent a person's name and the integer datatype to represent a person's age. It is natural to model each datatype as a separate concrete domain, and then to integrate several concrete domains for reasoning purposes. An approach for integrating concrete domains was already presented in [6], and was further extended by the *datatype groups* approach in [88], which additionally simplifies the integration process by taking care of the extension of the negative concrete literals.

Until now, all existing approaches consider reasoning with a concrete domain in tableaux and automata frameworks. In the resolution setting, many Prolog-like systems, such as XSB[1], provide built-in predicates to handle elementary datatypes. However, to the best of our knowledge, none of these approaches is complete for even the basic logic $\mathcal{ALC}(\mathcal{D})$. Built-in predicates are only capable of handling explicit datatype constants, and cannot cope with individuals introduced by existential quantification. Hence, ours is the first approach that we are aware of which supports reasoning with a concrete domain in the resolution setting and is complete w.r.t. semantics from [6].

Concrete domain resolution calculus can be viewed as an instance of theory resolution [111]. In fact, it is similar to ordered theory resolution [16], in which inferences with theory literals are restricted to maximal literals only. It has been argued in [16] that tautology deletion is not compatible with ordered theory resolution. However, the concrete domain resolution calculus is fully compatible with the standard notion of redundancy, so all existing deletion and simplification techniques can be freely used. Furthermore, in our work we explicitly address the issues related to concrete domains, such as the necessity to consider several "copies" of a clause in a concrete domain resolution inference.

[1] http://xsb.sourceforge.net/

Chapter 5

Reducing Description Logics to Disjunctive Datalog

Based on the decision procedure for $\mathcal{SHIQ}(\mathbf{D})$ from Chapter 3 and Chapter 4, in this chapter we present an algorithm for reducing a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB to a disjunctive datalog program $\mathsf{DD}(KB)$. The program $\mathsf{DD}(KB)$ entails the same set of ground facts as KB, so it can be used for query answering. Furthermore, we also present an algorithm for query answering in $\mathsf{DD}(KB)$ running in worst-case exponential time, and thus show that query answering by reduction to disjunctive datalog is worst-case optimal. As discussed in Section 5.6, although we do not use existing techniques for reasoning in disjunctive datalog, the reduction to disjunctive datalog still makes sense, as it allows applying the magic sets optimization [17, 48].

For the algorithms in this chapter the distinction between the skeleton and the substitution part of a closure is not important. Hence, instead of "closure", we use a more common term "clause". Furthermore, for readability purposes we do not explicitly emphasize $\mathbf{D}$ in notation and terminology related to entailment and satisfiability.

5.1 Overview

Our reduction is based on the observation that all ground functional terms encountered during saturation of $\Xi(KB)$ by $\mathcal{BS}_{DL}^{\mathbf{D},+}$ are of depth at most one. The main idea is to introduce a fresh constant for each ground functional term, and thus to simulate ground inference steps of $\mathcal{BS}_{DL}^{\mathbf{D},+}$ in a function-free version of $\Xi(KB)$. The reduction algorithm proceeds as follows:

- Transitivity axioms are eliminated using the transformation from Section 3.2. Hence, in the rest we focus on $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge bases only.
- The TBox and RBox clauses of $\Xi(KB)$ are saturated together with $\mathsf{gen}(KB)$ by $\mathcal{BS}_{DL}^{\mathbf{D},+}$. As shown by Lemma 5.2.1, certain clauses can be removed from the saturated set, as they may not participate in further inferences.

- If the saturated set does not contain the empty clause, function symbols are eliminated from saturated clauses cf. Section 5.2. Lemma 5.2.3 demonstrates that this transformation does not affect satisfiability. Intuitively, this is so because (*i*) if ABox clauses are added to the saturated set and saturation is continued, all remaining inferences will involve a ground clause and produce a ground clause and (*ii*) each such inference can be simulated in the function-free clause set.
- In order to reduce the size of the datalog program, some irrelevant clauses may be removed, cf. Section 5.3. Lemma 5.3.2 demonstrates that this transformation also does not affect satisfiability.
- Transformation of KB into a disjunctive datalog program, cf. Section 5.4, is now straightforward: it suffices to transform each clause into the equivalent sequent form. This transformation does not affect entailment, which is trivially demonstrated by Theorem 5.4.2.

5.2 Eliminating Function Symbols

For an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base KB, let $\Gamma_{\mathcal{TR}g} = \Xi(KB_{\mathcal{T}} \cup KB_{\mathcal{R}}) \cup \mathsf{gen}(KB)$. Let $\mathsf{Sat}_{\mathsf{R}}(\Gamma_{\mathcal{TR}g})$ denote the *relevant set of saturated clauses*, that is, clauses of type 1, 2, 3, 5, 7 and 9–13 obtained by saturating $\Gamma_{\mathcal{TR}g}$ using $\mathcal{BS}_{DL}^{\mathbf{D},+}$ with eager application of redundancy elimination rules. Finally, let $\Gamma = \mathsf{Sat}_{\mathsf{R}}(\Gamma_{\mathcal{TR}g}) \cup \Xi(KB_{\mathcal{A}})$. Intuitively, $\mathsf{Sat}(\Gamma_{\mathcal{TR}g})$ contains all non-redundant clauses derivable from TBox and RBox by $\mathcal{BS}_{DL}^{\mathbf{D},+}$. Hence, any inference in the saturation of Γ will involve a clause of type 8. Such a clause cannot participate in an inference with a clause of type 4 or 6, so we can safely delete these clauses and consider only the $\mathsf{Sat}_{\mathsf{R}}(\Gamma_{\mathcal{TR}g})$ subset. Adding $\mathsf{gen}(KB)$ is necessary to compute all non-ground consequences of clauses possibly introduced by decomposition.

Lemma 5.2.1. *KB is unsatisfiable if and only if Γ is unsatisfiable.*

Proof. KB is equisatisfiable with $\Xi(KB)$. Let $\Gamma' = \Xi(KB) \cup \mathsf{gen}(KB)$. Since all clauses from $\mathsf{gen}(KB)$ contain a new predicate $Q_{R,f}$, any interpretation of $\Xi(KB)$ can be extended to an interpretation of Γ' by adjusting the interpretation of $Q_{R,f}$ as needed. Hence, $\Xi(KB)$ is equisatisfiable with Γ'. Γ' is unsatisfiable if and only if the set of clauses derived by saturating Γ' with $\mathcal{BS}_{DL}^{\mathbf{D},+}$ contains the empty clause. Since the order in which the inferences are performed can be chosen don't-care non-deterministically, we perform all non-redundant inferences among clauses from $\Gamma_{\mathcal{TR}g}$ first. Let us denote the resulting set of intermediate clauses with $N_i = \mathsf{Sat}(\Gamma_{\mathcal{TR}g}) \cup \Xi(KB_{\mathcal{A}})$.

If N_i contains the empty clause, Γ contains it as well by definition (the empty clause is of type 5), and the claim of the lemma follows. Otherwise, we continue the saturation of N_i. In such a derivation, each $N_j, j > i$, is obtained from N_{j-1} by an inference involving at least one clause not in N_i. By induction on the derivation length, one can easily show that $N_j \setminus N_i$ contains only clauses of type 8: namely, all

non-redundant inferences between clauses of type other than 8 have been performed in N_i, and, by Corollary 4.2.2, each inference involving a clause of type 8 produces a clause of type 8. A clause of type 8 can be decomposed into a ground and a non-ground clause. However, according to Lemma 3.4.8, the non-ground clause of the form $\neg Q_{R,f}(x) \vee R(x, [f(x)])$ is in $\mathsf{gen}(KB)$, so only a clause of type 8 is added to N_j.

Furthermore, by Corollary 4.2.2, a clause of type 4 and 6 can never participate in an inference with a clause of type 8. Hence, such a clause cannot be used to derive a clause in $N_j \setminus N_i$, $j > i$, so N_i can safely be replaced by Γ. Any set of clauses N_j, $j > i$, which can be obtained by saturation from Γ', can be obtained by saturation from Γ as well, modulo clauses of type 4 and 6. Hence, the saturation of Γ' by $\mathcal{BS}_{DL}^{\mathbf{D},+}$ derives the empty clause if and only if the saturation of Γ by $\mathcal{BS}_{DL}^{\mathbf{D},+}$ derives the empty clause, so the claim of the lemma follows. □

If *KB* does not use number restrictions, further optimizations are possible. By Corollary 3.3.7, clauses of types 3 or 5 containing a functional term cannot participate in an inference with a clause of type 8. Hence, such clauses can also be eliminated from the saturated set and need not be included in $\mathsf{Sat}_{\mathsf{R}}(\Gamma_{\mathcal{TR}g})$. Observe that this does not necessarily hold for clauses of type 10 and 11; namely, if there is a clause of type 13 with more than one literal $\neg T_i(x_i, y_i^{\mathsf{c}})$, then such a clause might derive a ground literal containing a functional term.

We now show how to eliminate function symbols from clauses in Γ. Intuitively, the idea is to replace each ground functional term $f(a)$ with a new constant, denoted as a_f. For each function symbol f we introduce a new predicate symbol S_f, containing, for each constant a, a tuple of the form $S_f(a, a_f)$. Thus, S_f contains the f-successor of each constant. A clause C is transformed to replace each occurrence of a term $f(x)$ with a new variable x_f and, for each such x_f, to append the literal $\neg S_f(x, x_f)$ to C. Under this transformation, the Herbrand universe of the clause set becomes finite, and can be represented by a predicate symbol HU whose extension contains all constants a and a_f. The predicate symbol HU is used to bind unsafe variables in clauses. We formalize this process by defining the operator λ as follows:

Definition 5.2.2. *Let KB be an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base. The operator λ is defined on the set of terms and produces a term as follows:*

- $\lambda(a) = a$.
- $\lambda(f(a)) = a_f$, *where a_f is a new globally unique constant*[1].
- $\lambda(x) = x$.
- $\lambda(f(x)) = x_f$, *where x_f is a new globally unique variable.*

We extend λ to $\mathcal{ALCHIQ}(\mathbf{D})$-clauses, such that for an $\mathcal{ALCHIQ}(\mathbf{D})$-clause C, $\lambda(C)$ gives a function-free clause as follows:

[1] Globally unique means that, for some f and a, the constant a_f is always the one and the same.

1. *Each term t in the clause is replaced by $\lambda(t)$.*
2. *For each variable x_f introduced in the first step, the literal $\neg S_f(x, x_f)$ is appended to the clause.*
3. *If after steps 1 and 2 some variable x occurs in a positive literal, but not in a negative literal, the literal $\neg HU(x)$ is appended to the clause.*

For a position p in a clause C, let $\lambda(p)$ denote the corresponding position in $\lambda(C)$. For a substitution σ, let $\lambda(\sigma)$ denote the substitution obtained from σ by replacing each assignment $x \mapsto t$ with $x \mapsto \lambda(t)$. Let λ^- denote the inverse of λ (i.e. $\lambda^-(\lambda(\alpha)) = \alpha$ for any term, clause, position or a substitution α)[2].

Let $\mathsf{FF}(KB) = \mathsf{FF}_\lambda(KB) \cup \mathsf{FF}_{Succ}(KB) \cup \mathsf{FF}_{HU}(KB) \cup \Xi(KB_\mathcal{A})$ denote the function-free version of $\Xi(KB)$, where FF_λ, FF_{Succ} and FF_{HU} are defined as follows, with a and f ranging over all constant and function symbols in $\Xi(KB)$, and C ranging over all clauses in $\mathsf{Sat}_\mathsf{R}(\Gamma_{\mathcal{TR}g})$:

$$\begin{aligned} \mathsf{FF}_\lambda(KB) &= \bigcup \lambda(C) \\ \mathsf{FF}_{Succ}(KB) &= \bigcup S_f(a, \lambda(f(a))) \\ \mathsf{FF}_{HU}(KB) &= \bigcup HU(a) \cup \bigcup HU(\lambda(f(a))) \end{aligned}$$

We now show that KB and $\mathsf{FF}(KB)$ are equisatisfiable.

Lemma 5.2.3. *KB is unsatisfiable if and only if $\mathsf{FF}(KB)$ is unsatisfiable.*

Proof. Since KB and Γ are equisatisfiable by Lemma 5.2.1, the claim of the lemma can be demonstrated by showing that Γ and $\mathsf{FF}(KB)$ are equisatisfiable.

($\Leftarrow$) If $\mathsf{FF}(KB)$ is unsatisfiable, since hyperresolution with superposition, concrete domain resolution and splitting, where all negative literals are selected, is sound and complete [10], a derivation of the empty clause from $\mathsf{FF}(KB)$ exists. We now show that such a derivation can be reduced to a derivation of the empty clause from Γ by sound inference rules, in particular, hyperresolution, paramodulation, instantiation and splitting. In $\mathsf{FF}(KB)$, all clauses are safe, so electrons are always positive ground clauses, and each hyperresolvent is a positive ground clause. Furthermore, since superposition into variables is not necessary for completeness, positive and negative superposition inferences are performed only between ground clauses. Finally, splitting ground clauses simplifies the proof, since all ground clauses on a branch are unit clauses.

Let B be a branch $\mathsf{FF}(KB) = N_0, \ldots, N_n$ of a derivation from $\mathsf{FF}(KB)$ by hyperresolution with superposition and eager splitting, where all negative literals are selected. We show now by induction on n that, for any branch B, there exists a corresponding branch B' in a derivation from Γ by sound inference steps, and a set of clauses N'_m on B' such that: (*) if C is some clause in N_n not of the form $S_f(u, v)$ or $HU(u)$, then

[2] Notice that λ is injective, but not surjective, so the definition of λ^- is correct.

N'_m contains the *counterpart clause* of C, equal to $\lambda^-(C)$. The induction base $n = 0$ is obvious, since $\mathsf{FF}(KB)$ and Γ contain only one branch, on which, other than $S_f(u, v)$ or $HU(u)$, all ground clauses are ABox clauses. Now assume that the proposition (*) holds for some n and consider all possible inferences from premises in N_n deriving a clause C in $N_{n+1} = N_n \cup \{C\}$:

- A superposition into a literal $HU(u)$ is redundant, since HU is instantiated for each constant occurring in $\mathsf{FF}(KB)$, so the conclusion is already on the branch.
- Assume that the inference is a superposition from $s \approx t$ into the ground unit clause L. If L is of the form $S_f(u, v)$, then the proposition obviously holds. Otherwise, clauses $s \approx t$ and L are derived in at most n steps on B, so, by the induction hypothesis, counterpart clauses $\lambda^-(s \approx t)$ and $\lambda^-(L)$ are derivable on B'. Thus, superposition can be performed on these clauses on B' to derive the required counterpart clause.
- Reflexivity resolution can only be applied to a clause $u \not\approx u$ on B. By the induction hypothesis $\lambda^-(u \not\approx u)$ is then derivable on B'. Hence, reflexivity resolution can be applied on B' as well to derive the required counterpart clause.
- Equality factoring is not applicable to a clause on B, since all positive clauses on B are ground unit clauses.
- Consider a hyperresolution inference with a nucleus C, a set of positive ground electrons $E_1, \ldots, E_k$ and a unifier σ, resulting in a hyperresolvent H. Let σ' be the substitution obtained from σ by including a mapping $x \mapsto \lambda^-(x\sigma)$ for each variable $x \in \mathsf{dom}(\sigma)$ not of the form x_f. Let us now perform an instantiation step $C' = (\lambda^-(C))\sigma'$ on B'. Obviously, $\lambda^-(C\sigma)$ and C' may differ only at a position p in C, at which a variable of the form x_f occurs. Let $p' = \lambda^-(p)$. The term in $\lambda^-(C)$ at p' is $f(x)$, so with p'_x we denote the position of the inner x in $f(x)$. In the hyperresolution inference generating H, the variable x_f is instantiated by resolving $\neg S_f(x, x_f)$ with some ground literal $S_f(u, v)$. Hence, $C\sigma$ contains at p the term v, whereas C' contains at p' the term $f(u)$, and $\lambda^-(v) \neq f(u)$. We show that all such discrepancies can be eliminated with sound inferences on B'. Observe that the literal $S_f(u, v)$ is obtained on B from some $S_f(a, a_f)$ by n or less superposition inference steps. Let us with Δ_1 (Δ_2) denote the sequence of ground unit equalities applied to the first (second) argument of $S_f(a, a_f)$. All $s_i \approx t_i$ from Δ_1 or Δ_2 are derivable on B in n steps or less, so corresponding equalities $\lambda^-(s_i \approx t_i)$ are derivable on B' by the induction hypothesis; we denote these sequences with Δ'_1 and Δ'_2. We now perform superposition with equalities from Δ'_1 into C' at p'_x in the reverse order. After this, p'_x will contain the constant a, and p' will contain the term $f(a)$. Hence, we can apply superposition with equalities from Δ'_2 at p' in the original order. After this is done, each position p' will contain the term $\lambda^-(v)$. Let C'' denote the result of removing discrepancies at all positions. Obviously, $C'' = \lambda^-(C\sigma)$. All electrons E_i are

derivable in n steps or less on B, so if E_i is not of the form $S_f(u, v)$ or $HU(u)$, $\lambda^-(E_i)$ is derivable on B'. We hyperresolve these electrons with C'' to obtain H'. Obviously, $H' = \lambda^-(H)$, so the counterpart clause is derivable on B'.

- Since non-ground clauses contain selected literals and all concrete domain literals are positive, concrete domain resolution can be applied only to a set of positive ground clauses C_i. By the induction hypothesis, all $\lambda^-(C_i)$ are derivable on B'. Hence, concrete domain resolution can be applied on B' in the same way as on B, so the counterpart clause is derivable on B'.

- Since all ground clauses on branch B are unit clauses, and no clause in $\mathsf{FF}(KB)$ contains a positive literal with S_f or HU predicates, a ground non-unit clause C generated by an inference on B cannot contain $S_f(u, v)$ and $HU(a)$ literals. Hence, if C is of length k and causes B to be split into k sub-branches, then $\lambda^-(C)$ is of length k and B' can be split into k sub-branches, each of them satisfying (*).

Hence, if there is a derivation of the empty clause on all branches from $\mathsf{FF}(KB)$, then there is a derivation of the empty clause on all branches from Γ as well.

($\Rightarrow$) If Γ is unsatisfiable, since $\mathcal{BS}_{DL}^{\mathbf{D},+}$ is sound and complete, a derivation of the empty clause from Γ exists. We show that such a derivation can be reduced to a derivation of the empty clause in $\mathsf{FF}(KB)$ by sound inference rules.

Let B' be a derivation $\Gamma = N'_0, \ldots, N'_n$ by $\mathcal{BS}_{DL}^{\mathbf{D},+}$. We show by induction on n that there exists a corresponding derivation B of the form $\mathsf{FF}(KB) = N_0, \ldots, N_m$ by sound inference steps, such that: (**) if C' is some clause in N'_n, then N_m contains the counterpart clause $C = \lambda(C')$. The induction base $n = 0$ is trivial. Assume now that (**) holds for some n and consider possible inferences deriving $N'_{n+1} = N'_n \cup \{C'\}$, where the clause C' is derived from premises $P'_i \in N'_n, 1 \leq i \leq k$. By the induction hypothesis, we know that there is a derivation B from $\mathsf{FF}(KB)$ with a clause set N_m containing the counterpart clauses of each P'_i, denoted with $P_i, 1 \leq i \leq k$. Let σ' denote the unifier that the inference is performed with. By Corollary 4.2.2, σ' is ground and contains only assignments of the form $x_i \mapsto a$ or $x_i \mapsto f(a)$. Let $\sigma = \lambda(\sigma')$. Since all P_i are derivable by the induction hypothesis, we can instantiate each P_i into $P_i\sigma$. Obviously, apart from the literals involving S_f and HU, the only difference between $P_i\sigma$ and $\lambda(P'_i\sigma')$ may be that the latter contains a term $f(a)$ at position p, whereas the former contains x_f at $\lambda(p)$. But then $P_i\sigma$ contains a literal $\neg S_f(a, x_f)$, which can be resolved with $S_f(a, a_f)$ to produce a_f at $\lambda(p)$. All such differences can be removed iteratively, and the remaining ground literals involving HU can be resolved away. Hence, each $\lambda(P'_i\sigma')$ is derivable from premises in N_m.

Observe that in all literals of the form $f(a)$, the inner term is marked. Hence, superposition inferences are possible only on the outer position of such terms, which correspond via λ to a_f. Therefore, regardless of the inference type, $C = \lambda(C')$ can be derived from $\lambda(P'_i\sigma')$ by the same inference on the corresponding literals.

The result of a superposition inference in B' may be a clause C' containing a literal $R([a], [f(a)])$, which is decomposed into a clause C'_1 of type 8 and a clause C'_2 of type 3. However, since $\mathsf{gen}(KB) \subseteq \Gamma$, we have $C'_2 \in \Gamma$, so the conclusion C' should only be replaced with the conclusion C'_1. The decomposition inference rule can obviously be applied on B as well to produce a counterpart clause $C_1 = \lambda(C'_1)$. Since $\lambda(C'_2) \in N_m$, this inference is sound by Lemma 3.4.2, so the property (**) holds.

Now it is obvious that, if there is a derivation of the empty clause from Γ, then there is a derivation of the empty clause from $\mathsf{FF}(KB)$ as well. □

The result above means that $KB \models \alpha$ if and only if $\mathsf{FF}(KB) \models \alpha$, where α is of the form $(\neg)A(a)$ or $(\neg)R(a, b)$, for A an atomic concept and R a simple role. The proof also reveals the fact that, in checking satisfiability of $\mathsf{FF}(KB)$, it is not necessary to perform a superposition inference into a literal of the form $HU(a)$.

5.3 Removing Irrelevant Clauses

The saturation of $\Gamma_{\mathcal{TR}g}$ derives new clauses which enable the reduction to $\mathsf{FF}(KB)$. However, the same process introduces many clauses which are not necessary. Consider, for example, the knowledge base $KB = \{A \sqsubseteq C, C \sqsubseteq B\}$. If the precedence of the predicate symbols is $C >_P B >_P A$, the saturation process will derive the clause $\neg A(x) \vee B(x)$, which is not necessary: all ground consequences of this clause can be obtained from clauses $\neg A(x) \vee C(x)$ and $\neg C(x) \vee B(x)$ only. Hence, we present an optimization, by which we reduce the number of clauses in the resulting disjunctive datalog program.

Definition 5.3.1. *For $N \subseteq \mathsf{FF}(KB)$, let $C \in N$ be a clause such that $\lambda^-(C)$ was derived in the saturation of $\Gamma_{\mathcal{TR}g}$ from premises P_i, $1 \leq i \leq k$, by an inference with a substitution σ. Then C is* irrelevant *w.r.t. N if $\lambda^-(C)$ is not derived by the decomposition rule, and, for each premise P_i, $\lambda(P_i) \in N$ and each variable occurring in $\lambda(P_i\sigma)$ occurs in C.* Relevant *is the opposite of irrelevant.*

Let $C_1, C_2, \ldots, C_n$ be a sequence of clauses from $\mathsf{FF}(KB)$ such that the sequence of clauses $\lambda^-(C_n), \ldots, \lambda^-(C_2), \lambda^-(C_1)$ corresponds to the order in which clauses are derived in saturation of $\Gamma_{\mathcal{TR}g}$. Let $\mathsf{FF}(KB) = N_0, N_1, \ldots, N_n$ be a sequence of clause sets such that $N_i = N_{i-1}$ if C_i is relevant w.r.t. N_{i-1}, and $N_i = N_{i-1} \setminus \{C_i\}$ if C_i is irrelevant w.r.t. N_{i-1}, for $1 \leq i \leq n$. Then $\mathsf{FF}_\mathsf{R}(KB) = N_n$ is called the relevant subset *of $\mathsf{FF}(KB)$.*

Removing irrelevant clauses preserves satisfiability, as demonstrated by the following lemma.

Lemma 5.3.2. *$\mathsf{FF}_\mathsf{R}(KB)$ is unsatisfiable if and only if $\mathsf{FF}(KB)$ is unsatisfiable.*

Proof. Let N be a (not necessarily proper) subset of $\mathsf{FF}(KB)$. Furthermore, let $C \in N$ be an irrelevant clause w.r.t. N, where $\lambda^-(C)$ is derived in the saturation of $\Gamma_{\mathcal{TR}g}$ from premises P_i by an inference ξ with a substitution σ, and $\lambda(P_i) \in N$, $i \leq i \leq k$.

We now show the following property (***): N is unsatisfiable if and only if $N \setminus \{C\}$ is unsatisfiable. The ($\Leftarrow$) direction is trivial, since $N \setminus \{C\} \subset N$. For the ($\Rightarrow$) direction, by Herbrand's theorem, N is unsatisfiable if and only if some finite set M of ground instances of N is unsatisfiable. For such M, we construct the set of ground clauses M' in the following way:

- For each $D \in M$ such that D is not a ground instance of C, let $D \in M'$.
- For each $D \in M$ such that D is a ground instance of C with substitution τ, let $\lambda(P_i)\lambda(\sigma)\tau \in M'$, $1 \leq i \leq k$.

Let τ be a ground substitution such that $D = C\tau$. Clauses P_i can be of type 1, 2, 3, 5, 7 or 9–13, so σ can contain only mappings of the form $x \mapsto x'$, $x \mapsto f(x')$ or $y_i \mapsto f(x')$. The sets of variables in $\lambda(P_i\sigma)$ and $\lambda(P_i)\lambda(\sigma)$ obviously coincide. Since C is irrelevant, each variable in C occurs in $\lambda(P_i\sigma)$, so τ instantiates all variables in each $\lambda(P_i)\lambda(\sigma)$. Therefore, each $\lambda(P_i)\lambda(\sigma)\tau$ is a ground instance of a clause $\lambda(P_i)$ in $N \setminus \{C\}$. Furthermore, it is easy to see that $\lambda(P_i)\lambda(\sigma)\tau \subseteq \lambda(P_i\sigma)\tau$. If the inclusion is strict, this is due to literals of the form $\neg S_f(a, b)$ in the latter clause, which do not occur in the first one because σ instantiates some variable from P_i to a functional term $f(x')$ originating from some premise P_j. But then $\lambda(P_j)$ contains the literal $\neg S_f(x', x'_f)$, so $\lambda(P_j)\lambda(\sigma)\tau$ contains $\neg S_f(a, b)$. Therefore, all $\lambda(P_i)\lambda(\sigma)\tau$ can participate in a ground inference corresponding to ξ and derive D. Hence, if M is unsatisfiable, the set M' is unsatisfiable as well. Since M' is a finite unsatisfiable set of ground instances of $N \setminus \{C\}$, $N \setminus \{C\}$ is unsatisfiable by Herbrand's theorem, so the property (***) holds.

Consider the sequence of clause sets $\mathsf{FF}(KB) = N_0, N_1, \ldots, N_n = \mathsf{FF}_{\mathsf{R}}(KB)$ from Definition 5.3.1. For each $N_i = N_{i-1} \setminus \{C_i\}$, $i \geq 1$, the preconditions of property (***) are fulfilled, so by (***), N_i is satisfiable if and only if N_{i-1} is satisfiable. The claim of the lemma now follows by a straightforward induction on i. □

5.4 Reduction to Disjunctive Datalog

Reduction of an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base KB to a disjunctive datalog program is now easy.

Definition 5.4.1. *For an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base KB, the disjunctive datalog program $\mathsf{DD}(KB)$ is obtained by rewriting each clause $A_1 \vee \ldots \vee A_n \vee \neg B_1 \vee \ldots \vee \neg B_m$ from $\mathsf{FF}_{\mathsf{R}}(KB)$ as the rule $A_1 \vee \ldots \vee A_n \leftarrow B_1, \ldots, B_m$.*

Theorem 5.4.2. *Let KB be an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base, defined over a concrete domain $\mathbf{D}$, such that $\mathbf{D}$-satisfiability of finite conjunctions over $\Phi_{\mathbf{D}}$ can be decided in deterministic exponential time. Then the following claims hold:*

1. *KB is unsatisfiable if and only if $\mathsf{DD}(KB)$ is unsatisfiable.*

2. *$KB \models \alpha$ if and only if $\mathsf{DD}(KB) \models_c \alpha$, where α is of the form $A(a)$ or $R(a, b)$ and A is an atomic concept.*

3. *$KB \models C(a)$ for a non-atomic concept C if and only if, for Q a new atomic concept, $\mathsf{DD}(KB \cup \{C \sqsubseteq Q\}) \models_c Q(a)$.*

4. *The number of rules in $\mathsf{DD}(KB)$ is at most exponential, the number of literals in each rule is at most polynomial, and $\mathsf{DD}(KB)$ can be computed in exponential time in $|KB|$, assuming a bound on the arity of the concrete domain predicates and for unary coding of numbers in input.*

Proof. The first claim is an obvious consequence of Lemma 5.3.2. The second claim follows from the first one, since $\mathsf{DD}(KB \cup \{\neg\alpha\}) = \mathsf{DD}(KB) \cup \{\neg\alpha\}$ is unsatisfiable if and only if $\mathsf{DD}(KB) \models_c \alpha$. Furthermore, $KB \models C(a)$ if and only if $KB \cup \{\neg C(a)\}$ is unsatisfiable, which is the case if and only if $KB \cup \{\neg Q(a), \neg Q \sqsubseteq \neg C\} = KB \cup \{\neg Q(a), C \sqsubseteq Q\}$ is unsatisfiable. Now the third claim follows from the second one, and the fact that Q is atomic.

By Lemma 3.3.8, $|\mathsf{Sat}(\Gamma_{\mathcal{TR}g})|$ is at most exponential in $|KB|$, and, for each clause $C \in \mathsf{Sat}(\Gamma_{\mathcal{TR}g})$, the number of literals in C is at most polynomial in $|KB|$. It is easy to see that the application of λ to C can be performed in time polynomial in the number of terms and literals in C. The number of constants a_f added to $\mathsf{DD}(KB)$ is equal to $c \cdot f$, where c is the number of constants, and f the number of function symbols in the signature of $\Xi(KB)$. If numbers are unary coded, both c and f are polynomial in $|KB|$, so the number of constants a_f is also polynomial in $|KB|$. By Theorem 4.2.4, $\mathsf{Sat}(\Gamma_{\mathcal{TR}g})$ can be computed in time at most exponential in $|KB|$, so the fourth claim follows. □

5.5 Answering Queries in $\mathsf{DD}(KB)$

We discuss briefly the techniques for answering queries in $\mathsf{DD}(KB)$. Many techniques have been developed for disjunctive datalog without equality. These techniques can be used, provided that the usual congruence properties of equality are axiomatized correctly. This can be done by adding the following axioms to $\mathsf{DD}(KB)$, where the last axiom is instantiated for each predicate occurring in $\mathsf{DD}(KB)$ other than HU [42]:

$$x \approx x \leftarrow HU(x). \tag{5.1}$$

$$x \approx y \leftarrow y \approx x. \tag{5.2}$$

$$x \approx z \leftarrow x \approx y, y \approx z. \tag{5.3}$$

$$P(\ldots, y, \ldots) \leftarrow P(\ldots, x, \ldots), x \approx y. \tag{5.4}$$

Currently, the state-of-the-art technique for reasoning in disjunctive datalog is so-called intelligent grounding [39]. The algorithm is based on model building, which is performed by generating the ground instantiation of the program rules, generating

candidate models, and using model checking algorithms to eliminate models which do not satisfy the ground rules. In order to avoid generating the entire grounding of the program, carefully designed heuristics are applied to generate the subset of the ground rules which have exactly the same set of the stable models as the original program. Query answering is reduced to model building, since A is not a certain answer if and only if there is a model not containing A.

The intelligent grounding technique is currently the state-of-the-art technique for reasoning in disjunctive datalog and has been implemented successfully in the DLV disjunctive datalog engine [39]. Namely, computing models of a disjunctive program is of interest in many disjunctive datalog applications. For example, disjunctive datalog has been successfully applied to planning problems, where each plan is usually "decoded" from a model.

On the contrary, the models of $\mathsf{DD}(KB)$ are of no interest, as they do not reflect the structure of the models of KB. Hence, we propose query answering in $\mathsf{DD}(KB)$ by hyperresolution and basic superposition, which may be viewed as an extension of the fixpoint computation of plain datalog. A similar technique was presented in [24]. However, the algorithm presented there has two drawbacks: it does not take equality into account, and it does not specify whether application of redundancy elimination techniques is allowed.

Roughly speaking, our technique consists of saturating the rules and facts of $\mathsf{DD}(KB)$ by hyperresolution with basic superposition under an ordering in which all ground query literals are smallest. Additionally, it is required that the query predicate does not occur in the body of any rule. Under these assumptions, one may show that the saturated set of clauses contains all ground query literals cautiously entailed by the program. Because the ordering is total, in each ground disjunction there is exactly one maximal literal. Hence, the semi-naïve bottom-up computation or the join order optimizations can be adapted to the disjunctive case.

It is important to understand that even though we do not propose to reuse intelligent grounding, the reduction to disjunctive datalog has the benefit that it enables the application of the magic sets transformation [48]. This technique is independent of the query answering algorithm and can be reused as-is.

Definition 5.5.1. *For a predicate symbol Q, let $\mathcal{BS}^{\mathbf{D}}_Q$ denote the $\mathcal{BS}^{\mathbf{D}}$ calculus parameterized in the following way:*

- *All ground atoms of the form $Q(\mathbf{a})$ are smallest in the term ordering $\succ$.*
- *All negative literals are selected.*

Furthermore, for any two closures $s \approx t \vee C$ where $s \approx t$ is strictly maximal with respect to C and $s \succ t$, and $Q(\mathbf{a}) \vee D$ where $Q(\mathbf{a})$ is strictly maximal with respect to D, $\mathcal{BS}^{\mathbf{D}}_Q$ performs any possible superposition from t into $Q(\mathbf{a})$, even if the corresponding position in $Q(\mathbf{a})$ is marked.

A remark about the first condition of the above definition is in order. Namely, any admissible ordering is stable under contexts and under substitutions, and is total on ground terms; therefore, it has the subterm property for ground terms [7]. However, such an ordering does not fulfill the first condition from the above definition: for literals $a \approx b$ and $Q(a)$ with $a \succ b$, we always have $Q(a) \succ a$, so $Q(a) \succ a \approx b$.

This situation can be remedied by dropping the requirement on $\succ$ to be stable under substitutions. Hence, for ground terms the term ordering $\succ$ must be total, well-founded and stable under contexts (i.e. for all ground terms s, t and u, and all positions p, $s \succ t$ implies $u[s]_p \succ u[t]_p$). An example is a *query* ordering $\succ_Q$ induced over a total precedence of over constant symbols $>_C$ and predicate symbols $>_P$ such that Q is the smallest element in $>_P$, defined in the following way ($a_{(i)}$ and $b_{(i)}$ are arbitrary constants, P and R are arbitrary predicate symbols, and Q is the query predicate symbol):

- $a \succ_Q b$ if $a >_C b$,
- $P(a_1, \ldots a_n) \succ_Q b$,
- $P(a_1, \ldots a_n) \succ_Q R(b_1, \ldots, b_m)$ if $R >_P R$,
- $P(a_1, \ldots a_n) \succ_Q P(b_1, \ldots, b_n)$ if there is some k, $1 \leq k \leq n$, such that $a_i = b_i$ for $i < k$ and $a_k \succ b_k$,
- $a \succ_Q Q(b_1, \ldots, b_n)$.

It is easy to see that $\succ_Q$ is well-founded, stable under contexts, and that it fulfills the requirements of Definition 5.5.1. However, it is not defined for non-ground terms, since extending $\succ_Q$ to non-ground terms would require $x \succ Q(x)$. Therefore, $\succ_Q$ cannot be used to decide satisfiability of a general first-order theory by $\mathcal{BS}^{\mathbf{D}}$.

However, satisfiability of a positive program P can be decided by saturating P under $\mathcal{BS}^{\mathbf{D}}_Q$. Let P_∞ be the set of closures obtained by saturating P under $\mathcal{BS}^{\mathbf{D}}_Q$ and consider applying model-generation method to P_∞. All non-ground closures in P_∞ have selected negative literals, so they are not productive. Therefore, all productive clauses in P_∞ are positive ground clauses without functional terms. Literals from such clauses can be compared using $\succ_Q$, and a model can be generated in the same way as in [15, 83]. In fact, to saturate P and to generate a model of P_∞ it is not necessary to compare non-ground terms, so stability under substitutions is not needed. Therefore, we conclude that $\mathcal{BS}^{\mathbf{D}}_Q$ is sound and complete for deciding satisfiability of P. From this we obtain the following result:

Lemma 5.5.2. *Let P be a positive satisfiable disjunctive datalog program and Q a predicate not occurring in the body of any rule in P. Then $P \models_c Q(\mathbf{a})$ if and only if $Q(\mathbf{a}) \in N$, where N is the set of closures obtained by saturating P under $\mathcal{BS}^{\mathbf{D}}_Q$ up to redundancy.*

Proof. $P \models_c Q(\mathbf{a})$ if and only if the set of closures N', obtained as the result of saturating $P \cup \{\neg Q(\mathbf{a})\}$ by $\mathcal{BS}_Q^{\mathbf{D}}$ up to redundancy, contains the empty closure. Notice that, since all closures in P are safe, all hyperresolvents are positive ground closures.

Consider first the case when no superposition inference is applied to the literal $\neg Q(\mathbf{a})$ in the saturation of N'. Since P is satisfiable, N' contains the empty closure if and only if a hyperresolution with $\neg Q(\mathbf{a})$ is performed in saturation. Since the literals containing Q are smallest in the ordering, a positive literal $Q(\mathbf{a})$ can be maximal only in a closure $C = Q(\mathbf{a}) \vee D$, where D contains only literals with the Q predicate. Since $\neg Q(\mathbf{a})$ is the only closure where Q occurs negatively, if D is not empty, no literal from D can be eliminated by a subsequent hyperresolution inference. Hence, the empty closure can be derived from such C if and only if D is empty, which is the case if and only if $Q(\mathbf{a}) \in N$.

Assume now that, in the saturation deriving N', several negative superposition inferences from closures $a_i \approx b_i \vee C_i$, $a_i \succ b_i$, are applied to $\neg Q(\mathbf{a})$, resulting in a closure $\neg Q(\mathbf{b}) \vee C$, which then is resolved with a closure $Q(\mathbf{b}) \vee D$, producing $C \vee D$. Such a derivation can be transformed into a derivation where superposition inferences are performed on b_i into $Q(\mathbf{b}) \vee D$, yielding $Q(\mathbf{a}) \vee C \vee D$, which can then participate in a resolution with $\neg Q(\mathbf{a})$ to obtain $C \vee D$. Thus, we may successively eliminate each superposition into some $\neg Q(\mathbf{a})$ and obtain a derivation in which no superposition into $\neg Q(\mathbf{a})$ has been performed. Since in saturating N, all superposition inferences from the smaller side of the equality are performed into all literals containing Q, and all such inferences are sound, $Q(\mathbf{a}) \in N$, so the claim of the lemma follows. □

Assuming that Q is a single predicate, or that it does not occur in the body of any rule in P, does not reduce the generality of the approach, as one can always add a new rule of the form $Q(\mathbf{x}) \leftarrow \mathbf{A}(\mathbf{x})$ to satisfy the conditions of Lemma 5.5.2.

We now consider the complexity of query answering in $\mathsf{DD}(KB)$. Namely, it is well-known that the combined complexity of checking whether $P \models_c A$ for any program P is co-NExpTime$^{\text{NP}}$-complete in $|P|$ [38]. Since $|\mathsf{DD}(KB)|$ is exponential in $|KB|$, a straight-forward reasoning gives an algorithm in co-2NExpTime$^{\text{NP}}$. However, in $\mathsf{DD}(KB)$, each rule is of length polynomial in $|KB|$. Hence, the hardness argument from [38] does not apply directly to $\mathsf{DD}(KB)$, since $\mathsf{DD}(KB)$ is a program of a restricted form.

Theorem 5.5.3. *Let KB be an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base, defined over a concrete domain $\mathbf{D}$, such that $\mathbf{D}$-satisfiability of finite conjunctions over $\Phi_{\mathbf{D}}$ can be decided in deterministic exponential time. Then computing the set of all ground literals of the form $C(a)$ or $R(a, b)$, entailed by $\mathsf{DD}(KB)$, can be done in time exponential in $|KB|$, assuming a bound on the arity of concrete predicates and for unary coding of numbers in input.*

Proof. In a way similar to Lemma 3.3.8, it is easy to see that the maximal length of each ground clause obtained in the saturation of $\mathsf{DD}(KB)$ by $\mathcal{BS}_Q^{\mathbf{D}}$ is polynomial in $|KB|$, so the number of ground clauses is exponential in $|KB|$. Furthermore, in each

application of the hyperresolution inference to some rule r, one selects a ground clause for each body literal of r, which is polynomial in $|KB|$, giving rise to exponentially many different hyperresolution inferences. Hence, the saturation of $\mathsf{DD}(KB)$ may be performed in time exponential in $|KB|$. Since by Lemma 5.5.2 saturation of $\mathsf{DD}(KB)$ computes all certain answers of $\mathsf{DD}(KB)$, the claim of the theorem follows. □

We finish with a note about an optimization which may be applied if unique names assumption is assumed in KB. By assuming an ordering where all individual constants a are smaller in the ordering than all constants b_f, then superposition inferences from the smaller side of an equation are not needed. Namely, since $b_f \succ a$, no superposition inference may replace some a with b_f in a saturation of $\mathsf{DD}(KB) \cup \{\neg Q(\mathbf{a})\}$. Furthermore, for any clause $D = a \approx b \vee C$, there is a clause $a \not\approx b$, so D may be replaced immediately with C. Hence, no equality can reduce a position in $\neg Q(\mathbf{a})$, so we always have the first case from the proof of Lemma 5.5.2.

5.6 Discussion

It is important to note that our reduction to disjunctive datalog preserves entailment under *descriptive semantics*. Namely, in [82] Nebel has shown that knowledge bases containing terminological cycles are not definitorial. This means that, for some fixed partial interpretation of atomic concepts, several interpretations of non-atomic concepts may exist. In such a case, it might be reasonable to designate a particular interpretation as the intended one, with least and greatest fixpoint models being the obvious candidates. However, in [82] it is argued that it is not clear which interpretation best matches the intuition, as choosing either of the fixpoint models has its drawbacks. Consequently, most description logic systems implement the so-called descriptive semantics, which coincides with that of Definition 2.8.5.

Obviously, our decision procedure implements exactly the descriptive semantics. Furthermore, Theorem 5.4.2 shows that $\mathsf{DD}(KB)$ entails exactly those ground facts which are entailed by our decision procedure, so $\mathsf{DD}(KB)$ also implements the descriptive semantics. Hence, one may say that the set of facts contained in any minimal model of $\mathsf{DD}(KB)$ coincides with the set of facts entailed by KB under descriptive semantics. Intuitively, the saturation process is responsible for this fact.

We point out that basic superposition is crucial for the correctness of the reduction. Namely, the basicness restriction renders superposition into Skolem function symbols redundant, which allows treating ground functional terms as constants.

Finally, we note that, although all lemmas and theorems consider $\mathcal{ALCHIQ}(\mathbf{D})$, it is easy to see that they equally apply to the case of $\mathcal{ALCHIQ}b(\mathbf{D})$ as well.

5.7 Related Work

Our work was largely motivated by [49], where a decidable intersection of description logic and logic programming was investigated. In particular, the authors concentrate

on description logic constructs that can be encoded and executed using existing rule engines. Thus, the description logic component allows only existential quantifiers to occur under negative, and universal quantifiers to occur under positive polarity. The authors present an operator for translating a description logic knowledge base into a logic program. This approach was later extended in [116] to support more expressive description logic, provided that the rule engine supports advanced features. For example, if the rule engine supports equality, then functionality restrictions can be expressed in the knowledge base and executed by the translation. However, our approach is a significant extension, since we handle full $\mathcal{SHIQ}(\mathbf{D})$.

In [54] it was show how to convert $\mathcal{SHIQ}^*$ knowledge bases into so-called *conceptual logic programs* (CLP). The CLPs generalize the good properties of description logic to the answer set programming setting. Apart from the usual constructs, $\mathcal{SHIQ}^*$ supports the transitive closure of roles. Although the presented transformation preserves the semantics of the knowledge base, the obtained answer set program is not safe. Hence, its grounding is infinite and therefore cannot be evaluated using existing answer set solvers. The problem of decidable reasoning for CLPs is addressed by an automata-based technique. On the contrary, our transformation produces a safe program with a finite grounding. Hence, issues related to decidability of reasoning are is handled by the transformation, and not by the query answering algorithm: the disjunctive program obtained by our approach can be evaluated using any technique for reasoning in disjunctive datalog programs.

Another approach for reducing description logic knowledge bases to answer set programming was presented in [3]. To deal with existential quantification, this approach uses function symbols. Thus, Herbrand's universe of the programs obtained by the reduction is infinite, so existing answer set solvers cannot be used for decidable reasoning. In fact, decidability of reasoning is not considered at all.

The query answering algorithm from Section 5.5 was inspired by [24], where the authors suggested that the fixpoint computation of disjunctive semantics can be optimized by introducing an appropriate literal ordering. Our approach extends the one from [24] by showing that simplification techniques can be applied in the fixpoint computation, and by extending the calculus to handle equality.

Chapter 6

Data Complexity of Reasoning

Algorithms from chapters 3, 4 and 5 run in worst-case exponential time in $|KB|$, assuming unary coding of numbers, a bound on the arity of concrete predicates and an exponential oracle for reasoning with a concrete domain. In [114] $\mathcal{SHIQ}$ was shown to be ExpTime-complete for any coding of numbers, so our algorithms are (almost) worst-case optimal. We stress that our complexity results, as well as the results from [114], actually address the *combined complexity* of reasoning, which is measured in the size of the knowledge base $|KB|$, including the sizes of RBox, TBox and ABox.

ExpTime-completeness is a rather discouraging result, since $|KB|$ can be large in practice. However, by drawing a parallel with traditional database applications, $|KB|$ often depends mainly on the size of the ABox, which has a role of a database instance; on the contrary, the size of the TBox, having the role of a database schema, is usually limited. For such applications, *data complexity*, where $|KB_{\mathcal{R}} \cup KB_{\mathcal{T}}|$ is assumed to be bound by a constant and complexity is measured in $|KB_{\mathcal{A}}|$ (the size of the ABox of KB) only, provides a much better performance estimate. In practice, data complexity provides a much better performance estimate when the size of the assertional knowledge is much bigger than the size of the terminological knowledge, i.e. where $|KB_{\mathcal{R}}| + |KB_{\mathcal{T}}| = |KB_{\mathcal{TR}}| \ll |KB_{\mathcal{A}}|$.

To fully separate the terminological from assertional knowledge, in this chapter we assume that KB is *extensionally reduced*, i.e. that ABox axioms contain only (negation of) atomic concepts. Namely, complex concept axioms in ABox assertions actually specify terminological knowledge. For extensionally reduced knowledge bases, $|KB_{\mathcal{A}}|$ is the measure of "raw" input data processed by the algorithm.

Until now, data complexity of testing knowledge base satisfiability even for the basic logic $\mathcal{ALC}$ with general TBoxes was unknown. Based on results from Chapter 5, in Section 6.1 we show that satisfiability checking is NP-complete in the size of ABox for any logic between $\mathcal{ALC}$ and $\mathcal{SHIQ}(\mathbf{D})$ (regardless of how numbers are encoded), and that instance and role checking — the basic query answering problems for description logics — are co-NP-complete.

In Section 6.2 we define Horn-$\mathcal{SHIQ}(\mathbf{D})$, a logic related to $\mathcal{SHIQ}(\mathbf{D})$ analogously as Horn logic is related to full first-order logic. Namely, Horn-$\mathcal{SHIQ}(\mathbf{D})$ provides ex-

istential and universal quantifiers, but does not provide for disjunctive reasoning. This restriction allows us to show that basic reasoning problems for Horn-$\mathcal{SHIQ}(\mathbf{D})$ are P-complete in $|KB_{\mathcal{A}}|$. Hence, along the traditional lines of knowledge representation research, the capability of representing disjunctive information is traded for polynomial data complexity. Since disjunctive reasoning is not needed for many applications, Horn-$\mathcal{SHIQ}(\mathbf{D})$ is a very appealing logic because it is still very expressive, but gives theoretical reasons for hope that it can be implemented efficiently in practice.

To develop an intuition and to provide a more detailed account behind these interesting results, in Section 6.3 we compare our results with the similar known results for datalog and its variants [31].

6.1 Data Complexity of Satisfiability

Our results from Chapter 5 show almost immediately that checking satisfiability of a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB is NP-complete in the size of data.

Lemma 6.1.1 (Membership)**.** *For an extensionally reduced $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB, if $\mathbf{D}$-satisfiability of finite conjunctions over $\Phi_{\mathbf{D}}$ can be decided in polynomial time, data complexity of checking $\mathbf{D}$-satisfiability of KB is in* NP*, assuming a bound on the arity of concrete predicates.*

Proof. Let c be the number of constants, f the number of function symbols in the signature of $\Xi(KB)$, and s the number of facts in $\Xi(KB)$. By Definition 5.2.2, the number of constants in $\mathsf{DD}(KB)$ is bounded by $\ell_1 = c + cf$ (cf accounts for constants of the form a_f), and the number of facts in $\mathsf{DD}(KB)$ is bounded by $\ell_2 = s + c + 2cf$ (c accounts for facts of the form $HU(a)$, one cf accounts for the facts of the form $S_f(a, a_f)$, and the other cf accounts for the facts of the form $HU(a_f)$). Since we assume that $|KB_{\mathcal{TR}}|$ is bounded by a constant, f is bounded by a constant (regardless of how numbers are encoded), so both ℓ_1 and ℓ_2 are polynomial in $|KB_{\mathcal{A}}|$.

Hence, $|\mathsf{DD}(KB)|$ is exponential in $|KB|$ only because the number of rules in $\mathsf{DD}(KB)$ is bounded by the number of closures obtained by saturating the set of closures $\Gamma_{\mathcal{TR}g} = \Xi(KB_{\mathcal{T}} \cup KB_{\mathcal{R}}) \cup \mathsf{gen}(KB)$ by $\mathcal{BS}_{DL}^{\mathbf{D},+}$. Since $\Gamma_{\mathcal{TR}g}$ does not contain ABox closures, by Lemma 3.3.8 the number of closures in $\mathsf{Sat}(\Gamma_{\mathcal{TR}g})$ is exponential in $|KB_{\mathcal{TR}}|$. Since we assume that the latter is bounded by a constant, both the number of rules in $\mathsf{DD}(KB)$ and their length are bounded by constants, so $|\mathsf{DD}(KB)|$ is polynomial in $|KB_{\mathcal{A}}|$, and can be computed from KB in polynomial time. Now the claim of the lemma follows from a well-known fact that checking satisfiability of a disjunctive datalog program is NP-complete [31]. A minor difference is that $\mathsf{DD}(KB)$ may contain concrete predicates, so for the sake of completeness we prove this result as well.

Assuming $\mathsf{DD}(KB)$ contains r rules and at most v variables in a rule, the number of literals in a ground instantiation $\mathsf{ground}(\mathsf{DD}(KB))$ is bounded by $r \cdot \ell_1^v + \ell_2$ (in each rule, each variable can be replaced in ℓ_1 possible ways). Assuming r and v are bounded by constants, $|\mathsf{ground}(\mathsf{DD}(KB))|$ is polynomial in $|KB_{\mathcal{A}}|$. Satisfiability of $\mathsf{ground}(\mathsf{DD}(KB))$ can be checked by non-deterministically generating an interpretation

I and then checking whether it is a $\mathbf{D}$-model. Checking whether I is a model can be performed in polynomial time. To additionally check whether I is a $\mathbf{D}$-model, it is sufficient to check whether $\widehat{I}$ is $\mathbf{D}$-satisfiable, which can be done in polynomial time by assumption. Since $\mathsf{DD}(KB)$ and KB are equisatisfiabile by Theorem 5.4.2, the claim of the lemma follows. □

Lemma 6.1.2 (Hardness)**.** *Checking satisfiability of an $\mathcal{ALC}$ knowledge base KB is* NP*-hard in $|KB_{\mathcal{A}}|$.*

Proof. The proof is by the reduction from a well-known Graph 3-Coloring problem: a graph G is 3-colorable if and only if it is possible to assign a singe color from the set $\{Red, Green, Blue\}$ to each of the graph's vertices such that no two adjoining vertices have the same color. Deciding whether G is 3-colorable is known to be NP-complete in the number of edges [90].

Assuming that G is given as a symmetric relation *edge*, we construct the knowledge base KB_G whose ABox contains only the edge assertions, and whose TBox contains the following axioms:

$$\top \sqsubseteq Red \sqcup Green \sqcup Blue \tag{6.1}$$
$$Red \sqcap Green \sqsubseteq \bot \tag{6.2}$$
$$Green \sqcap Blue \sqsubseteq \bot \tag{6.3}$$
$$Red \sqcap Blue \sqsubseteq \bot \tag{6.4}$$
$$Red \sqsubseteq \forall edge.\neg Red \tag{6.5}$$
$$Green \sqsubseteq \forall edge.\neg Green \tag{6.6}$$
$$Blue \sqsubseteq \forall edge.\neg Blue \tag{6.7}$$

Obviously, G is 3-colorable if and only if KB_G is satisfiable. Namely, (6.1) specifies that each individual (i.e. vertex) in the model must be assigned at least one color, (6.2) – (6.4) specify that colors are pair-wise disjoint so each vertex is assigned at most one color, and (6.5) – (6.7) specify the conditions of 3-colorability. Hence, each model of KB_G gives an assignment of colors to vertices of G; conversely, each assignment of colors of G specifies a model of KB_G. Since the size of the TBox of KB_G is constant, and the size of ABox of KB_G is double the number of edges of G, the claim of the lemma follows. □

We now state the main result of this section.

Theorem 6.1.3. *Let KB be an extensionally reduced knowledge base expressed in any logic between $\mathcal{ALC}$ and $\mathcal{SHIQ}(\mathbf{D})$. Assuming a polynomial oracle for reasoning with concrete domains and a bound on the arity of concrete domain predicates, data complexity of certain reasoning problems is as follows:*

- *checking KB satisfiability is* NP*-complete,*

- *checking KB unsatisfiability is co-*NP*-complete, and*
- *checking whether* $KB \models \alpha$*, for* α *of the form* $(\neg)C(a)$ *with* $|C|$ *bounded or of the form* $(\neg)R(a,b)$*, is also co-*NP*-complete.*

Proof. The first claim is a direct consequence of lemmata 6.1.1 and 6.1.2. Checking unsatisfiability is a complementary problem to checking satisfiability, so the second claim follows from the first one. Furthermore, both instance checking (assuming $|C|$ is bounded) and role checking can be reduced to unsatisfiability in constant time, so the third claim follows from the second one. □

We finish this section with a remark that, if a knowledge base is extensionally reduced, then ABox assertions do not contain number restrictions, so $|KB_{\mathcal{A}}|$ and data complexity do not depend on the used number coding.

6.2 A Horn Fragment of $\mathcal{SHIQ}(\mathbf{D})$

Horn logic is a well-known fragment of first-order logic where formulae are restricted to clauses containing at most one positive literal. Hence, each Horn clause can be understood as a rule where several body literals imply one head literal. The main limitation of Horn logic is the inability to represent disjunctive information; however, the main benefit of Horn logic is the existence of refutation procedures, such as SLD-resolution, which can be efficiently implemented in practice [94]. Furthermore, if function symbols are not used, data complexity of query answering in Horn logic is P-complete [31], thus making function-free Horn logic suitable for practical usage.

Following this idea, in this section we identify the Horn fragment of $\mathcal{SHIQ}(\mathbf{D})$ enjoying similar properties. In Horn-$\mathcal{SHIQ}(\mathbf{D})$ the capability of representing disjunctive information is traded for P-complete data complexity of reasoning. This fragment is not defined by enumerating DL constructors; rather, it is defined by the way constructors are used, as specified by the following test:

Definition 6.2.1. *Let* pl^+ *and* pl^- *be two mutually recursive functions assigning a non-negative integer to a* $\mathcal{SHIQ}(\mathbf{D})$ *concept* D*, defined inductively as in Table 6.1, where* $\lceil 0 \rceil = 0$ *and* $\lceil n \rceil = 1$ *for* $n > 0$*. For a concept* D *and a position* p *of a subconcept in* D*, let* $\mathsf{pl}(D,p)$ *be defined as follows:*

$$\mathsf{pl}(D,p) = \begin{cases} \mathsf{pl}^+(D|_p) & \text{if } \mathsf{pol}(D,p) = 1 \\ \mathsf{pl}^-(D|_p) & \text{if } \mathsf{pol}(D,p) = -1 \end{cases}$$

A $\mathcal{SHIQ}(\mathbf{D})$ *concept* C *is a* Horn *concept if* $\mathsf{pl}^+(C,p) \leq 1$ *for each position* p *of a subconcept in* C *(including the empty position* ϵ*). An extensionally reduced* $\mathcal{ALCHIQ}(\mathbf{D})$ *knowledge base* KB *is* Horn *if for each axiom* $C \sqsubseteq D \in KB_{\mathcal{A}}$*, the concept* $\neg C \sqcup D$ *is Horn. An extensionally reduced* $\mathcal{SHIQ}(\mathbf{D})$ *knowledge base* KB *is* Horn *if* $\Omega(KB)$ *is Horn.*

Table 6.1: Definitions of pl^+ and pl^-

D	$\mathsf{pl}^+(D)$	$\mathsf{pl}^-(D)$
$\top$	0	0
$\bot$	0	0
A	1	0
$\neg C$	$\mathsf{pl}^-(C)$	$\mathsf{pl}^+(C)$
$C_1 \sqcap \ldots \sqcap C_n$	$\max_{1\leq i\leq n}\lceil \mathsf{pl}^+(C_i)\rceil$	$\sum_{1\leq i\leq n} \lceil \mathsf{pl}^-(C_i)\rceil$
$C_1 \sqcup \ldots \sqcup C_n$	$\sum_{1\leq i\leq n} \lceil \mathsf{pl}^+(C_i)\rceil$	$\max_{1\leq i\leq n} \mathsf{pl}^-(C_i)$
$\exists R.C$	1	$\lceil \mathsf{pl}^-(C)\rceil$
$\forall R.C$	$\lceil \mathsf{pl}^+(C)\rceil$	1
$\geq n\, R.C$	1	$\frac{(n-1)n}{2} + n \cdot \lceil \mathsf{pl}^+(C)\rceil$
$\leq n\, R.C$	$\frac{n(n+1)}{2} + (n+1) \cdot \lceil \mathsf{pl}^-(C)\rceil$	1
$\exists T_1, \ldots, T_m.d$	1	1
$\forall T_1, \ldots, T_m.d$	1	1
$\geq n\, T$	1	$\frac{(n-1)n}{2}$
$\leq n\, T$	$\frac{n(n+1)}{2}$	1

As we show in the example below, the structural transformation from Subsection 3.3.1 should be fine-tuned in order to preserve Horness. We first define the extended version of the transformation, and then discuss the intuition behind this definition.

Definition 6.2.2. *A* Horn-compatible *structural transformation is identical to the one from Definition 3.3.1 with the following difference in the definition of* $\mathsf{Def}(C)$*, where* $\alpha = Q$ *if* $\mathsf{pl}(C,p) > 0$*, and* $\alpha = \neg Q$ *if* $\mathsf{pl}(C,p) = 0$*,* Q *is a new atomic concept, and* $\neg(\neg Q) = Q$*:*

$$\mathsf{Def}(C) = \begin{cases} \{C\} & \text{if } \Lambda(C) = \emptyset \\ \{\neg\alpha \sqcup C|_p\} \cup \mathsf{Def}(C[\alpha]_p) & \text{if } p \in \Lambda(C) \text{ and } \mathsf{pol}(C,p) = 1 \\ \{\neg\alpha \sqcup \neg C|_p\} \cup \mathsf{Def}(C[\neg\alpha]_p) & \text{if } p \in \Lambda(C) \text{ and } \mathsf{pol}(C,p) = -1 \end{cases}$$

Unless otherwise explicitly mentioned, in the rest of this section we assume that $\Xi(KB)$ is computed using the Horn-compatible structural transformation. It is obvious that Lemma 3.3.2 holds in this case as well, i.e. that KB and $\Xi(KB)$ are equisatisfiable. The only difference to Definition 3.3.1 is that a concept at a position p in C can be replaced with $\neg Q$ instead of with Q, depending on $\mathsf{pol}(C,p)$ and $\mathsf{pl}(C,p)$. This change does not affect the correctness of the transformation.

We now explain the intuition behind definitions 6.2.1 and 6.2.2. By Table 2.1, one can see that $\mathsf{pl}^+(C)$ yields the maximal number of positive literals in closures obtained by clausifying $\forall x : \pi_y(C,x)$ for a concept C. In counting the literals, one has to take care in handling subconcepts. For example, let $C = \forall R.D_1 \sqcup \forall R.\neg D_2$; in clausification of C, subconcepts $\forall R.D_1$ and $\forall R.\neg D_2$ are replaced with new names α_1

and α_2. The above definitions ensure that $\alpha_1 = \neg Q_1$ and $\alpha_2 = Q_2$. Namely, clausifying $\forall R.\neg D_2$ does not yield positive literals; therefore, one can freely let $\neg\alpha_1$ be a positive atomic concept. On the contrary, translating $\forall R.D_1$ does yield one positive literal, so one should ensure that $\neg\alpha_2$ is a negative atomic concept; otherwise, the structural transformation would introduce a non-Horn concept $\neg\alpha \sqcup C|_p$. To summarize, $C|_p$ is replaced with α being a positive atomic concept if clausification of $C|_p$ produces a positive literal, and a negative atomic concept otherwise.

The function $\lceil\ \rceil$ takes into account that $C|_p$ will be replaced with only one positive atomic concept even if clausification of $C|_p$ produces more than one positive literal. For example, clausifying $D_1 \sqcup D_2$ in the concept $C = \forall R.(D_1 \sqcup D_2)$ produces two positive literals; however, $D_1 \sqcup D_2$ will be replaced in C by structural transformation with only one positive literal.

One also needs to distinguish between subconcepts occurring under positive and negative polarity: in $\neg(\neg A \sqcap \neg B)$ the concepts $\neg A$ and $\neg B$ are effectively positive, and $\sqcap$ is effectively $\sqcup$. Hence, $\mathsf{pl}^+(C|_p)$ ($\mathsf{pl}^-(C|_p)$) counts the number of positive literals used to clausify $C|_p$, provided that $C|_p$ occurs in C under positive (negative) polarity.

Now a concept C is Horn if closures obtained by clausifying C (and indirectly each subconcept of C) contain at most one positive literal. For Horn knowledge bases, special care must be taken in handling transitivity axioms. Namely, although transitivity axioms are translated by π into Horn closures in the reduction to disjunctive datalog they are encoded by axioms of the form $\forall R.C \sqsubseteq \forall S.(\forall S.C)$. Now $\mathsf{pl}^+(\exists R.\neg C \sqcup \forall S.(\forall S.C)) = 1 + \mathsf{pl}^+(C)$, so if $\mathsf{pl}^+(C) > 0$, $\Omega(KB)$ is not a Horn knowledge base. Definition 6.2.1 simply takes the effect of Ω into account.

Lemma 6.2.3. *For a Horn-$\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB, each closure from $\Xi(KB)$ contains at most one positive literal.*

Proof. We first show the following property (*): for a Horn concept C, all concept in $\mathsf{Def}(C)$ are Horn concepts. The proof is by induction on the application of the operator Def. The induction base for $\Lambda(C) = \emptyset$ is obvious. Consider an application of $\mathsf{Def}(C)$, where C is a Horn concept and p a position of a subconcept of C, such that $C|_p$ is neither an atomic concept, nor a negation of an atomic concept, and for each position q below p, $C|_q$ is either an atomic concept or a negation of an atomic concept. Observe that in all cases, we have $\mathsf{pl}^+(\alpha) = \mathsf{pl}(C,p)$ and $\mathsf{pl}^+(\neg\alpha) = 1 - \mathsf{pl}(C,p)$. We now consider two cases, depending on $\mathsf{pol}(C,p)$:

- Consider $\mathsf{pol}(C,p) = 1$. We have $\mathsf{pl}^+(\neg\alpha \sqcup C|_p) = \mathsf{pl}^+(\neg\alpha) + \mathsf{pl}^+(C|_p) = \mathsf{pl}^+(\neg\alpha) + \mathsf{pl}(C,p) = 1$. Furthermore, since $\mathsf{pl}(C,p) = \mathsf{pl}(C[\alpha]_p, p)$, $C[\alpha]_p$ is a Horn concept.
- Consider $\mathsf{pol}(C,p) = -1$. We have $\mathsf{pl}^+(\neg\alpha \sqcup \neg C|_p) = \mathsf{pl}^+(\neg\alpha) + \mathsf{pl}^-(C|_p) = \mathsf{pl}^+(\neg\alpha) + \mathsf{pl}(C,p) = 1$. Furthermore, since $\mathsf{pl}(C,p) = \mathsf{pl}(C[\neg\alpha]_p, p)$, $C[\neg\alpha]_p$ is a Horn concept.

Hence, each application of the operator Def decomposes a Horn concept into C two simpler Horn concepts, so (*) holds. Furthermore, for each $C|_p$ or $\neg C|_p$ in the definition

of Def, each immediate subconcept is either an atomic concept, or a negation of an atomic concept.

For $D \in \mathsf{Def}(C)$, by definition of π it is easy to see that $\mathsf{pl}^+(D)$ gives the maximal number of positive literals occurring in a closure from $\mathsf{Cls}(\forall x : \pi_y(D, x))$. Thus, if C is a Horn concept, all closures from $\mathsf{Cls}(\mathsf{Def}(C))$ have at most one positive literal. All closures obtained by translating RBox and ABox axioms of $\Omega(KB)$ obviously contain at most one positive literal, so the claim of the lemma follows. □

We now show that $\mathcal{BS}_{DL}^{\mathbf{D},+}$, when applied to Horn premises, produces only Horn conclusions.

Lemma 6.2.4. *If all premises of an inference by $\mathcal{BS}_{DL}^{\mathbf{D},+}$ contain at most one positive literal, the inference conclusions contain at most one positive literal as well.*

Proof. In ordered hyperresolution and positive or negative superposition, each side premise participates in an inference on the positive literal, which does not occur in the conclusion. Hence, the number of the positive literals in the conclusion is equal to the number of positive literals in the main premise. Furthermore, reflexivity resolution only reduces the number of negative literals in a closure, and equality factoring is never applicable to a closure with only one positive literal. All side premises of a concrete domain resolution inference participate in the inference on the positive literals, so the conclusion contains only negative literals. Finally, a closure participating in a decomposition inference must contain the single positive literal $R(t, f(t))$, so both resulting closures have exactly one positive literal. □

The following corollary is a direct consequence of lemmas 6.2.3 and 6.2.4:

Corollary 6.2.5. *If KB is a Horn $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base, then $\mathsf{DD}(KB)$ is a Horn datalog program.*

We are now ready to show the main result of this section.

Lemma 6.2.6 (Membership)**.** *For an extensionally reduced Horn-$\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB, if $\mathbf{D}$-satisfiability of finite conjunctions over $\Phi_{\mathbf{D}}$ can be decided in polynomial time, data complexity of checking $\mathbf{D}$-satisfiability of KB is in* P*, assuming a bound on the arity of concrete predicates.*

Proof. Since KB is satisfiable if and only if $\mathsf{DD}(KB)$ is satisfiable, data complexity of checking satisfiability of KB is bounded by the data complexity of checking satisfiability of a Horn program $\mathsf{DD}(KB)$. It is well-known that the latter can be performed e.g. using bottom-up fixpoint saturation in polynomial time [31]. In our case $\mathsf{DD}(KB)$ can contain constraints and concrete predicates, which may be used to derive the empty closure. However, concrete domain resolution may be applied once, after fixpoint has been computed. Hence, saturation of $\mathsf{DD}(KB)$ can be performed in time polynomial in the number of facts. As in Lemma 6.1.2, $|\mathsf{DD}(KB)|$ is polynomial in $|KB_{\mathcal{A}}|$, so the claim of this lemma follows. □

Lemma 6.2.7 (Hardness)**.** *Checking satisfiability of a Horn $\mathcal{ALC}$ knowledge base KB is* P*-hard in $|KB_{\mathcal{A}}|$.*

Proof. The proof is by the reduction from the well-known Reachability problem, i.e. deciding whether there is a path from a node a_1 to a node a_n in a graph G. This problem is known to be P-complete [90] in the number of edges of G.

Assuming that G is given as a symmetric relation *edge*, we construct the knowledge base KB_G whose ABox contains only the edge assertions and the axioms $C(a_1)$ and $\neg C(a_n)$, and whose TBox contains only the following axiom:

$$C \sqsubseteq \forall edge.C \tag{6.8}$$

Obviously, there is a path from a_1 to a_n in G if and only if KB_G is unsatisfiable. Namely, if there is such a path in G, the axiom 6.8 makes sure that for all individuals a_i on a path, $a_i^I \in C^I$ for any interpretation I, thus forcing a contradiction on a_n. Conversely, if there is not such a path, an model I of KB_G can be generated from G by setting $edge^I = edge$, and $a_i^I \in C^I$ if and only if a_i is reachable from a_1. Since the size of the TBox of KB_G is constant, the size of ABox of KB_G is double the number of edges of G plus two, and KB_G is obviously a Horn knowledge base, unsatisfiability is P-complete. Since satisfiability is a complementary problem to unsatisfiability, it is also P-complete. □

The following theorem is an immediate consequence of these two lemmata.

Theorem 6.2.8. *Let KB be an extensionally reduced Horn knowledge base expressed in any logic between $\mathcal{ALC}$ and $\mathcal{SHIQ}(\mathbf{D})$. Assuming a polynomial oracle for reasoning with concrete domains and a bound on the arity of concrete domain predicates, then the problems of checking KB (un)satisfiability, and checking whether $KB \models \alpha$, for α of the form $(\neg)C(a)$ with $|C|$ bounded or of the form $(\neg)R(a,b)$, are* P*-complete in the size of data.*

6.3 Discussion

To better understand the results from the previous two sections, we contrast them with well-known results for (disjunctive) datalog [31]. Since datalog has been successfully applied in practice, this analysis gives interesting insights into practical applicability of DLs.

Interestingly, the data complexities of datalog variants and of corresponding $\mathcal{SHIQ}$ fragments coincide. Namely, without disjunctions a $\mathcal{SHIQ}$ knowledge base and a datalog program always have at most one model, which can be computed in polynomial time. With disjunctions, several models are possible, and this must be dealt with using reasoning-by-cases. Intuitively, one needs to "guess" a model, and this increases data complexity to NP.

The key difference between datalog and DLs is reflected in the *parametric complexity*. For a datalog program P and a ground atom α, checking whether $P \models \alpha$ can

be performed in time $\mathcal{O}(|P|^v)$, where v is the maximal number of distinct variables in a rule of P. Namely, the problem can be solved by grounding P, i.e. replacing in each rule of P all variables with individuals from P in all possible ways. The size of the grounding is obviously bounded by $|P|^v$, and propositional logic programming is P-complete, which gives the above estimate. In general, v is linear in $|P|$, so the size of the grounding is exponential; thus, the combined complexity of datalog coincides with the combined complexity of $\mathcal{SHIQ}$. However, in practical applications v is usually small, so it makes sense to assume a bound on v. Under this assumption, datalog actually exhibits polynomial behavior.

An intuitive analogous limitation for DLs might be to restrict the size of concepts in axioms. However, this is not an adequate limitation, since axioms with complex concepts can be polynomially reduced to axioms with just elementary DL concepts by structural transformation. Namely, DLs are closely related to two-variable fragment of first-order logic: $\mathcal{ALC}$ concepts correspond to first-order formulae with only two variables regardless of nesting. By limiting the numbers occurring in number restrictions, for $\mathcal{SHIQ}$ the number of variables in axioms is "intrinsically" bounded.

We summarize the difference between datalog and DLs as follows: assuming a bound on the axiom length, but not on the number of axioms, reasoning in datalog is (non-deterministically) polynomial, but in DLs it is exponential. The reason for this is that DLs provide the existential quantifier, which can succinctly encode models with paths of exponential length. The saturation step eliminates the function symbols, but it also incurs an exponential blowup in the program size to account for such paths. Hence, although combined complexity of both datalog and DLs is exponential, the reasons for exponential behavior are different.

In [5, Chapter 5] two sources of complexity in DLs have been identified: OR-branching caused by the existence of numerous possible models, and AND-branching caused by the existence of paths within a model. Our results show that OR-branching is not so "bad" as AND-branching: the former incurs "only" an increase to NP, whereas the latter incurs an increase in complexity to ExpTime.

6.4 Related Work

Data complexity of reasoning in description logics was not considered by many researchers so far. We are only aware of [102], where the complexity bounds of instance checking for certain description logics were given. In particular, it was shown that combined complexity of instance checking for languages with polynomial subsumption algorithm, such as $\mathcal{AL}$ and $\mathcal{ALN}$, is also polynomial. For language $\mathcal{ALE}$ it was shown that instance checking is co-NP-hard in the size of data; however, Π_2^{P} is given as the upper bound. Furthermore, it was shown that the combined complexity of instance checking in $\mathcal{ALE}$ is PSpace-complete. Finally, for $\mathcal{ALR}$, it was shown that the combined complexity of instance checking is NP-complete. It was pointed out that the additional source of complexity in $\mathcal{ALE}$ over $\mathcal{ALR}$ arises from qualified existential quantification.

None of the logics considered in [102] is as expressive as $\mathcal{ALC}$ or $\mathcal{SHIQ}(\mathbf{D})$. In particular, no considered logic allows general inclusion axioms, which are known to have a significant impact on the computation complexity. Furthermore, the complexity of reasoning in [102] is measured in the size of the ABox which is allowed to contain complex concept expressions. Our work addresses the logic with general inclusion axioms, but requires ABox not to contain complex concept expressions. Hence, our results measure the complexity in the number of simple facts, with no terminological knowledge in the ABox.

Chapter 7

Integrating Description Logics with Rules

Although $\mathcal{SHIQ}(\mathbf{D})$ is very expressive, it is a *decidable* fragment of first-order logic, and thus cannot express arbitrary axioms. More importantly, the only axioms it can express are of a certain tree-structure [49]. Decidable rule-based formalism such as function-free Horn rules[1] do not have this restriction, but lack some of the expressive power of $\mathcal{SHIQ}(\mathbf{D})$, since they are restricted to universal quantification and provide, in their basic form, no negation. To overcome the limitations of both approaches, description logics were extended with rules in [58], but this extension is undecidable [58]. Intuitively, the undecidability is due to the fact that adding rules to $\mathcal{SHIQ}(\mathbf{D})$ causes the loss of any form of *tree model property*. In a logic with such a property, every satisfiable knowledge base has a model of a certain tree-shaped form, so to decide satisfiability, it suffices to search for such a model only. For most DLs, it is possible to ensure termination of such a search.

It is natural to ask what kind of (non-tree) rules can be added to $\mathcal{SHIQ}(\mathbf{D})$ while preserving decidability. This follows a classic line of research in knowledge representation, namely to investigate the trade-off between expressivity and complexity, and to provide different formalisms with varying expressive power and complexity. This not only allows to understand the causes for the undecidability of the full combination, but also enables a more detailed analysis of the complexity and, ultimately, the design of "specialized" decision procedures. Applications that do not require the expressive power of the full combination can use such procedures, and rely on the known upper time and space bounds required to return a correct answer. Finally, in the last decade, it turned out that these specialized decision procedures are amenable to optimizations, thus achieving surprisingly good performance in practice even for logics with high worst-case complexity [5, Chapter 9].

In this chapter, we present a decidable combination of $\mathcal{SHIQ}(\mathbf{D})$ with rules, where decidability is due to restricting the rules to so-called *DL-safe* ones. Importantly, we

[1] Throughout this chapter, we use "rules" and "clauses" synonymously, following [58].

Table 7.1: Example Knowledge Base

$Person(Peter)$	Peter is a person.
$Person \sqsubseteq \exists father.Person$	Each person has a father who is a person.
$\exists father.(\exists father.Person) \sqsubseteq Grandchild$	Things having a father of a father who is a person are grandchildren.

do not restrict, the component languages, but only reduce the interface between them. Generalizing the approach of other decidable combinations of rules and description logics [72, 35], in DL-safe rules, concepts and roles are allowed to occur in both rule bodies and heads as unary, respective binary predicates in atoms, but each variable of a rule is required to occur in some body literal whose predicate is neither a concept nor a role. We discuss the expressive power and the limitations of our approach on a non-trivial example. Moreover, we show that query answering with DL-safe rules can be performed by simply appending the rules to a program obtained by the reduction from Chapter 5.

7.1 Reasons for Undecidability of $\mathcal{SHIQ}(\mathbf{D})$ with Rules

In [58], an extension of OWL-DL with rules was presented, where the integration is achieved by requiring that $N_C \cup N_{R_a} \cup N_{R_c} \subseteq N_P$. In other words, integration of OWL-DL and rules is achieved by simply allowing concepts and roles to be used in rules as unary and binary atoms, respectively. Furthermore, it was shown that such an extension unfortunately leads to undecidability of the following problem: given an OWL-DL knowledge base KB and a program P, is there a common model of $\pi(KB)$ and P, i.e. is KB consistent with P? As a consequence, subsumption and query answering w.r.t. knowledge bases and programs are also undecidable. Investigating this proof and the ones in [72] more closely, we note that the undecidability is caused by the interaction between some very basic features of description logics and rules. In this section, we try to give an intuitive explanation of this result and its consequences.

Consider the simple knowledge base KB from Table 7.1. It is not too difficult to see that this knowledge base implies the existence of an infinite chain of fathers: since *Peter* must have a father, there is some x_1 who is a *Person*. In turn, x_1 must have some father x_2, which must be a *Person*, and so on. An infinite model with such a chain is shown in Figure 7.1, upper part a). Observe that *Peter* is a grandchild, since he has a father of a father, who is a person.

Let us now check whether $KB \models Grandchild(Jane)$; this is the case if and only if $KB \cup \{\neg Grandchild(Jane)\}$ is unsatisfiable, i.e. if it does not have a model. We can check this by trying to build such a model; if we fail, then we conclude that $KB \cup \{\neg Grandchild(Jane)\}$ is unsatisfiable. However, we have a problem: starting

from *Peter*, a naïve approach to building a model will expand the chain of Peter's fathers indefinitely, and will therefore not terminate.

This very simple example intuitively shows that we have to be careful if we want to ensure termination of a satisfiability checking algorithm. For many DLs, termination can be ensured without losing completeness because we can restrict our attention to certain "nice" models. For numerous DLs, we can restrict our attention to *tree models*, i.e. to models where the underlying relational structure forms a tree [115]. This is so because every satisfiable knowledge base has such a tree model (to be precise, we actually consider tree-like abstractions of non-tree models). Even if such a tree model is infinite, we can *wind* this infinite tree model into a finite one. In our example, since KB does not require each father in the chain to be distinct (i.e. there is no axiom requiring the role *father* to be acyclic), the model in Figure 7.1, lower part b) is the result of "winding" an infinite tree into a "nice", finite model. Due to their regular structure, these "windings" of tree models can be easily constructed in an automated way. To understand why every satisfiable $\mathcal{SHIQ}(\mathbf{D})$ knowledge base has a tree model [61], consider the mapping π from tables 2.1 and 2.3 more closely (we abstract some technicalities caused by the transitive roles): in all formulae obtained by transforming the result of π into prenex normal form, variables are connected by roles only in a tree-like manner, as shown in the following example:

$$\begin{array}{ll} \exists S.(\exists R.C \sqcap \exists R.D) \sqsubseteq Q & \Rightarrow \\ \forall x : \{[\exists y : S(x,y) \wedge (\exists x : R(y,x) \wedge C(x)) \wedge (\exists x : R(y,x) \wedge D(x))] \rightarrow Q(x)\} & \Rightarrow \\ \forall x, x_1, x_2, x_3 : \{S(x,x_1) \wedge R(x_1,x_2) \wedge C(x_2) \wedge R(x_1,x_3) \wedge D(x_3) \rightarrow Q(x)\} & \end{array}$$

Let us contrast these observations with the kind of reasoning required for function-free Horn rules. In such rules, all variables are universally quantified, i.e. there are no existentially quantified variables in rule consequents. Hence, we never have to infer the existence of "new" objects. Thus, reasoning algorithms must consider only individuals which are explicitly introduced and are given a name in the knowledge base. Reasoning can be performed by *grounding* the rules, i.e. replacing the variables in the rules with all individuals from the knowledge base in all possible ways. Through grounding, first-order reasoning becomes propositional, since a ground rule is essentially equivalent to a propositional clause. For a finite program, the number of ground rules is also finite, and satisfiability of a set of propositional clauses is decidable. Hence, the rules, such as

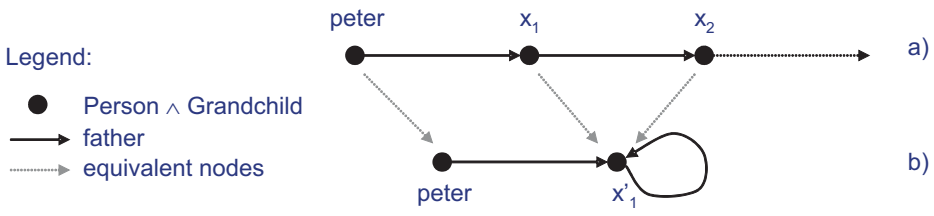

Figure 7.1: Two Similar Models

the one defining $hasAunt(x, y)$ from the introduction, are allowed to enforce arbitrary but finite, non-tree models, and not only "nice" models.

Now let us see what happens if we extend $\mathcal{SHIQ}(\mathbf{D})$ with function-free Horn rules. Then, we combine a logic whose decidability is due to the fact that we can restrict our attention to "nice" models (but with individuals whose existence may be implied by a knowledge base) with the one whose decidability is due to the fact that we can restrict our attention to "known" individuals (but with arbitrary relations between them). Unsurprisingly, this and similar combinations are undecidable [72, 58].

7.2 Combining Description Logics and Rules

We now formalize the interface between $\mathcal{SHIQ}(\mathbf{D})$ and rules.

Definition 7.2.1 (DL Rules)**.** *Let KB be a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base and let N_P be the set of predicate symbols such that $\{\approx\} \cup N_C \cup N_{R_a} \cup N_{R_c} \subseteq N_P$. For s and t constants or variables, a* DL-atom *is an atom of the form $A(s)$, where $A \in N_C$, or of the form $R(s,t)$, where $R \in N_{R_a} \cup N_{R_c}$ and is simple in KB. A* non-DL-atom *is an atom with a predicate from $N_P \setminus (N_C \cup N_{R_a} \cup N_{R_c} \cup \{\approx\})$. A (disjunctive)* DL rule *is a (disjunctive) rule allowed to contain DL- and non-DL-atoms. A DL program is a set of (disjunctive) DL rules.*

The semantics of the combined knowledge base *(KB, P), where KB is a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base and P is a DL program, is given by translation into first-order logic as $\pi(KB) \cup P$, where each rule $A_1 \vee ... \vee A_n \leftarrow B_1, ..., B_m$ is treated as a clause $A_1 \vee ... \vee A_n \vee \neg B_1 \vee ... \vee \neg B_m$. The main inferences in (KB, P) are* satisfiability checking*, i.e. determining whether a first-order model of $\pi(KB) \cup P$ exists, and* query answering*, i.e. determining whether $\pi(KB) \cup P \models \alpha$ for a ground atom α, written as $(KB, P) \models \alpha$.*

A few remarks regarding Definition 7.2.1 are in order.

Relationship with Existing Formalisms. The above definition yields a formalism compatible with the ones from [58, 72]. The main difference from [58] is that we allow non-DL-atoms to occur in a rule, and that we require only complex concepts to occur in a rule. The latter is a technical assumption and is not really a restriction: for a complex concept C, one can always introduce a new atomic concept A_C, add the axiom $A_C \equiv C$ to the TBox, and use A_C in the rule. This transformation is obviously linear in the size of P.

Decidability. Since the formalism is compatible with [58], we immediately have that the reasoning in DL rules is undecidable. To achieve decidability we introduce the notion of DL-safety in Section 7.3.

Minimal vs. First-order Models. Rules are usually interpreted under *minimal model* semantics, i.e. only models minimal w.r.t. set inclusion are considered; we write $P \models_c \alpha$ if a formula α is true in all minimal models of P. However, in Definition 7.3.1 we assume the standard first-order semantics for rules, where $P \models \alpha$ means that α is true in all models of P. We briefly discuss the differences between these two approaches, and their practical consequences.

Assume that α is a positive ground atom. It is easy to see that in such a case, $P \models \alpha$ if and only if $P \models_c \alpha$. Namely, if α is true in each model of P, it is true in each minimal model of P as well, and vice versa. Therefore, for entailment of positive ground atoms, it is not important whether the semantics of P is defined w.r.t. minimal or w.r.t. general first-order models.

Assume now that α is a negative ground atom. In this case, there is a difference between minimal model semantics and first-order semantics, as shown by the following example. For $\alpha = \neg A(b)$ and $P = \{A(a)\}$, it is clear that $P \not\models \alpha$. Namely, $\neg A(b)$ is not explicitly derivable from the facts in P: $M_1 = \{A(a), A(b)\}$ is a perfectly valid first-order model of P and α is false in M_1. However, P has exactly one minimal model $M_2 = \{A(a)\}$ and $\neg A(b)$ is obviously true in M_2, so $P \models_c \alpha$.

The type of semantics also affects concept subsumption: let $\alpha = \forall x : C(x) \rightarrow D(x)$ and $P = \{C(a), D(a)\}$. Similarly as above, $P \not\models \alpha$; namely, $M_1 = \{C(a), D(a), C(b)\}$ is a model of P in which α is false. However, $M_2 = \{C(a), D(a)\}$ is the only minimal model of P and α is true in M_2, so $P \models_c \alpha$. The distinction between minimal models and general first-order models fundamentally changes the computational properties of concept subsumption: equivalence of general programs under minimal model semantics is undecidable [110], whereas under first-order semantics it is decidable and can be reduced to satisfiability checking using standard transformations.

To summarize, the difference between first-order and minimal model semantics is not relevant for query answering if queries are positive atoms; however, it is relevant for queries which involve negation or for concept subsumption. Negative queries are usually considered in a more general framework of *negation-as-failure*, where negation is interpreted as failure to prove a query, thus yielding a *non-monotonic* formalism. Whereas non-monotonic features are certainly very important for the Semantic Web, we do not address them in this paper. Instead, our results are an initial step towards providing a practical hybrid knowledge representation formalism integrating description logics and rules. We also believe that our work may be used as a sound basis for future non-monotonic extensions.

7.3 DL-safety

We now introduce DL-safety restriction as one possible way to make reasoning with DL rules decidable.

Definition 7.3.1 (DL-safe Rules)**.** *A (disjunctive) DL rule r is* DL-safe *if each variable occurring in r also occurs in a non-DL-atom in the body of r. A (disjunctive) program P is DL-safe if all its rules are DL-safe.*

DL-safety is similar to safety in datalog. In a safe rule, each variable occurs in a positive atom in the body, and may therefore be bound only to constants explicitly present in the database. Similarly, DL-safety ensures that each variable is bound only to individuals explicitly introduced in the ABox. For example, if *Person*, *livesAt*, and *worksAt* are concepts and roles from KB, the following rule is not DL-safe:

$$Homeworker(x) \leftarrow Person(x), livesAt(x, y), worksAt(x, y)$$

The reason for this is that both variables x and y occur in DL-atoms, but do not occur in a body atom with a predicate outside of KB. This rule can be made DL-safe by adding special non-DL-atoms $\mathcal{O}(x)$, $\mathcal{O}(y)$ and $\mathcal{O}(z)$ to the body of the rule, and by adding a fact $\mathcal{O}(a)$ for each individual a occurring in KB and P. Thus, the above rule becomes

$$Homeworker(x) \leftarrow Person(x), livesAt(x, y), worksAt(x, y), \mathcal{O}(x), \mathcal{O}(y), \mathcal{O}(z)$$

This rule is obviously DL-safe. In Subsection Section 7.4 we discuss the consequences that this transformation has on the semantics.

7.4 Expressivity of DL-safe Rules

It is important to notice that, to achieve decidability, we do not restrict the component languages. Rather, we combine full $\mathcal{SHIQ}(\mathbf{D})$ with function-free Horn rules, and thus extend both formalisms. DL-safety only restricts the interchange of consequences between the component languages to those consequences involving individuals explicitly introduced in the ABox.

To illustrate the expressive power of DL-safe rules, consider the axioms and rules from Table 7.2. We use a rule to define the only non-DL-predicate *BadChild* as a grandchild who hates some of his siblings (or himself). Notice that this rule involves relations forming a triangle between two siblings and a parent, and thus cannot be expressed in $\mathcal{SHIQ}(\mathbf{D})$. Moreover, the rule is not DL-safe since each variable in the rule does not occur in a non-DL-atom in the rule body.

Now consider the first group of ABox facts. Since *Cain* is a *Person*, as in Section 7.1 one may infer that *Cain* is a *Grandchild*. Since *Cain* and *Abel* are children of *Adam*, and *Cain hates Abel*, we derive that *Cain* is a *BadChild*.

Similarly, since *Romulus* and *Remus* are persons, they have a father. Due to the second rule, the father of *Romulus* and *Remus* must be the same. Now *Romulus hates Remus*, so *Romulus* is a *BadChild* as well. We are able to derive this without knowing exactly who the father of *Romulus* and *Remus* is[2].

[2]Actually, the father of Romulus and Remus is Mars, but this fact is not well-known. Still, the father's existence is certain, and this is exactly what the knowledge base represents.

Table 7.2: Example with DL-safe Rules

$Person \sqsubseteq \exists father.Person$	Each person has a father who is a person.
$\exists father.(\exists father.Person) \sqsubseteq Grandchild$	Things having a father of a father who is a person are grandchildren.
$father \sqsubseteq parent$	Fatherhood is a kind of parenthood.
$BadChild(x) \leftarrow Grandchild(x),$ $parent(x,y), parent(z,y), hates(x,z)$	A bad child is a grandchild who hates one of his siblings.
$BadChild'(x) \leftarrow Grandchild(x),$ $parent(x,y), parent(z,y), hates(x,z),$ $\mathcal{O}(x), \mathcal{O}(y), \mathcal{O}(z)$	DL-safe version of a bad child.
$Person(Cain)$	Cain is a person.
$father(Cain, Adam)$	Cain's father is Adam.
$father(Abel, Adam)$	Abel's father is Adam.
$hates(Cain, Abel)$	Cain hates Abel.
$Person(Romulus)$	Romulus is a person.
$Person(Remus)$	Remus is a person.
$x \approx y \leftarrow father(Romulus, x), father(Remus, y)$	Romulus and Remus have the same father.
$x \approx y \leftarrow father(Romulus, x), father(Remus, y),$ $\mathcal{O}(x), \mathcal{O}(y)$	DL-safe version.
$hates(Romulus, Remus)$	Romulus hates Remus.
$Child(x) \leftarrow GoodChild(x), \mathcal{O}(x)$	Good children are children.
$Child(x) \leftarrow BadChild'(x), \mathcal{O}(x)$	Bad children are children.
$(GoodChild \sqcup BadChild')(Oedipus)$	Oedipus is a good or a bad child.
$\mathcal{O}(\alpha)$ for each explicitly named individual α	Enumeration of all ABox individuals.

Consider now the DL-safe rule defining *BadChild′*: since the father of *Cain* and *Abel* is known by name (i.e. *Adam* is in the ABox), the literal $\mathcal{O}(y)$ from the rule for *BadChild′* can be matched to $\mathcal{O}(Adam)$, and we may conclude that *Cain* is a *BadChild′*. In contrast, the father of *Romulus* and *Remus* is not known in the ABox. Hence, in the DL-safe version of the second rule, $\mathcal{O}(x)$ and $\mathcal{O}(y)$ cannot be matched to the father's name, so the rule does not derive that the fathers of *Romulus* and *Remus* are the same. Similarly, in the rule defining *BadChild′*, the literal $\mathcal{O}(y)$ cannot be matched to the father's name, so we cannot derive that *Romulus* is a *BadChild′*.

This may seem confusing. However, DL-safe rules do have a "natural" reading: just append the phrase "where the identity of all objects is known" to the meaning of the rule. For example, the rule defining *BadChild′* can be read as "A *BadChild′* is a *known* grandchild for which we *know* a parent, and who hates one of his *known* siblings."

Combining description logics with DL-safe rules increases the expressivity of both components. Namely, a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base cannot imply that *Cain* is a

BadChild′ because the "triangle" rule cannot be expressed in $\mathcal{SHIQ}(\mathbf{D})$. Similarly, a set of function-free Horn rules cannot imply this either: we know that *Cain* has a grandfather because *Cain* is a person, but we do not know who he is. Hence, we need the existential quantifier to *infer* the existence of ancestors, and then infer that *Cain* is a *Grandchild*.

Finally, we point out that it is incorrect to compute all consequences of the description logic component first, and then to apply the rules to the consequences. Consider the *KB* part about *Oedipus*: he is a *GoodChild* or a *BadChild′*, but we do not know exactly which. Either way, one of the rules derives that *Oedipus* is a *Child*, so $(KB, P) \models \mathit{Child}(\mathit{Oedipus})$. This would not be derived by applying the rules defining *Child* to the consequences of *KB*, since $KB \not\models \mathit{GoodChild}(\mathit{Oedipus})$ and $KB \not\models \mathit{BadChild}'(\mathit{Oedipus})$.

7.5 Query Answering for DL-safe Rules

In a nutshell, we show that reasoning in (KB, P) can be performed by simply appending P to $\mathsf{DD}(KB)$. To show that, we modify the proofs of several lemmata leading to Theorem 5.4.2.

For a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB, the first step in query answering is to eliminate transitivity axioms by encoding KB into an equisatisfiable $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base $\Omega(KB)$, as explained in Section 3.2. Observe that Definition 7.3.1 allows only simple roles to occur in DL-atoms, and that in Section 3.2 we show that the encoding preserves entailment of such atoms. Hence, for a DL-safe program P, it is easy to see that (KB, P) and $(\Omega(KB), P)$ are equisatisfiable. Hence, in the rest we assume without loss of generality that KB is an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base.

We now show that saturation of $\Xi(KB) \cup P$ by $\mathcal{BS}_{DL}^{\mathbf{D},+}$ decides satisfiability of (KB, P). To obtain a decision procedure, we extend the selection function of $\mathcal{BS}_{DL}^{\mathbf{D},+}$ in the following way: if a closure contains negative non-DL-atoms, then all such atoms are selected and nothing else is selected; if there are no negative non-DL-atoms, the selection function is the same as in Definition 3.3.3, i.e. it selects all negative binary literals.

We define the *extended* $\mathcal{ALCHIQ}(\mathbf{D})$-closures to include the closure types from Table 3.2 without conditions (*iii*) – (*vi*), the closure types from Table 4.2, and closures corresponding to DL-safe rules. Furthermore, closures of type 8 are allowed to contain positive or negative non-functional ground non-DL-atoms.

Lemma 7.5.1. *For an* $\mathcal{ALCHIQ}(\mathbf{D})$ *knowledge base KB, saturation of* $\Xi(KB) \cup P$ *by* $\mathcal{BS}_{DL}^{\mathbf{D},+}$ *decides satisfiability of* (KB, P).

Proof. (KB, P) is satisfiable if and only if $\Xi(KB) \cup P$ is satisfiable. Furthermore, all closures from $\Xi(KB) \cup P$ are obviously extended $\mathcal{ALCHIQ}(\mathbf{D})$-closures. To show the claim of the lemma, it is sufficient to show that the following property (*) holds: let $\Xi(KB) \cup P = N_0, \ldots, N_i \cup \{C\}$ be a $\mathcal{BS}_{DL}^{\mathbf{D},+}$-derivation, where C is the conclusion

derived from premises in N_i. Then C is either an extended $\mathcal{ALCHIQ}(\mathbf{D})$-closure or it is redundant in N_i.

All ground non-DL-atoms in $\Xi(KB) \cup P$ contain constants. Hence, a superposition into a ground non-functional non-DL-atom is possible only from a literal $\langle a \rangle \approx \langle b \rangle$, and in the superposition conclusion all non-DL-atoms contain constants. Consider an inference with some rule r. Since r is DL-safe, it can participate only in a hyperresolution inference with electrons of type 8 on non-DL-literals. Furthermore, r is safe, so, since all ground non-DL-atoms contain constants, the hyperresolution binds all variables in a rule to constants. Hence, the resolvent is a ground closure where non-DL-atoms do not contain functional terms, so it is of type 8. Closures of type 8 can participate in $\mathcal{BS}_{DL}^{\mathbf{D},+}$ inferences with other closures in exactly the same way as in Lemma 3.3.5, Lemma 4.2.1 and Theorem 3.4.6. Hence, the property (*) holds. Since $\mathcal{BS}_{DL}^{\mathbf{D},+}$ is sound and complete, the claim of the lemma follows. □

The next step is to show that rules can simply be appended to the function-free version of KB.

Lemma 7.5.2. *(KB, P) is unsatisfiable if and only if $\mathsf{FF}(KB) \cup P$ is unsatisfiable.*

Proof. (KB, P) is unsatisfiable if and only if $\Xi(KB) \cup P$ is unsatisfiable. The latter can be decided by a $\mathcal{BS}_{DL}^{\mathbf{D},+}$ saturation, in which all non-ground inferences are performed first. Since rules from P contain negative non-DL-atoms selected, they cannot participate in an inference with non-ground closures from $\Xi(KB) \cup P$. Hence, as in Lemma 5.2.1, $\Xi(KB) \cup P$ is satisfiable if and only if $\Gamma = \mathsf{Sat}_{\mathsf{R}}(\Gamma_{\mathcal{TR}g}) \cup \Xi(KB_{\mathcal{A}}) \cup P$ is satisfiable.

It is now straightforward to extend Lemma 5.2.3 to show that Γ is unsatisfiable if and only if $\mathsf{FF}(KB) \cup P$ is unsatisfiable. For both directions of the proof, a hyperresolution with a rule r in one closure set can directly be simulated with a hyperresolution with r in the other closure set. □

We now state our main result:

Theorem 7.5.3. *Let KB be an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base and P a DL-safe disjunctive datalog program. Then (KB, P) is unsatisfiable if and only if $\mathsf{DD}(KB) \cup P$ is unsatisfiable. Furthermore, $(KB, P) \models \alpha$ if and only if $\mathsf{DD}(KB) \cup P \models_c \alpha$, where α is a DL-atom $A(a)$ or $R(a, b)$, or α is a ground non-DL-atom.*

Proof. The first claim is a direct consequence of Lemma 7.5.2. For the second claim, observe that $(KB, P) \models \alpha$ if and only if $(KB \cup \{\neg\alpha\}, P)$ is unsatisfiable. This is the case if and only if $\mathsf{FF}(KB \cup \{\neg\alpha\}) \cup P = \mathsf{FF}(KB) \cup P \cup \{\neg\alpha\}$ is unsatisfiable, which is the case if and only if $\mathsf{DD}(KB) \cup P \models_c \alpha$. □

Query answering in $\mathsf{DD}(KB) \cup P$ can be performed using the algorithm from Section 5.5. We now determine its complexity.

Theorem 7.5.4. *Let KB be an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base, defined over a concrete domain $\mathbf{D}$, such that $\mathbf{D}$-satisfiability of finite conjunctions over $\Phi_{\mathbf{D}}$ can be decided in deterministic exponential time, and let P be a DL-safe program. Assuming unary coding of numbers and a bound on the arity of concrete domain predicates, computing all answers to a non-ground query in (KB, P) can be done in time exponential in $|KB| + |P|$, assuming a bound on the arity of predicates in P, and in time doubly exponential in $|KB| + |P|$ otherwise.*

Proof. Similarly to Lemma 3.3.8, to determine the complexity of the algorithm we compute the maximal number of closures derived during saturation by $\mathcal{BS}_Q^{\mathbf{D}}$. Let p denote the number of non-DL-predicates, r the maximal arity of a predicate, c the number of constants occurring in (KB, P), and b the maximal number of literals in a body of a rule from P. Under the assumptions of the theorem, p, c and b are linear in $|KB| + |P|$. If r is not bounded, it is also linear in $|KB| + |P|$.

The number of non-DL-literals occurring in a maximal ground closure is bounded by $\ell_1 = 2p(2c)^r$ (the first factor 2 allows each literal to occur positively or negatively, and the second factor 2 allows each term to be marked or not). If the predicate arity is bounded, then ℓ_1 is polynomial, and if the predicate arity is unbounded, it is exponential in $|KB| + |P|$. From Lemma 3.3.8 we know that the maximal number of DL-atoms in a closure, denoted as ℓ_2, is polynomial in $|KB|$ assuming a bound on the arity of concrete domain predicates and for unary coding of numbers. Since each ground closure can contain an arbitrary subset of these literals, the maximal number of ground closures derived by $\mathcal{BS}_Q^{\mathbf{D}}$ is bounded by $k = 2^{\ell_1+\ell_2}$, which is exponential for bounded arity, and doubly exponential for unbounded arity of predicates in P.

Each rule from $\mathsf{DD}(KB) \cup P$ can participate in a hyperresolution inference with ground closures in k^b ways, which is exponential in $|P|$. Furthermore, by Theorem 5.4.2 the number of rules t in $\mathsf{DD}(KB) \cup P$ is exponential in $|KB| + |P|$. Hence, the number of hyperresolution inference steps is bounded by $tk^b = t \cdot 2^{b\cdot(\ell_1+\ell_2)}$. For bounded arity of predicates in P, this number is exponential, and doubly exponential otherwise. Hence, in the same way as in Lemma 4.2.4, the number of concrete domain inference steps is exponential for bounded arity of predicates in P, and doubly exponential otherwise, thus implying the claim of the theorem. □

Note that most applications require predicates of small arity. Hence, the assumption that the arity of predicates is bounded is realistic in practice, thus giving a worst-case optimal algorithm.

Finally, it is easy to see that the same results apply even if the description logic component is $\mathcal{ALCHIQ}b(\mathbf{D})$, instead of $\mathcal{ALCHIQ}(\mathbf{D})$.

7.6 Related Work

$\mathcal{AL}$-log [35] is a logic which combines a TBox and ABox expressed in the basic description logic $\mathcal{ALC}$ with datalog rules, which may be constrained with unary atoms

having $\mathcal{ALC}$ concepts as predicates in the body. Query answering in $\mathcal{AL}$-log is decided by a variant of constrained resolution, combined with a tableaux algorithm for $\mathcal{ALC}$. The combined algorithm is shown to run in single non-deterministic exponential time. The fact that atoms with concept predicates can occur only as constraints in the body makes rules applicable only to explicitly named objects. Our restriction to DL-safe rules has the same effect. However, our approach is more general in the following ways: (*i*) it supports a more expressive description logic, (*ii*) it allows using both concepts and roles in DL-atoms and (*iii*) DL-atoms can be used in rule heads as well. Furthermore, (*iv*) we present a query answering algorithm as an extension of deductive database techniques which runs in deterministic exponential time.

A comprehensive study of the effects of combining datalog rules with description logics is presented in [72]. The logic considered is $\mathcal{ALCNR}$, which, although less expressive than $\mathcal{SHIQ}$, contains constructors that are characteristic of most DL languages. The results of the study can be summarized as follows: (*i*) answering conjunctive queries over $\mathcal{ALCNR}$ knowledge bases is decidable, (*ii*) query answering in a logic obtained by extending $\mathcal{ALCNR}$ with non-recursive datalog rules, where both concepts and roles can occur in rule bodies, is also decidable, as it can be reduced to computing a union of conjunctive query answers, (*iii*) if rules are recursive, query answering becomes undecidable, (*iv*) decidability can be regained by disallowing certain combinations of constructors in the logic, and (*v*) decidability can be regained by requiring rules to be *role-safe*, where at least one variable from each role literal must occur in some non-DL-atom. As in $\mathcal{AL}$-log, query answering is decided using constrained resolution and a modified version of the tableaux calculus. Besides the fact that we treat a more expressive logic, in our approach all variables in a rule must occur in at least one non-DL-atom, but concepts and roles are allowed to occur in rule heads. Hence, when compared to the variant (*v*), our approach is slightly less general in some, and slightly more general in other aspects.

The Semantic Web Rule Language (SWRL) [58] combines OWL-DL with rules in which concept and role predicates are allowed to occur in the head and in the body, without any restrictions. Hence, apart from technicalities such as allowing concept expressions to occur in the rules, the formalism is compatible with DL rules. As mentioned before, this combination is undecidable but, as pointed out by the authors, (incomplete) reasoning in such a logic can be performed using general first-order theorem provers. DL-safe rules are a proper subset of SWRL, where some expressivity is traded for decidability. Furthermore, we provide an optimal query answering algorithm covering a significant portion of OWL-DL.

In [40] an approach for combining answer set programming with description logic reasoning was presented. The interaction between the subsystems is enabled by exchanging ground consequences between the two components. Hence, the consequences of the description logic knowledge base can be pushed as facts into the answer set program and vice versa. The final set of derived facts is obtained by fixpoint computation. In this approach, the two systems are not tightly integrated, since interaction between the systems is performed through the exchange of consequences only.

The approaches from [49] and [116] for reducing certain fragments of description logics to logic programming can easily be extended with rules, by simply appending the rules to the result of the transformation. However, the description logic considered there does not support existential quantifiers, negation, or disjunction under positive polarity, so it is significantly less expressive than $\mathcal{SHIQ}(\mathbf{D})$. Hence, our approach is a proper extension.

Chapter 8

Answering Conjunctive Queries

Conjunctive queries have been introduced in [29] as a formalism capable of expressing the class of selection/projection/join/renaming relational queries [2]. The vast majority of relational queries used in the practice can be expressed using the formalism of conjunctive queries, so a great deal of database research has been devoted to devising efficient algorithms for query answering and deciding query containment.

Since conjunctive queries have been found useful in diverse practical applications, it is natural to consider conjunctive queries as an expressive formalism for querying description logic knowledge bases. Hence, in this chapter we extend our algorithms to handle reasoning with conjunctive queries. We first develop the query answering algorithm, and then show how to apply this algorithm to decide query containment. To the best of our knowledge, this is the first attempt to reasoning with conjunctive queries over description logic knowledge bases in the framework of resolution.

8.1 Definition of Conjunctive Queries

The description logic we consider is $\mathcal{SHIQ}(\mathbf{D})$. However, to eliminate transitivity axioms, we encode a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB into an equisatisfiable $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base $\Omega(KB)$. As mentioned in Section 3.2, this transformation does not preserve entailment of ground complex role atoms. Hence, in the rest we prohibit the usage of complex roles in conjunctive queries, and focus on $\mathcal{ALCHIQ}(\mathbf{D})$.

Definition 8.1.1. *Let KB be an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base, and let $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$ be sets of* distinguished *and* non-distinguished *variables, denoted as $\mathbf{x}$ and $\mathbf{y}$, respectively. A* conjunctive query *over KB, written as $Q(\mathbf{x}, \mathbf{y})$, is a conjunction of DL-atoms of the form $(\neg)A(s)$ or $R(s,t)$, where s and t are individuals from KB, distinguished or non-distinguished variables. The basic inferences are:*

- Query answering. *An* answer *of a query $Q(\mathbf{x}, \mathbf{y})$ w.r.t. KB is an assignment θ of individuals to distinguished variables, such that $KB \models \exists \mathbf{y} : Q(\mathbf{x}\theta, \mathbf{y})$.*

- Query containment. *A conjunctive query* $Q_2(\mathbf{x}, \mathbf{y_1})$ *is contained in a conjunctive query* $Q_1(\mathbf{x}, \mathbf{y_2})$ *w.r.t. KB if* $KB \models \forall \mathbf{x} : [\exists \mathbf{y_2} : Q_2(\mathbf{x}, \mathbf{y_2}) \rightarrow \exists \mathbf{y_1} : Q_1(\mathbf{x}, \mathbf{y_1})]$.

Negative concept atoms are usually not allowed in a conjunctive query. However, allowing such atoms to occur in conjunctive queries makes the following presentation much simpler. Such a definition is fully compatible with all previously considered definitions.

8.2 Answering Conjunctive Queries

Let KB be an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base. It is easy to see that, for a conjunctive query $Q(\mathbf{x}, \mathbf{y})$, the assignment θ such that $\theta\mathbf{x} = \mathbf{a}$, is an answer of the query w.r.t. KB if and only if the set of closures $\Gamma' = \Xi(KB) \cup \{\neg Q(\mathbf{a}, \mathbf{y})\}$ is unsatisfiable, where $\neg Q(\mathbf{a}, \mathbf{y})$ is a closure obtained by negating each conjunct of $Q(\mathbf{a}, \mathbf{y})$. To decide satisfiability of Γ' by basic superposition, we define first the notion of a query graph:

Definition 8.2.1. *A* query graph *for a conjunctive query* $Q(\mathbf{a}, \mathbf{y})$ *is a directed graph with the following structure, where* s *and* t *are terms in the query:*

- *Each variable* $y \in \mathbf{y}$ *is associated with a unique node.*
- *Each occurrence of a constant in* $Q(\mathbf{a}, \mathbf{y})$ *is associated with a unique node, i.e. occurrences of the same constant are associated with distinct nodes.*
- *For each literal* $(\neg)A(s) \in Q(\mathbf{a}, \mathbf{y})$*, the node* s *is labeled with* $(\neg)A$*.*
- *For each literal* $R(s, t) \in Q(\mathbf{a}, \mathbf{y})$*, the nodes* s *and* t *are connected with a directed arc labeled* R*.*

Each conjunctive query defines a distinct query graph, so we do not make an explicit distinction between the two. Hence, by saying that a query is connected, tree-like or acyclic, we refer to the properties of the query graph.

Without loss of generality, we can assume that a conjunctive query graph is weakly connected, i.e. that for each two nodes s and t in the graph, there is either a path from s to t, or a path from t to s. Namely, assume that a conjunctive query $Q(\mathbf{a}, \mathbf{y})$ can be split into n weakly connected mutually disjoint subqueries $Q_1(\mathbf{a_1}, \mathbf{y_1}), \ldots, Q_n(\mathbf{a_n}, \mathbf{y_n})$. It is obvious that $KB \models \bigwedge_{1 \leq i \leq n} Q_i(\mathbf{a_i}, \mathbf{y_i})$ if and only if $KB \models Q_i(\mathbf{a_i}, \mathbf{y_i})$ for all $1 \leq i \leq n$. The subqueries $Q_i(\mathbf{a_i}, \mathbf{y_i})$ can be computed in polynomial time, so this assumption does not increase the complexity of reasoning.

A slight problem arises if $\neg Q(\mathbf{a}, \mathbf{y})$ contains unmarked constants. In such a case, assuming that $a_i \in \mathbf{a_i}$ and $a'_i \in \mathbf{a'_i}$ for $i \in \{1, 2\}$, a superposition of $a_1 \approx a'_1 \vee a_2 \approx a'_2$ into $\neg Q_1(\mathbf{a_1}, \mathbf{y_1})$ and $\neg Q_2(\mathbf{a_2}, \mathbf{y_2})$ may produce a closure $\neg Q_1(\mathbf{a'_1}, \mathbf{y_1}) \vee \neg Q_2(\mathbf{a'_2}, \mathbf{y_2})$. Such an inference produces a conclusion with more variables than each of its premises, thus leading to non-termination. To prevent this, we apply the structural transformation

to $\neg Q(\mathbf{a}, \mathbf{y})$ and replace Γ' with Γ, where for each $a \in \mathbf{a}$, $\mathcal{O}_a$ is a new predicate unique for a, x_a is a new variable unique for a, and $\mathbf{x_a}$ is the vector of variables obtained from $\mathbf{a}$ by replacing each $a \in \mathbf{a}$ with x_a:

$$\Gamma = \Xi(KB) \cup \{\neg Q(\mathbf{x_a}, \mathbf{y}) \vee \bigvee_{a \in \mathbf{a}} \neg \mathcal{O}_a(x_a)\} \cup \bigcup_{a \in \mathbf{a}} \{\mathcal{O}_a(a)\}$$

The sets Γ' and Γ are obviously equisatisfiable. In the rest we write $\neg \mathcal{O}_{\mathbf{a}}(\mathbf{x_a})$ for $\bigvee_{a \in \mathbf{a}} \neg \mathcal{O}_a(x_a)$. We now define the calculus used to decide query answering:

Definition 8.2.2. *$\mathcal{BS}^{\mathbf{D},+}_{CQ}$ is the $\mathcal{BS}^{\mathbf{D}}$ calculus parameterized as follows:*

- *The precedence for LPO is $f >_P c >_P p >_P \mathcal{O}_a >_P Q_{R,f} >_P p_{a,b} >_P \top$ for any function symbol f, constant c, non-definition predicate p, predicate $\mathcal{O}_a$, definition predicate $Q_{R,f}$, and propositional symbol $p_{a,b}$.*
- *If a closure C contains a literal $\neg \mathcal{O}_a(x_a)$, then all such literals are selected; otherwise, all negative binary literals are selected.*
- *Inference conclusions, whenever possible, are decomposed according to the following table, where t_i are terms of the form $f_{i,1}(\dots f_{i,m}(x)\dots)$:*

$$\begin{array}{lcl}
D \cdot \rho \vee R([t], [f(t)]) & \rightsquigarrow & \begin{array}{c} D \cdot \rho \ \vee \ Q_{R,f}([t]) \\ \neg Q_{R,f}(x) \ \vee \ R(x, [f(x)]) \end{array} \\[2ex]
D \cdot \rho \vee R([f(t)], [t]) & \rightsquigarrow & \begin{array}{c} D \cdot \rho \ \vee \ Q_{\mathsf{Inv}(R),f}([t]) \\ \neg Q_{\mathsf{Inv}(R),f}(x) \ \vee \ R([f(x)], x) \end{array} \\[2ex]
(\neg)A_1([t_1]) \vee \ldots \vee (\neg)A_n([t_n]) & \rightsquigarrow & \begin{array}{c} Q_{(\neg)A_1,t_1}(x) \vee \ldots \vee Q_{(\neg)A_n,t_n}(x) \\ \neg Q_{(\neg)A_i,t_i}(x) \vee (\neg)A_i([t_i]),\ 1 \leq i \leq n \end{array} \\[2ex]
C \cdot \rho \vee \mathcal{O}_a(\langle b \rangle) & \rightsquigarrow & \begin{array}{c} C \cdot \rho \vee p_{a,b} \\ \neg p_{a,b} \vee \mathcal{O}_a(b) \end{array}
\end{array}$$

We extend the $\mathcal{ALCHIQ}(\mathbf{D})$-closures to closures obtained in a saturation of Γ by $\mathcal{BS}^{\mathbf{D},+}_{CQ}$:

Definition 8.2.3. *The class of $\mathcal{CQ}$-closures w.r.t. a conjunctive query $Q(\mathbf{a}, \mathbf{y})$ over an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base KB is obtained as a generalization of closures from Table 3.2 and Table 4.2, with the following changes:*

- *Conditions (*iii*) – (*vi*) are dropped.*
- *Closure types 5 and 6 are replaced with a new type $5'$, which contains all closures C satisfying each of the following conditions:*

1. *C contains only equality, unary or propositional literals.*
2. *C contains only one variable x.*
3. *The depth of a term in C is bounded by the number of literals of $Q(\mathbf{a}, \mathbf{y})$.*
4. *If C contains a term of the form $f(t)$, then all terms of the same depth in C are of the form $g(t)$, and all terms of smaller depth are (not necessarily proper) subterms of t.*
5. *Only the outmost position of a term in C can be unmarked, i.e. each functional term is either of the form $[f(t)]$ or of the form $f([t])$.*
6. *Equality and inequality literals in C can have the form $[f(t)] \circ [g(t)]$ or $[f(g(t))] \circ [t]$ for $\circ \in \{\approx, \not\approx\}$.*

- *Closure type 8 is modified to allow unary and (in)equality literals to contain unary terms whose depth is bounded by the number of literals in $Q(\mathbf{a}, \mathbf{y})$; only outermost positions in a term can be unmarked; all (in)equality literals are of the form $[f(a)] \circ [b]$, $[f(t)] \circ [g(t)]$, $[f(g(t))] \circ [t]$ or $\langle a \rangle \circ \langle b \rangle$, for $\circ \in \{\approx, \not\approx\}$ and t a ground term; and a closure can contain propositional literals $(\neg)p_{a,b}$.*
- *A new* query closure *type contains closures of the form $\neg Q([\mathbf{a}], \mathbf{y}) \vee \mathbf{p}$, where $Q([\mathbf{a}], \mathbf{y})$ is weakly connected, it contains at least one binary literal and $\mathbf{p}$ is a possibly empty disjunction of propositional literals $\mathbf{p} = \bigvee(\neg)p_{a,b}$.*
- *A new* initial closure *type contains closures of the form $\neg \mathcal{O}_{\mathbf{a}}(\mathbf{x_a}) \vee \neg Q(\mathbf{x_a}, \mathbf{y})$.*

We now show that saturation of Γ by $\mathcal{BS}^{\mathbf{D},+}_{CQ}$ terminates for each Γ, thus yielding a decision procedure for answering conjunctive queries over KB:

Theorem 8.2.4. *For a conjunctive query $Q(\mathbf{a}, \mathbf{y})$ over an $\mathcal{ALCHIQ}$ knowledge base KB, saturation of Γ by $\mathcal{BS}^{\mathbf{D},+}_{CQ}$ decides satisfiability of Γ in time doubly exponential in $|KB| + |Q(\mathbf{a}, \mathbf{y})|$, assuming a bound on the arity of the concrete domain predicates and for unary coding of numbers in input.*

Proof. As in Lemma 3.3.5, Condition 5 of Definition 8.2.3 ensures that in each $\mathcal{CQ}$-closure of type $5'$, the maximal literal contains the deepest term of a closure. Based on this, we show the following property (*): application of a $\mathcal{BS}^{\mathbf{D},+}_{CQ}$ inference to $\mathcal{CQ}$-closures results in one or more $\mathcal{CQ}$-closures.

Namely, hyperresolution with a closure of type 7 is possible only with closures of types 3, 4 and 8, and results in a closure of type $5'$ or 8. Furthermore, each conclusion of a superposition inference into a generator closure is decomposed into a generator and a closure of type $5'$. Notice that, since $R([f(t)], [t])$ and $\mathsf{Inv}(R)([t], [f(t)])$ are logically equivalent due to the translation operator π, the predicate $Q_{\mathsf{Inv}(R),f}$ can be used as the definition predicate for $R([f(x)], x)$. Finally, since only outermost position of any term may be unmarked, two terms t_1 and t_2 can be unified only at the outer positions. Since both terms are unary, they may be unified only if one of them is of

the form $f_1(\dots f_i(x)\dots)$ and the other is of the form $f_1(\dots f_i(\dots f_n(x')\dots)\dots)$, so the unifier is of the form $x \mapsto f_{i+1}(\dots f_n(x')\dots)$. Therefore, the maximal term depth in the conclusion is n, and the conclusion is a $\mathcal{CQ}$-closure.

Observe that, due to decomposition, all ground closures involving the $\mathcal{O}$ predicate are either of the form $\mathcal{O}_a(a)$ or of the form $\neg p_{a,b} \vee \mathcal{O}_a(b)$. Consider an inference with an initial closure of the form $\neg\mathcal{O}_{\mathbf{a}}(\mathbf{x_a}) \vee \neg Q(\mathbf{x_a}, \mathbf{y})$. For all variables x_a this closure contains a literal $\neg\mathcal{O}_a(x_a)$, and all such literals are selected, the only possible inference is by hyperresolution on all $\neg\mathcal{O}_a(x_a)$. This inference generates a query closure of the form $\neg Q([\mathbf{a}], \mathbf{y}) \vee \mathbf{p}$.

A more complex case is an inference with a query closure $\neg Q([\mathbf{a}], \mathbf{y}) \vee \mathbf{p}$. All constants in such a closure are marked, so superposition into it is not possible. Furthermore, since the closure always contains at least one binary literal, it may only participate as a main premise in a hyperresolution with a unifier σ on all binary literals, with side premises E_i being of type 3, 4 or 8. For side premises of type 3 and 4, with x_i we denote the free variable of the i-th side premise. Observe that $Q([\mathbf{a}], \mathbf{y})$ is weakly connected.

Assume that at least one side premise is of type 8 or that $Q([\mathbf{a}], \mathbf{y})$ contains a constant term. Then, some term in $Q([\mathbf{a}], \mathbf{y})\sigma$ is a ground term α. Let E_i be the side premise matched to the binary literal containing α and some other term β. If E_i is ground, then β is obviously a ground term, and if E_i is non-ground, since the maximal literal of E_i can only be of the form $R(x_i, \langle f(x_i)\rangle)$ or $R([f(x_i)], x_i)$, $E_i\sigma$ is ground, so β is again a ground term. Since $Q([\mathbf{a}], \mathbf{y})$ is weakly connected, constants are propagated to the entire query, so $Q([\mathbf{a}], \mathbf{y})\sigma$ is ground. Since all functional terms in side premises are of the form $f_i(x_i)$ and are of depth one, the maximal depth of a functional term after unification is bounded by the length of the maximal path in $Q([\mathbf{a}], \mathbf{y})$, which is bounded by the number of binary literals in $Q([\mathbf{a}], \mathbf{y})$. Hence, the conclusion is a $\mathcal{CQ}$-closure of type 8.

Assume that no side premise is of type 8 and that $Q([\mathbf{a}], \mathbf{y})$ does not contain a constant. Since $Q([\mathbf{a}], \mathbf{y})$ is weakly connected, similarly as in the previous case we conclude that the unifier σ contains mappings of the form $x_i \mapsto s_i$ and $y_i \mapsto t_i$, where s_i and t_i are terms of the form $f_{i,1}(\dots f_{i,m}(x)\dots)$, and m is bounded by the length of the maximal path in $Q([\mathbf{a}], \mathbf{y})$. Hence, the hyperresolution conclusion C contains only unary literals of the form $(\neg)A([t_i])$, where t_i is of the form $f_{1,m}(\dots f_{i,m}(x)\dots)$ and m is bounded by the number of binary literals in $Q([\mathbf{a}], \mathbf{y})$. Since the conclusion does not have equality literals, it satisfies conditions 1, 2, 3, 5 and 6 of the $\mathcal{CQ}$-closure type $5'$, but terms t_i need not satisfy condition 4. However, the closure is decomposed by $\mathcal{BS}^{\mathbf{D},+}_{CQ}$ into several $\mathcal{CQ}$-closures.

This covers all different $\mathcal{BS}^{\mathbf{D},+}_{CQ}$ inferences on $\mathcal{CQ}$-closures, so the property (*) holds.

Let p and f be the number of predicates and function symbols occurring in $\Xi(KB)$, which are linear in $|KB|$ for unary coding of numbers. The number p' of predicates $Q_{S,f}$ introduced by decomposition after superposition into a generator is quadratic in $|KB|$. Similarly, the number p'' of propositional symbols $p_{a,b}$ introduced by decomposition

into a literal $\mathcal{O}_a(b)$ is also quadratic in $|KB|$. For n the number of literals in $Q(\mathbf{a}, \mathbf{y})$, the number of terms of the form $f_1(\ldots f_i(x)\ldots)$ is bounded by $\ell = f + f^2 + \ldots + f^n$, which is exponential in $|KB| + |Q(\mathbf{a}, \mathbf{y})|$.

Consider a hyperresolution inference with a query closure. The conclusion of such an inference is decomposed only if the conclusion is non-ground, which is possible only if all premises are of type 3 or 4. Therefore, the hyperresolution conclusion may only contain predicates from $\Xi(KB)$ and definition predicates $Q_{S,f}$. The number of such predicates is bounded by $2(p + p')$ (the factor 2 takes into account that unary literals may occur positively or negatively), so the number of predicates $Q_{(\neg)A,t}$ introduced by decomposition after resolution with a query closure is bounded by $\wp = 2(p + p')\ell$, which is exponential in $|KB| + |Q(\mathbf{a}, \mathbf{y})|$. Furthermore, the number of literals in the longest closure of type $5'$ is bounded by $2(\wp + p + p')\ell + p''$, which is exponential in $|KB| + |Q(\mathbf{a}, \mathbf{y})|$. Similarly, the maximal number of ground functional terms is $c \cdot \ell$, where c is the number of constants in $\Xi(KB)$, which gives an exponential bound on the length of the maximal closure of type 8. Since in all cases a maximal closure is exponential in length, the number of possible $\mathcal{CQ}$-closures is doubly exponential in $|KB| + |Q(\mathbf{a}, \mathbf{y})|$.

For Γ as assumed by the theorem, all closures in Γ are $\mathcal{CQ}$-closures: closures from $\Xi(KB)$ are $\mathcal{ALCHIQ}$-closures, and $\neg Q(\mathbf{a}, \mathbf{y})$ is either an initial closure, or of type $5'$, 7 or 8. By property (*), in any derivation $\Gamma = N_0, \ldots, N_i$, each set N_i contains only $\mathcal{CQ}$-closures. Since the number of closures in each N_i is at most doubly exponential in $|KB|+|Q(\mathbf{a}, \mathbf{y})|$, saturation by $\mathcal{BS}^{\mathbf{D},+}_{CQ}$ terminates in doubly exponential time. Since the number of predicates introduced by decomposition is finite, by Theorem 3.4.4 $\mathcal{BS}^{\mathbf{D},+}_{CQ}$ is sound and complete, so it decides satisfiability of Γ. □

8.3 Deciding Conjunctive Query Containment

The algorithm for answering conjunctive queries over $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge bases is a versatile tool which can be used for other reasoning tasks with conjunctive queries. In this section we show that it can be used to decide containment of conjunctive queries.

Lemma 8.3.1. *Let $Q_1(\mathbf{x}, \mathbf{y_1})$ and $Q_2(\mathbf{x}, \mathbf{y_2})$ be two conjunctive queries with the same set of distinguished variables over an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base KB. The query $Q_2(\mathbf{x}, \mathbf{y_1})$ is contained in the query $Q_1(\mathbf{x}, \mathbf{y_2})$ w.r.t. KB if $\mathbf{a}$ is an answer to $Q_1(\mathbf{x}, \mathbf{y_1})$ over $KB \cup \{Q_2(\mathbf{a}, \mathbf{b})\}$, where $\mathbf{a}$ and $\mathbf{b}$ are sets of new distinct individuals, not occurring in $Q_1(\mathbf{x}, \mathbf{y_1})$, $Q_2(\mathbf{x}, \mathbf{y_2})$ and KB.*

Proof. The claim of the lemma can easily be established by transforming the definition of query containment by well-known identities, as presented below:

$$
\begin{array}{c}
KB \models \forall \mathbf{x} : [\exists \mathbf{y_2} : Q_2(\mathbf{x}, \mathbf{y_2}) \rightarrow \exists \mathbf{y_1} : Q_1(\mathbf{x}, \mathbf{y_1})] \quad \Leftrightarrow \\
KB \cup \{\neg \forall \mathbf{x} : [\neg \exists \mathbf{y_2} : Q_2(\mathbf{x}, \mathbf{y_2}) \vee \exists \mathbf{y_1} : Q_1(\mathbf{x}, \mathbf{y_1})]\} \text{ is unsatisfiable} \quad \Leftrightarrow \\
KB \cup \{\exists \mathbf{x}, \mathbf{y_2} : Q_2(\mathbf{x}, \mathbf{y_2}) \wedge \forall \mathbf{y_1} : \neg Q_1(\mathbf{x}, \mathbf{y_1})\} \text{ is unsatisfiable} \quad \Leftrightarrow \\
KB \cup \{Q_2(\mathbf{a}, \mathbf{b}), \forall \mathbf{y_1} : \neg Q_1(\mathbf{a}, \mathbf{y_1})\} \text{ is unsatisfiable} \quad \Leftrightarrow \\
KB \cup \{Q_2(\mathbf{a}, \mathbf{b})\} \models \exists \mathbf{y_1} : Q_1(\mathbf{a}, \mathbf{y_1})\} \quad \Leftrightarrow \\
\mathbf{a} \text{ is an answer to } Q_1(\mathbf{x}, \mathbf{y_1}) \text{ over } KB \cup \{Q_2(\mathbf{a}, \mathbf{b})\}
\end{array}
$$

In the above transformation $\mathbf{a}$ and $\mathbf{b}$ are introduced by skolemization of $\exists \mathbf{x}, \mathbf{y_2}$. □

8.4 Related Work

Conjunctive queries have been introduced in [29] as a query language for the relational model, capable of expressing a large number of practically relevant queries. The problem of deciding conjunctive query containment was shown to be NP-complete, where the algorithm is based on deciding whether a homomorphism (i.e. an embedding of non-distinguished variables) between the subsumed and the subsuming query exists. Furthermore, the authors identify the close relationship between query answering and query containment. In practice, query equivalence is used extensively for query optimization in relation databases: the general idea is to transform a complex query into an equivalent query which can be executed more efficiently [2].

In [109] it was shown that deciding equivalence of recursive queries is undecidable. However, it is important to understand that this is so only if the minimal fixpoint semantics is assumed for the queries. Such a semantics is common in logic programming; however, it is principally different from the first-order semantics considered in our work. Hence, the results from [109] do not directly apply to our results.

Answering conjunctive queries and conjunctive query containment over description logic knowledge bases was studied in the CARIN system [72]. The description logic considered is $\mathcal{ALCNR}$, and is significantly less expressive than the logic considered in our work. The decision procedure for query answering is based on constrained resolution, which combines SLD-resolution backward-chaining with tableaux reasoning.

In [26] the authors study the containment of conjunctive queries over constraints, expressed using the description logic $\mathcal{DLR}_{reg}$. This logic is distinguished by allowing n-ary relations and regular expressions over projections of relations. The technique used to decide query containment is based on a reduction to satisfiability of CPDL$_g$ (propositional dynamic logic with converse and graded modalities) programs. A similar technique was used to derive a procedure for rewriting queries over description logics using views in [28].

In [60] a procedure for deciding containment of conjunctive queries over constraints, based on the reduction to $\mathcal{SHIQ}$, was presented. In this way the authors obtain a practical procedure. Namely, they argue that the approach from [26] is not practical, since a reasoner for CPDL$_g$ does not exists.

In [63] conjunctive queries have been proposed as the query language for the Semantic Web. The approach presented there is restricted only to tree-like queries, possibly

containing constants. However, the approach was generalized later in [113], where an algorithm for answering conjunctive queries over $\mathcal{SH}f$ knowledge bases was presented, thus being the first algorithm for answering conjunctive queries over a logic with transitive roles. The algorithm reduces query answering to deciding satisfiability of $\mathcal{SHIQ}$ knowledge bases.

Contrary to the approach from [113], our approach supports inverse roles, but it allows only simple roles to occur in the queries. To the best of our knowledge, this is the first algorithm for query answering and deciding query containment based on the resolution framework.

Chapter 9

Semantics of Metamodeling

Most description logics separate the domain being modeled into two distinct parts: the intensional part, consisting of concepts and roles, and the extensional part, consisting of individuals and relations among them. Although such a modeling style is intuitive, in complex domains it is often difficult to decide whether an entity should be modeled as a concept or as an individual.

A classical example has been presented in [108] for the biology domain. Consider modeling relations among different species, and classifying animal specimens into appropriate species. In such a model, the notion of an "Ape" might be considered an individual of the "Species" concept. This modeling choice is justified by fact that "Species" might be viewed as the set of all species, containing further species, such as "Cat" or "Dog". At the same time, one might want to represent information about particular apes in the same knowledge base as well, such as "Nkima is an Ape". This observation suggests that "Ape" should be modeled as a concept.

Another discussion on metamodeling was presented in [117], where it was argued that the choice whether some entity is a concept or an individual is context dependent. A flexible knowledge representation system should be able to easily switch between the contexts, thus providing a concept or an individual view on demand.

Such concerns exerted significant influence on the design of the family of OWL languages used for ontology modeling in the Semantic Web. As the result, OWL-Full [91] — the most general language of the family — supports metamodeling, i.e. it allows a symbol to be used as a concept and as an instance in a knowledge base. Apart from metamodeling, OWL-Full allows transitive roles to occur in number restrictions. As this is known to lead to undecidability [61], OWL-Full is trivially undecidable. Currently, various first-order theorem proving techniques provide the only known means for sound but incomplete reasoning in OWL-Full, and are as such often too cumbersome for the application in the Semantic Web.

In contrast, many practical algorithms for reasoning with very expressive description logics are known [61, 56]. To be able to reuse these algorithms in the Semantic Web, the OWL-DL variant of OWL was defined, which enforces all necessary constraints needed to make reasoning decidable. Since OWL-DL follows the traditional

modeling paradigm where concepts and instances are strictly separated, metamodeling is not allowed in OWL-DL. In fact, the definition of OWL-DL explicitly requires the sets of atomic concepts, individuals, abstract and concrete roles to be pairwise disjoint, thus even prohibiting metamodeling at the syntactic level.

In this chapter we investigate whether it is possible to extend OWL-DL with metamodeling features, to obtain a decidable logic with a practical reasoning algorithm. To achieve this goal, we first investigate whether metamodeling features of OWL-Full themselves are decidable, i.e. whether imposing well-known restrictions of OWL-DL on OWL-Full yields a decidable logic. Unfortunately, as we show in Section 9.1, even $\mathcal{ALC}$-Full— the basic $\mathcal{ALC}$ description logic extended with metamodeling features of OWL-Full— is undecidable.

A careful analysis of this undecidability result shows that the problem lies is the fact that OWL-Full not only allows interpreting concepts as individuals, but also allows using modeling primitives (objects from the meta-language) as first-class objects. It is not difficult to argue that allowing modelers to redefine the semantics of a modeling primitive is simply a bad idea. Hence, in Section 9.2, we consider two alternative semantics for metamodeling. The first one we call *contextual*, which is essentially equivalent to the first-order semantics considered thus far. Furthermore, we present a so-called *HiLog* semantics for metamodeling which corresponds more closely with OWL-Full. To achieve decidability, we prohibit using modeling primitives as first-class objects. The name of the semantics if due to HiLog [30] — a logic which simulates second-order reasoning in a first-order setting — which use as the logical foundation for our proposal. We show that our algorithms from chapters 3 and 4 can easily be extended to handle such a semantics.

Finally, in Section 9.3, we analyze the added expressivity of metamodeling. It turns out that, form a logical perspective, metamodeling does not add much, and that the only new consequences that can be drawn involve reasoning with equality. By taking this into account, along with the added complexity of the reasoning procedure and the non-standard semantics, it is not clear whether metamodeling is really needed at the logical level.

9.1 Undecidability of Metamodeling in OWL-Full

In this section we show that the metamodeling features of OWL-Full are undecidable, even if combined with a very simple description logic $\mathcal{ALC}$. We start by presenting the definition of the $\mathcal{ALC}$-Full syntax and semantics. We base the semantics on the semantics of OWL-Full [91]; however, this semantics is quite complex, so we abstract numerous technical details to make it easier to understand. Essentially, the semantics is defined by mapping an $\mathcal{ALC}$-Full knowledge base into a set of triples cf. [91, Section 4], and then interpreting the set of triples cf. [91, Section 5]. We assume *rdf:*, *rdfs:* and *owl:* to be the RDF, RDFS and OWL namespace prefixes, respectively, cf. [91].

Definition 9.1.1. *Let N be the set of* names. *Then each name is an $\mathcal{ALC}$-Full concept and, for $A, R \in N$, $\neg C$, $C_1 \sqcap C_2$, $C_1 \sqcup C_2$, $\exists R.C$ and $\forall R.C$ are also $\mathcal{ALC}$-Full concepts. An $\mathcal{ALC}$-Full TBox $KB_{\mathcal{T}}$ consists of a finite set of* concept inclusion axioms *of the form $C \sqsubseteq D$, where C and D are $\mathcal{ALC}$-Full concepts. An $\mathcal{ALC}$-Full ABox $KB_{\mathcal{A}}$ consists of a finite set of concept and role membership axioms, $C(a)$ and $R(a, b)$, respectively, and individual (in)equality axioms $a \circ b$ with $\circ \in \{\approx, \not\approx\}$, where C is an $\mathcal{ALC}$-Full concept and $R, a, b \in N$. An $\mathcal{ALC}$-Full knowledge base KB consists of a TBox $KB_{\mathcal{T}}$ and an ABox $KB_{\mathcal{A}}$.*

Let μ and ξ be operators which, when applied to an $\mathcal{ALC}$-Full concept, produce a set of triples and a symbol, respectively, defined as follows, where x is a new symbol unique for each application of μ:

D	$\mu(D)$	$\xi(D)$
A	$\emptyset$	A
$\neg C$	$\{\langle x, \mathit{owl{:}complementOf}, \xi(C)\rangle\} \cup \mu(C)$	x
$C_1 \sqcup C_2$	$\{\langle x, \mathit{owl{:}unionOf}_1, \xi(C_1)\rangle, \langle x, \mathit{owl{:}unionOf}_2, \xi(C_2)\rangle\} \cup \mu(C_1) \cup \mu(C_2)$	x
$C_1 \sqcap C_2$	$\{\langle x, \mathit{owl{:}intersectionOf}_1, \xi(C_1)\rangle, \langle x, \mathit{owl{:}intersectionOf}_2, \xi(C_2)\rangle\}$ $\cup\, \mu(C_1) \cup \mu(C_2)$	x
$\exists R.C$	$\{\langle x, \mathit{owl{:}onProperty}, R\rangle, \langle x, \mathit{owl{:}someValuesFrom}, \xi(C)\rangle\} \cup \mu(C)$	x
$\forall R.C$	$\{\langle x, \mathit{owl{:}onProperty}, R\rangle, \langle x, \mathit{owl{:}allValuesFrom}, \xi(C)\rangle\} \cup \mu(C)$	x

We extend μ to axioms and an $\mathcal{ALC}$-Full knowledge base KB in the following way:

$$\begin{aligned}
\mu(C \sqsubseteq D) &= \{\langle \xi(C), \mathit{rdfs{:}subClassOf}, \xi(D)\rangle\} \cup \mu(C) \cup \mu(D) \\
\mu(C(a)) &= \{\langle a, \mathit{rdf{:}type}, \xi(C)\rangle\} \cup \mu(C) \\
\mu(R(a, b)) &= \{\langle a, R, b\rangle\} \\
\mu(a \approx b) &= \{\langle a, \mathit{owl{:}sameAs}, b\rangle\} \\
\mu(a \not\approx b) &= \{\langle a, \mathit{owl{:}differentFrom}, b\rangle\} \\
\mu(KB) &= \textstyle\bigcup_{\alpha \in KB} \mu(\alpha)
\end{aligned}$$

Let N_{KB} denote the set of objects occurring in triples of $\mu(KB)$. An interpretation *I of KB is a structure $(\triangle^I, \cdot^I, EXT^I)$, where $\triangle^I$ is a set called the* interpretation domain, *$\cdot^I : N_{KB} \to \triangle^I$ is a* name interpretation function *and $EXT^I : \triangle^I \to 2^{\triangle^I \times \triangle^I}$ is an* extension function. *Let $CEXT^I : \triangle^I \to 2^{\triangle^I}$ be the* concept extension function *defined as*

$$CEXT^I(x) = \{y \mid (y, x) \in EXT^I(\mathit{rdf{:}type}^I)\}$$

An interpretation I is a model *of KB if, for every $\langle s, p, o\rangle \in \mu(KB)$, we have $(s^I, o^I) \in EXT^I(p^I)$, and the following conditions are satisfied:*

1. *If $(x, y) \in EXT^I(\mathit{owl{:}sameAs}^I)$, then $x = y$.*
2. *If $(x, y) \in EXT^I(\mathit{owl{:}differentFrom}^I)$, then $x \neq y$.*
3. *If $(x, y) \in EXT^I(\mathit{rdfs{:}subClassOf}^I)$, then $CEXT^I(x) \subseteq CEXT^I(y)$.*
4. *If $(x, y) \in EXT^I(\mathit{owl{:}complementOf}^I)$, then $CEXT^I(x) = \triangle^I \setminus CEXT^I(y)$.*

5. *If* $(x,u) \in EXT^I(owl{:}unionOf_1^I)$ *and* $(x,v) \in EXT^I(owl{:}unionOf_2^I))$, *then* $CEXT^I(x) = CEXT^I(u) \cup CEXT^I(v)$.

6. *If* $(x,u) \in EXT^I(owl{:}intersectionOf_1^I)$ *and* $(x,v) \in EXT^I(owl{:}intersectionOf_2^I)$, *then* $CEXT^I(x) = CEXT^I(u) \cap CEXT^I(v)$.

7. *If* $(x,y) \in EXT^I(owl{:}someValuesFrom^I)$ *and* $(x,p) \in EXT^I(owl{:}onProperty^I)$, *then* $CEXT^I(x) = \{u \mid (u,v) \in EXT^I(p) \wedge v \in CEXT^I(y)\}$.

8. *If* $(x,y) \in EXT^I(owl{:}allValuesFrom^I)$ *and* $(x,p) \in EXT^I(owl{:}onProperty^I)$, *then* $CEXT^I(x) = \{u \mid (u,v) \in EXT^I(p) \rightarrow v \in CEXT^I(y)\}$.

KB is satisfiable if and only if a model of KB exists. Checking KB satisfiability is the main inference for $\mathcal{ALC}$-Full.

The above definition differs from the one in [91] in that it does not provide for concrete predicates, the translation into triples does not include the meta-level resources such as *owl:Class*, and the syntax is limited to only binary union and intersection of classes. For the undecidability proof given later, these distinctions are not relevant.

We show now that, for *KB* an $\mathcal{ALC}$-Full knowledge base, deciding whether *KB* is satisfiable is undecidable. The proof is by reduction from a well-known *domino tiling* problem [19]. A *domino system* is a triple $\mathcal{D} = (D, H, V)$, where $D = \{D_1, \ldots, D_n\}$ is a finite set of *domino types*, and $H \subseteq D \times D$ and $V \subseteq D \times D$ are *horizontal* and *vertical compatibility relations*, respectively. A $\mathcal{D}$-*tiling* of an infinite grid is a function $t : \mathbb{N} \times \mathbb{N} \rightarrow D$ such that $t(0,0) = D_0$ and, for all $i, j \in \mathbb{N}$, $(f(i,j), f(i,j+1)) \in H$ and $(f(i,j), f(i+1,j)) \in V$. For an arbitrary domino system $\mathcal{D}$, determining whether a $\mathcal{D}$-tiling exists is undecidable [19].

For a domino system $\mathcal{D}$, let $KB_\mathcal{D}$ be the following $\mathcal{ALC}$-Full knowledge base:

$$D_i \sqcap D_j \sqsubseteq \bot \text{ for } 1 \leq i < j \leq n \tag{9.1}$$

$$GRID \sqsubseteq D_1 \sqcap \ldots \sqcap D_n \tag{9.2}$$

$$D_i \sqsubseteq \forall owl{:}allValuesFrom.\bigsqcup_{(D_i,d)\in H} d \tag{9.3}$$

$$D_i \sqsubseteq \forall rdf{:}type.\bigsqcup_{(D_i,d)\in V} d \tag{9.4}$$

$$GRID \sqsubseteq \exists owl{:}allValuesFrom.GRID \tag{9.5}$$

$$GRID \sqsubseteq \exists rdf{:}type.GRID \tag{9.6}$$

$$owl{:}allValuesFrom \approx GRID \tag{9.7}$$

$$rdf{:}type \approx owl{:}onProperty \tag{9.8}$$

$$(GRID \sqcap D_0)(a) \tag{9.9}$$

We show next that $KB_\mathcal{D}$ exactly encodes the domino tiling problem.

Lemma 9.1.2. *For a domino system $\mathcal{D}$, a $\mathcal{D}$-tiling exists if and only if $KB_\mathcal{D}$ is satisfiable.*

Proof. The ($\Rightarrow$) direction is trivial, since each $\mathcal{D}$-tiling uniquely defines a model of $KB_{\mathcal{D}}$, where horizontal links are represented by the *owl:allValuesFrom* relation, and vertical links are represented by the *rdf:type* relation.

For the ($\Leftarrow$) direction, assume that $KB_{\mathcal{D}}$ has a model I. An excerpt of I is shown in Figure 9.1, where a triple $\langle s, p, o\rangle$ is represented as an arc with label p, pointing from the node s to the node o. To easily refer to arcs, we assign them names t_i, h_i and v_i. These names do not represent the arc labels; rather, the arc label is encoded using the legend at the bottom of the figure. For example, the arc h_1 represents the triple $\langle a, \mathit{owl{:}allValuesFrom}, b\rangle$. Due to axiom (9.7), symbols *owl:allValuesFrom* and *GRID* are synonyms, so the arc h_1 also represents the triple $\langle a, \mathit{GRID}, b\rangle$.

The central node in the model corresponds to the symbol *GRID*. Due to axiom (9.9), the node a is linked by t_1 to *GRID*. Because of that and due to axioms (9.6) and (9.7), a is linked to b and c through h_1 and v_1, respectively, and b and c are also members of the *GRID* concept, i.e. arcs t_2 and t_3 exist. Finally, due to axiom (9.5), c is linked by h_2 to d, which is a member of the *GRID* concept by t_4.

Consider now the situation at node c. The central node *GRID* can, due to axiom (9.9), be read as *owl:allValuesFrom*. Hence, arc t_3 can, due to axiom (9.8), be read as $\langle c, \mathit{owl{:}onProperty}, \mathit{GRID}\rangle$. By the item 8 of Definition 9.1.1, arcs t_3 and h_2 make the node c correspond to the concept $\forall \mathit{owl{:}allValuesFrom}.d$. Since a is a member of c due to arc v_1, and it is linked to b through arc h_1 which is labeled with *owl:allValuesFrom*, this implies that b is a member of d, i.e. that b is connected to d through arc v_2. Hence, a, b, c and d are arranged in a two-dimensional grid, which continues indefinitely due to axioms (9.5) and (9.6).

A node x in I is allowed to have multiple *owl:allValuesFrom* and *rdf:type* successors, so I need not be a two-dimensional grid. However, a two-dimensional grid may easily be extracted from I. Namely, one can arbitrarily choose any *owl:allValuesFrom* successor x_1 of x, any *rdf:type* successor x_2 of x, and any *owl:allValuesFrom* successor x_3 of x_2. Regardless of the choices, x_3 is always connected to x_2 by *rdf:type*, so x, x_1, x_2 and x_3 are connected in a grid-like manner.

Hence, each interpretation I of $KB_{\mathcal{D}}$ contains a two-dimensional infinite grid where *owl:allValuesFrom* are horizontal, and *rdf:type* are vertical arcs. Axioms (9.1) – (9.4) obviously correspond to compatibility relations H and V of $\mathcal{D}$ so it is possible to construct a $\mathcal{D}$-tiling from I. □

Undecidability of $\mathcal{ALC}$-Full follows as an immediate consequence of Lemma 9.1.2 and the known undecidability result for the domino tiling problem [19]:

Theorem 9.1.3. *Checking whether an $\mathcal{ALC}$-Full knowledge base KB is satisfiable is undecidable.*

9.2 Extending DLs with Decidable Metamodeling

The reduction used to prove Theorem 9.1.3 shows that undecidability arises due to metamodeling features of $\mathcal{ALC}$-Full. Hence, adding OWL-Full-style metamodeling to

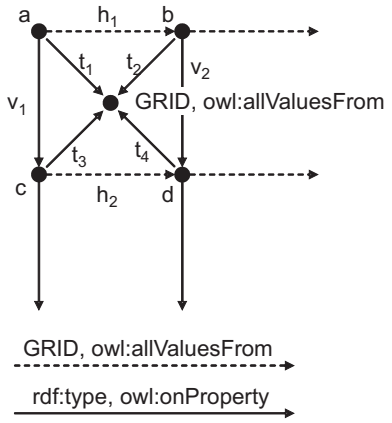

Figure 9.1: Grid Structure in a Model of $KB_{\mathcal{D}}$

OWL-DL also results in an undecidable logic. It is a natural question to ask whether metamodeling can be added to OWL-DL without losing decidability. In this section, we show that this is indeed possible.

Notice that in the reduction in Lemma 9.1.2 we use *owl:allValuesFrom* and *rdf:type* in role restrictions. This allowed us to change the default meaning of the modeling primitives, and thus create a knowledge base capable of encoding a two-dimensional grid. The approach we present in this section differs in an important aspect that, while using concepts as individuals is allowed, using modeling primitives in the knowledge base is not. Such a semantics is inspired by HiLog [30] — a logic with the goal of providing a second-order modeling flavor without leaving the confines of first-order logic. Due to certain technical problems which we discuss later, we first extend $\mathcal{ALCHIQ}(\mathbf{D})$ with metamodeling, and consider transitive roles subsequently.

9.2.1 Metamodeling Semantics for $\mathcal{ALCHIQ}(\mathbf{D})$

In the rest, to facilitate metamodeling at the syntactic level, we adapt the syntax of description logic to allow $N_C = N_I = N_{R_a} = N_{R_c} = N$. Hence, the same symbol can be used to denote a concept, an individual, an abstract and a concrete role. Due to technical complications, we assume that the set of concrete predicates $\Phi_{\mathbf{D}}$ is disjoint from N. We do not believe that this poses any real restrictions in practice, i.e. we cannot think of any example where treating concrete predicates as individuals might be required.

We now define the so-called *contextual* semantics of metamodeling, which got its name due to the fact that a symbol in the knowledge is interpreted as a concept, role or individual due to syntactic context where it occurs.

Definition 9.2.1 (Contextual Semantics)**.** *Let KB be an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base over an admissible concrete domain $\mathbf{D}$. A π-interpretation $I = (\triangle^I, \cdot^I, C^I, R_a^I, R_c^I)$ is*

a 5-tuple where $\triangle^I$ *is a* domain set, $\cdot^I : N \to \triangle^I$ *is a* symbol interpretation function, $C^I : N \to 2^{\triangle^I}$ *is an* atomic concept extension function, $R_a^I : N \to 2^{\triangle^I \times \triangle^I}$ *is an* abstract role extension function *and* $R_c^I : N \to 2^{\triangle^I \times \triangle_{\mathbf{D}}}$ *is a* concrete role extension function.

The function C^I *is extended to concepts as specified in tables 2.2 and 2.4, where symbols are interpreted* contextually, *i.e.* $\cdot^I$ *(*C^I*,* R_a^I *and* R_c^I*) is used for symbols occurring in axioms as individuals (concepts, abstract roles and concrete roles, respectively). For example,* C^I *is extended to* $\exists R.C$ *in the following way:*

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

A π*-interpretation* I *is a* π*-model of* KB *if it satisfies all axioms conditions tables 2.2 and 2.4. The notions of* π*-satisfiability,* π*-unsatisfiability and* π*-entailment (written* $\models_\pi$*) are defined as usual.*

It is easy to see that contextual semantics is actually equivalent to standard first-order semantics. Namely, first-order logic does not require the set of predicate symbols and function symbols to be disjoint; rather, the interpretation of symbols depends on the place of their occurrence. For example, in a first-order formula $C(C)$, the outer C is clearly a predicate symbol, and the inner C is clearly a constant. Hence, KB is π-satisfiable if and only if $\pi(KB)$ is (first-order) satisfiable. Notice that, as mentioned in Section 2.4, in basic superposition we encode predicate symbols as function symbols. For a correct encoding of contextual syntax, we simply use a separate sort for function symbols obtained by encoding predicate symbols.

We now define the *HiLog* semantic for metamodeling which is more in the spirit of OWL-Full.

Definition 9.2.2 (HiLog Semantics)**.** *Let* KB *be an* $\mathcal{ALCHIQ}(\mathbf{D})$ *knowledge base over an admissible concrete domain* $\mathbf{D}$*. A* ν*-interpretation* $I = (\triangle^I, \cdot^I, C^I, R_a^I, R_c^I)$ *is a 5-tuple where* $\triangle^I$ *is a* domain set, $\cdot^I : N \to \triangle^I$ *is a* symbol interpretation function, $C^I : \triangle^I \to 2^{\triangle^I}$ *is an* atomic concept extension function, $R_a^I : \triangle^I \to 2^{\triangle^I \times \triangle^I}$ *is an* abstract role extension function *and* $R_c^I : \triangle^I \to 2^{\triangle^I \times \triangle_{\mathbf{D}}}$ *is a* concrete role extension function.

The function C^I *is extended to concepts as specified in Table 9.1, upper part, where* A *is an atomic concept symbol,* C *and* D *are concepts,* R *and* S *are abstract roles,* $T_{(i)}$ *and* U *are concrete roles,* n *an integer and* d *a concrete predicate.* I *is a* ν*-model of* KB *if each axiom in* KB *is satisfied as specified in Table 9.1, lower part. The notions of* ν*-satisfiability,* ν*-unsatisfiability and* ν*-entailment (written* $\models_\nu$*) are defined as usual.*

We briefly discuss the essential difference between these two semantics. Consider the knowledge base consisting only of one axiom $C(C)$, where the symbol C is used as an individual and as a concept. The π-model of such a knowledge base is depicted on the left hand side of Figure 9.2, where both the individual interpretation $\cdot^I$ and the concept interpretation C^I are assigned directly to the symbol C. On the contrary,

Table 9.1: Direct Model-theoretic Semantics for Metamodeling

Interpreting Concepts		
$C^I(A)$	$=$	$C^I(A^I)$
$C^I(\neg C)$	$=$	$\triangle^I \setminus C^I(C)$
$C^I(C \sqcap D)$	$=$	$C^I(C) \cap C^I(D)$
$C^I(C \sqcup D)$	$=$	$C^I(C) \cup C^I(D)$
$C^I(\forall R.C)$	$=$	$\{x \mid \forall y : (x,y) \in R_a^I(R^I) \rightarrow y \in C^I(C)\}$
$C^I(\exists R.C)$	$=$	$\{x \mid \exists y : (x,y) \in R_a^I(R^I) \wedge y \in C^I(C)\}$
$C^I(\leq n\, R.C)$	$=$	$\{x \mid \sharp\{y \mid (x,y) \in R_a^I(R^I) \wedge y \in C^I(C)\} \leq n\}$
$C^I(\geq n\, R.C)$	$=$	$\{x \mid \sharp\{y \mid (x,y) \in R_a^I(R^I) \wedge y \in C^I(C)\} \geq n\}$
$C^I(\forall T_1, \ldots, T_m.d)$	$=$	$\{x \mid \forall y_1, \ldots, y_m : (x,y_1) \in R_c^I(T_1^I) \wedge \ldots \wedge (x,y_m) \in R_c^I(T_m^I) \rightarrow (y_1, \ldots, y_m) \in d^{\mathbf{D}}\}$
$(\exists T_1, \ldots, T_m.d)^I$	$=$	$\{x \mid \exists y_1, \ldots, y_m : (x,y_1) \in R_c^I(T_1^I) \wedge \ldots \wedge (x,y_m) \in R_c^I(T_m^I) \wedge (y_1, \ldots, y_m) \in d^{\mathbf{D}}\}$
$C^I(\leq n\, T)$	$=$	$\{x \mid \sharp\{y \mid (x,y) \in R_c^I(T^I)\} \leq n\}$
$C^I(\geq n\, T)$	$=$	$\{x \mid \sharp\{y \mid (x,y) \in R_c^I(T^I)\} \geq n\}$
Semantics of Axioms		
$C \sqsubseteq D$		$C^I(C) \subseteq C^I(D)$
$C \equiv D$		$C^I(C) = C^I(D)$
$R \sqsubseteq S$		$R_a^I(R^I) \subseteq R_a^I(S^I)$
$T \sqsubseteq U$		$R_c^I(S^I) \subseteq R_c^I(U^I)$
$(\neg)C(a)$		$a^I \in (\notin)\ C^I(C)$
$(\neg)R(a,b)$		$(a^I, b^I) \in (\notin)\ R_a^I(R^I)$
$(\neg)T(a,c)$		$(a^I, c^I) \in (\notin)\ R_c^I(T^I)$
$a \circ b$		$a^I \circ b^I$ for $\circ \in \{\approx, \not\approx\}$

a ν-model of the knowledge base is depicted on the right hand side of Figure 9.2. There, the domain object x is assigned by the individual interpretation $\cdot^I$ directly to the symbol C; however, the concept interpretation is not assigned to the symbol C, but to the individual x. We discuss the practical consequences of HiLog semantics on entailment in Section 9.3.

Since our algorithms are based on resolution calculi, we present an alternative equivalent definition of ν-models by translation into first-order logic. This translation follows the principles of transforming HiLog formulae into first-order formulae, i.e. it reifies concept, abstract and concrete role symbols into constants, and represents functions C^I, R_a^I and R_c^I explicitly by predicates isa, arole and crole.

Definition 9.2.3. *For an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base KB, let $\nu(KB)$ be the translation of KB into first-order formulae, as specified in Table 9.2, where* isa *is a binary predicate with signature* a × a, arole *is a ternary predicate with signature* a × a × a *and* crole *is a ternary predicate with signature* a × a × c.

Lemma 9.2.4. *For an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base KB, KB is ν-satisfiable if and only if a first-order model of $\nu(KB)$ exists.*

Table 9.2: Semantics of Metamodeling by Mapping into First-order Logic

Mapping Concepts to FOL		
$\nu_y(A, X)$	$=$	$\mathsf{isa}(A, X)$
$\nu_y(\neg C, X)$	$=$	$\neg\nu_y(C, X)$
$\nu_y(C \sqcap D, X)$	$=$	$\nu_y(C, X) \wedge \nu_y(D, X)$
$\nu_y(C \sqcup D, X)$	$=$	$\nu_y(C, X) \vee \nu_y(D, X)$
$\nu_y(\forall R.C, X)$	$=$	$\forall y : \mathsf{arole}(R, X, y) \rightarrow \nu_x(C, y)$
$\nu_y(\exists R.C, X)$	$=$	$\exists y : \mathsf{arole}(R, X, y) \wedge \nu_x(C, y)$
$\nu_y(\leq n\, R.C, X)$	$=$	$\forall y_1, \ldots, y_{n+1} : \bigwedge \mathsf{arole}(R, X, y_i) \wedge \bigwedge\ \nu_x(C, y_i) \rightarrow \bigvee\ y_i \approx y_j$
$\nu_y(\geq n\, R.C, X)$	$=$	$\exists y_1, \ldots, y_n : \bigwedge \mathsf{arole}(R, X, y_i) \wedge \bigwedge\ \nu_x(C, y_i) \wedge \bigwedge\ y_i \not\approx y_j$
$\nu_y(\forall T_1, \ldots, T_m.d, X)$	$=$	$\forall y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}} : \bigwedge \mathsf{crole}(T_i, X, y_i^{\mathsf{c}}) \rightarrow d(y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}})$
$\nu_y(\exists T_1, \ldots, T_m.d, X)$	$=$	$\exists y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}} : \bigwedge \mathsf{crole}(T_i, X, y_i^{\mathsf{c}}) \wedge d(y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}})$
$\nu_y(\leq n\, T, X)$	$=$	$\forall y_1^{\mathsf{c}}, \ldots, y_{n+1}^{\mathsf{c}} : \bigwedge \mathsf{crole}(T, X, y_i^{\mathsf{c}}) \rightarrow \bigvee\ y_i^{\mathsf{c}} \approx y_j^{\mathsf{c}}$
$\nu_y(\geq n\, T, X)$	$=$	$\exists y_1^{\mathsf{c}}, \ldots, y_n^{\mathsf{c}} : \bigwedge \mathsf{crole}(T, X, y_i^{\mathsf{c}}) \wedge \bigwedge\ y_i^{\mathsf{c}} \not\approx y_j^{\mathsf{c}}$
Mapping Axioms to FOL		
$\nu(C \sqsubseteq D)$	$=$	$\forall x : \nu_y(C, x) \rightarrow \nu_y(D, x)$
$\nu(C \equiv D)$	$=$	$\forall x : \nu_y(C, x) \leftrightarrow \nu_y(D, x)$
$\nu(R \sqsubseteq S)$	$=$	$\forall x, y : \mathsf{arole}(R, x, y) \rightarrow \mathsf{arole}(S, x, y)$
$\nu(T \sqsubseteq U)$	$=$	$\forall x, y : \mathsf{crole}(T, x, y) \rightarrow \mathsf{crole}(U, x, y)$
$\nu((\neg)C(a))$	$=$	$(\neg)\nu_y(C, a)$
$\nu((\neg)R(a, b))$	$=$	$(\neg)\mathsf{arole}(R, a, b)$
$\nu((\neg)T(a, b^{\mathsf{c}}))$	$=$	$(\neg)\mathsf{crole}(T, a, b^{\mathsf{c}})$
$\nu(a^{(\mathsf{c})} \circ b^{(\mathsf{c})})$	$=$	$a^{(\mathsf{c})} \circ b^{(\mathsf{c})}$ for $\circ \in \{\approx, \not\approx\}$
Mapping KB to FOL		
$\nu(R)$	$=$	$\forall x, y : \mathsf{arole}(R, x, y) \leftrightarrow \mathsf{arole}(R^-, y, x) \wedge$
		$\forall x, y : \mathsf{crole}(R, x, y) \leftrightarrow \mathsf{crole}(R^-, y, x)$
$\nu(KB_{\mathcal{R}})$	$=$	$\bigwedge_{\alpha \in KB_{\mathcal{R}}} \nu(\alpha) \wedge \bigwedge_{R \in N} \nu(R)$
$\nu(KB_{\mathcal{T}})$	$=$	$\bigwedge_{\alpha \in KB_{\mathcal{T}}} \nu(\alpha)$
$\nu(KB_{\mathcal{A}})$	$=$	$\bigwedge_{\alpha \in KB_{\mathcal{A}}} \nu(\alpha)$
$\nu(KB)$	$=$	$\nu(KB_{\mathcal{R}}) \wedge \nu(KB_{\mathcal{T}}) \wedge \nu(KB_{\mathcal{A}})$

Notes:
(i): X is a meta variable and is substituted by the actual variable,
(ii): ν_x is defined as ν_y by substituting x and x_i for all y and y_i, respectively.

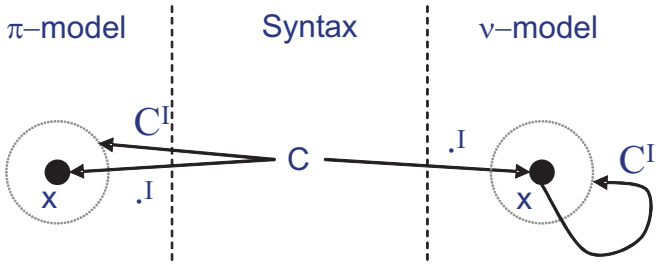

Figure 9.2: π- and ν-models of the Example Knowledge Base

Proof. For the ($\Rightarrow$) direction, let I_ν be a ν-model of KB. We construct a first-order interpretation I for $\nu(KB)$ by setting $\triangle^I = \triangle^{I_\nu}$, $a^I = a^{I_\nu}$, $(x, y) \in \mathsf{isa}^I$ if and only if $y \in C^{I_\nu}(x)$, $(x, y, z) \in \mathsf{arole}^I$ if and only if $(y, z) \in R_a^{I_\nu}(x)$, and $(x, y, z) \in \mathsf{crole}^I$ if and only if $(y, z) \in R_c^{I_\nu}(x)$. By induction on the structure of formulae in $\nu(KB)$, one can easily show that I is a model of $\nu(KB)$. The ($\Leftarrow$) direction is similar. □

HiLog semantics is compatible with OWL-Full if resources defining modeling primitives of OWL-Full are not used in knowledge base axioms. The formal proof for this is simple, but is lengthy due to the very complex semantics of OWL-Full, so we omit it for the sake of brevity.

9.2.2 Deciding ν-satisfiability

We now show that ν-satisfiability can be decided by extending the algorithms from chapters 3 and 4 without an increase in complexity. It is difficult to ensure termination of direct saturation of closures obtained by structural transformation of $\nu(KB)$, since such closures contain unmarked constants corresponding to names of concepts and roles. Consider, for example, the following set of closures obtained during saturation:

$$\mathsf{isa}(C, x) \tag{9.10}$$
$$\mathsf{isa}(D, x) \tag{9.11}$$
$$C \approx C' \vee D \approx D' \tag{9.12}$$

Superposition of (9.12) into (9.10) yields (9.13), which can be superposed into (9.11) to obtain (9.14). The problem with this closure is that it contains two variables. Further inferences with (9.14) might yield a closure with even more variables, thus preventing us from establishing the upper bound on the length of closures and invalidating the termination proof.

$$\mathsf{isa}(C', x) \vee D \approx D' \tag{9.13}$$
$$\mathsf{isa}(C', x) \vee \mathsf{isa}(D', x') \tag{9.14}$$

Another problem occurs due to decomposition. Namely, a superposition inference might derive a closure with a literal of the form $\mathsf{arole}(\langle R\rangle, [t], [f(t)])$. The problem is that R may, but need not be marked, thus giving two different definition predicates for a pair of R and f.

We solve these problems by modifying the preprocessing step. Firstly, a closure such as (9.10) is decomposed into closures $\neg\mathcal{O}_C(z) \vee \mathsf{isa}(z, x)$ and $\mathcal{O}_C(C)$. Next, the literal $\neg\mathcal{O}_C(z)$ is selected, so the first resolution inference involving that closure produces $\mathsf{isa}([C], x)$. Finally, we modify the calculus so that a ground closure of the form $\mathcal{O}_P(\langle Q\rangle) \vee C \cdot \rho$ is decomposed into $\neg q_{P,Q} \vee \mathcal{O}_P(Q)$ and $q_{P,Q} \vee C \cdot \rho$.

Definition 9.2.5. *For an $\mathcal{ALCHIQ}(\mathbf{D})$-closure C, let*

$$\mathsf{Cls}_\nu(\mathsf{Def}(C)) = \bigcup_{D \in \mathsf{Def}(C)} \mathsf{Cls}(\forall x : \nu_y(D, x))$$

Furthermore, let $\zeta(C)$ be the closure obtained from C by replacing all literals according to the following table, where the predicate $\mathcal{O}_P$ is globally unique for the predicate symbol P, z_P is a new variable globally unique for P, and u and v are arbitrary terms:

$$\begin{array}{lcl} \mathsf{isa}(P, u) & \rightsquigarrow & \mathsf{isa}(z_P, u) \vee \neg\mathcal{O}_P(z_P) \\ \mathsf{arole}(P, u, v) & \rightsquigarrow & \mathsf{arole}(z_P, u, v) \vee \neg\mathcal{O}_P(z_P) \\ \mathsf{crole}(P, u, v^{\mathsf{c}}) & \rightsquigarrow & \mathsf{crole}(z_P, u, v^{\mathsf{c}}) \vee \neg\mathcal{O}_P(z_P) \end{array}$$

The operator ζ is extended to a set of closures by applying it to each member of the set. For an $\mathcal{ALCHIQ}$ knowledge base KB, let $\Xi_\nu(KB)$ denote the smallest set of closures satisfying the following conditions:

- *For each name $R \in N$, $\zeta(\mathsf{Cls}(\nu(R))) \subseteq \Xi_\nu(KB)$.*
- *For each RBox or ABox axiom α in KB, $\zeta(\mathsf{Cls}(\nu(\alpha))) \subseteq \Xi_\nu(KB)$.*
- *For each TBox axiom $C \sqsubseteq D$ in KB, $\zeta(\mathsf{Cls}_\nu(\mathsf{Def}(\neg C \sqcup D))) \subseteq \Xi_\nu(KB)$.*
- *For each TBox axiom $C \equiv D$ in KB, $\zeta(\mathsf{Cls}_\nu(\mathsf{Def}(\neg C \sqcup D))) \subseteq \Xi_\nu(KB)$ and $\zeta(\mathsf{Cls}_\nu(\mathsf{Def}(\neg D \sqcup C))) \subseteq \Xi_\nu(KB)$.*
- *For each predicate $\mathcal{O}_P$ introduced by ζ, $\mathcal{O}_P(P) \in \Xi_\nu(KB)$.*

It is easy to see that $\Xi_\nu(KB)$ is satisfiable if and only if KB is ν-satisfiable.

Lemma 9.2.6. *Let KB be an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base. Then KB is ν-satisfiable if and only if $\Xi_\nu(KB)$ is satisfiable. Furthermore, $\Xi_\nu(KB)$ can be computed in time polynomial in $|KB|$ for unary coding of numbers in input.*

Proof. By Lemma 9.2.4, KB is ν-satisfiable if and only if $\nu(KB)$ is satisfiable. $\Xi_\nu(KB)$ is obtained from $\nu(KB)$ by (i) applying structural transformation which is known not to affect satisfiability, and by (ii) applying decomposition to literals $\mathsf{isa}(P, u)$, $\mathsf{arole}(P, u, v)$ and $\mathsf{crole}(P, u, v^{\mathsf{c}})$, which is shown not to affect satisfiability by Theorem 3.4.4. □

Definition 9.2.7. *Let $\mathcal{BS}_{\nu}^{\mathbf{D},+}$ be the $\mathcal{BS}^{\mathbf{D}}$ calculus parameterized as follows:*

- *The term ordering $\succ$ is a lexicographic path ordering induced over a total precedence $>_P$ over function, constant and predicate symbols, such that, for any function symbol f, constant symbol c, predicate symbol p, and a propositional symbol $q_{P,Q}$, we have $f >_P c >_P p >_P q_{P,Q} >_P \top$.*
- *If a closure contains a literal of the form $\neg\mathcal{O}_C(z)$, then all such literals are selected; otherwise, all negative* arole *and* crole *literals are selected.*
- *Inference conclusions, whenever possible, are decomposed according to the following table, for an arbitrary term t:*

$$D \cdot \rho \vee \mathsf{arole}([R], [t], [f(t)]) \rightsquigarrow \begin{array}{c} D \cdot \rho \vee Q_{R,f}([t]) \\ \neg Q_{R,f}(x) \vee \mathsf{arole}([R], x, [f(x)]) \end{array}$$

$$D \cdot \rho \vee \mathsf{arole}([R], [f(x)], x) \rightsquigarrow \begin{array}{c} D \cdot \rho \vee Q_{\mathsf{Inv}(R),f}(x) \\ \neg Q_{\mathsf{Inv}(R),f}(x) \vee \mathsf{arole}([R], [f(x)], x) \end{array}$$

$$D \cdot \rho \vee \mathsf{crole}([T], [t], [f^{\mathsf{c}}(t)]) \rightsquigarrow \begin{array}{c} D \cdot \rho \vee Q_{T,f^{\mathsf{c}}}([t]) \\ \neg Q_{T,f^{\mathsf{c}}}(x) \vee \mathsf{crole}([T], x, [f^{\mathsf{c}}(x)]) \end{array}$$

$$\mathcal{O}_P(\langle Q \rangle) \vee C \cdot \rho \rightsquigarrow \begin{array}{c} C \cdot \rho \vee q_{P,Q} \\ \neg q_{P,Q} \vee \mathcal{O}_P(Q) \end{array}$$

We extend the notion of $\mathcal{ALCHIQ}(\mathbf{D})$-closures defined in Section 3.4 to so-called ν-$\mathcal{ALCHIQ}(\mathbf{D})$-closures, which have the same form as in Table 3.1 and Table 4.1, without conditions (*iii*) – (*vi*), with the following differences:

- Atoms of the form $C(t)$ take the form $\mathsf{isa}([C], t)$.
- Atoms of the form $R(u, v)$ take the form $\mathsf{arole}([R], u, v)$.
- Atoms of the form $T(u, v^{\mathsf{c}})$ take the form $\mathsf{crole}([T], u, v^{\mathsf{c}})$.
- All closures may contain a disjunction of the form $\mathbf{q} = \bigvee(\neg)q_{P,Q}$.

Notice that closures in $\Xi_{\nu}(KB)$ are technically not ν-$\mathcal{ALCHIQ}(\mathbf{D})$-closures: instead of literals of the form $(\neg)\mathsf{isa}([C], x)$, they contain $(\neg)\mathsf{isa}(z, x) \vee \neg\mathcal{O}_C(z)$. We call such closures *initial closures*.

Lemma 9.2.8. *Let $\Xi_{\nu}(KB) = N_0, \ldots, N_i \cup \{C\}$ be a $\mathcal{BS}_{\nu}^{\mathbf{D},+}$-derivation, where C is the conclusion derived from premises in N_i. Then C is either a ν-$\mathcal{ALCHIQ}(\mathbf{D})$-closure or it is redundant in N_i.*

Proof. Due to decomposition, it is obvious that the following property (*) holds: positive literals of the form $\mathcal{O}_P(Q)$ occur only as unary closures, or in closures of the form $\mathcal{O}_P(Q) \vee \neg q_{P,Q}$.

Now consider an inference with an initial closure C. Since C contains literals of the form $\neg\mathcal{O}_P(z)$, and all such literals are selected, C can participate only in a hyperresolution inference on all $\neg\mathcal{O}_P(z_i)$. Because of (*), the inference instantiates all variables z_i, and the conclusion is a ν-$\mathcal{ALCHIQ}(\mathbf{D})$-closure.

Due to term ordering of $\mathcal{BS}_\nu^{\mathbf{D},+}$, only a literal with a functional term of the maximum depth can be maximal in a ν-$\mathcal{ALCHIQ}(\mathbf{D})$-closure. Also, the term ordering ensures that a propositional letter $q_{P,Q}$ is not maximal if a closure contains some other literal. Hence, the claim of this lemma can be shown in the same way as in the proofs of Lemma 3.3.5, Theorem 3.4.6 and Lemma 4.2.1. □

Theorem 9.2.9. *Let KB be an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base, defined over an admissible concrete domain $\mathbf{D}$, for which $\mathbf{D}$-satisfiability of finite conjunctions over $\Phi_{\mathbf{D}}$ can be decided in deterministic exponential time. Then saturation of $\Xi_\nu(KB)$ by $\mathcal{BS}_\nu^{\mathbf{D},+}$ with eager application of redundancy elimination rules decides ν-satisfiability of KB and runs in time exponential in $|KB|$, for unary coding of numbers and assuming a bound on the arity of concrete predicates.*

Proof. The number of propositional letters $q_{P,Q}$ is quadratic in $|KB|$, and the number of predicates $\mathcal{O}_P$ is linear in $|KB|$. Therefore, it is possible to obtain an exponential bound on the number of closures derived in the same way as in Lemma 3.3.8, Theorem 3.4.6 and Theorem 4.2.4. Since decomposition is sound and complete by Theorem 3.4.4, the claim of this theorem follows. □

9.2.3 Metamodeling and Transitivity

Since the decision procedure for checking ν-satisfiability of an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base KB does not differ essentially from a decision procedure for checking satisfiability of KB, one might intuitively expect that allowing transitivity axioms in KB does not affect termination of the algorithm. Unfortunately, this is not the case: for a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB, checking ν-satisfiability of KB is undecidable.

The reason for this dramatic change is that metamodeling allows referring to the same role by different names. This, in turn, makes it impossible to define simple roles by syntactic means. Consider the following knowledge base KB:

$$\top \sqsubseteq \geq 3\, R \tag{9.15}$$

$$R \approx S \tag{9.16}$$

$$\mathsf{Trans}(S) \tag{9.17}$$

Notice that KB is a $\mathcal{SHIQ}$ knowledge base: the role R is simple, since it passes the syntactic criterion specified in Definition 2.7.1 (i.e. it is neither transitive nor it has transitive subroles). However, in any ν-interpretation I, axiom (9.16) ensures that

R and S are interpreted as the same domain element α. Due to axiom (9.17), $R_a^I(\alpha)$ is transitive. Hence, a transitive role is used in a number restriction in axiom (9.15), although, syntactically, R is a simple role.

Extending the definition of simple roles might be difficult in general, since equality of role names might be entailed by other facts in KB. Since in [61] it was shown that using transitive roles in number restrictions leads to undecidability in general, we have the following result:

Corollary 9.2.10. *For a $\mathcal{SHIQ}$ knowledge base KB, deciding ν-satisfiability is undecidable.*

Decidability can be regained by using *unique role assumption*. Intuitively, this assumption requires two distinct role symbols to be interpreted as distinct domain individuals. In such a case, simple roles can be defined and checked as usual.

Definition 9.2.11 (Unique Role Assumption)**.** *A $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB is said to employ the* unique role assumption *if it contains an axiom $R \not\approx S$ for each two distinct symbols R and S occurring as roles in KB.*

It is easy that to see that transitivity axioms can be eliminated from knowledge bases employing unique roles assumption using the transformation from Section 3.2, so we thus obtain an algorithm for checking ν-satisfiability of such knowledge bases.

Lemma 9.2.12. *For a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB employing the unique role assumption, $\Omega(KB)$ is ν-satisfiable if and only if KB is ν-satisfiable.*

Proof. Due to unique role assumption, in each ν-model I_ν of KB, $R^{I_\nu} \neq S^{I_\nu}$ for each two distinct symbols R and S occurring as roles. Due to this fact, the proof of Theorem 3.2.3 can be adapted with minor differences. □

9.3 Added Expressivity of Metamodeling

In this section we investigate how metamodeling increases the expressivity of the logic. Here we consider only the logical aspects, namely, what kind of new consequences can be drawn. Our discussion in this section does not consider non-logical aspects which may be relevant in practice (such as e.g. increased search flexibility). The following results are similar to the ones for HiLog in [30].

It is easy to see that ν-satisfiability is a strictly stronger notion than π-satisfiability. Consider the following knowledge base KB:

$$C(a) \tag{9.18}$$

$$\neg D(a) \tag{9.19}$$

$$C \approx D \tag{9.20}$$

Under the contextual semantics, the interpretations of symbols C and D as concepts and as individuals are completely independent, so KB is π-satisfiable. However, KB

is ν-unsatisfiable: in each ν-interpretation we have $C^I = D^I = \alpha$, so it cannot be that $a^I \in C^I(\alpha)$ and $a^I \notin C^I(\alpha)$. For the other direction, we have the following simple lemma:

Lemma 9.3.1. *Each ν-satisfiable $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base is π-satisfiable.*

Proof. Let I_ν be a ν-model of an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base KB. We construct a π-interpretation I_π in the following way: we set $\triangle^{I_\pi} = \triangle^{I_\nu}$ and we set $x^{I_\pi} = x^{I_\nu}$, $C^{I_\pi}(x) = C^{I_\nu}(x^{I_\nu})$, $R_a^{I_\pi}(x) = R_a^{I_\nu}(x^{I_\nu})$, and $R_c^{I_\pi}(x) = R_c^{I_\nu}(x^{I_\nu})$, where $x \in N$. It is obvious that I_π is a π-model of KB. □

Furthermore, for a knowledge base with unique names assumption or without equality (either explicit or implicit, introduced through number restrictions), π-satisfiability and ν-satisfiability coincide:

Lemma 9.3.2. *Let KB be an $\mathcal{ALCHIQ}(\mathbf{D})$ knowledge base such that it employs unique names assumption, or it contains neither explicit equality statements nor number restrictions. Then KB is π-satisfiable if and only if it is ν-satisfiable.*

Proof. The ($\Leftarrow$) direction follows from Lemma 9.3.1. For the ($\Rightarrow$) direction, let KB be π-satisfiable in some model I_π. Due to the lemma assumptions, without loss of generality we may assume that for $a, b \in N$, $a \neq b$ implies $a^{I_\pi} \neq b^{I_\pi}$ (in the first case, this is because KB employs unique names assumption, and in the second case this is because KB does not employ equality).

Given such a π-model I_π, we construct a ν-interpretation I_ν by setting $\triangle^{I_\nu} = \triangle^{I_\pi}$, $a^{I_\nu} = a^{I_\pi}$, $C^{I_\nu}(a^{I_\nu}) = C^{I_\pi}(a)$, $R_a^{I_\nu}(a^{I_\nu}) = R_a^{I_\pi}(a)$, and $R_c^{I_\nu}(a^{I_\nu}) = R_c^{I_\pi}(a)$, where $a \in N$. Furthermore, for all $\alpha \in \triangle^{I_\nu}$ for which there is no $a \in N$ such that $\alpha = a^{I_\nu}$, let $C^{I_\nu}(\alpha) = R_a^{I_\nu}(\alpha) = R_c^{I_\nu}(\alpha) = \emptyset$. Since we can assume that different names from N are interpreted as different elements from $\triangle^{I_\nu}$, such a construction defines exactly one value of $C^{I_\nu}(\alpha)$, $R_a^{I_\nu}(\alpha)$ and $R_c^{I_\nu}(\alpha)$ for each $\alpha \in \triangle^{I_\nu}$, so I_ν is correctly defined. Furthermore, it is obvious that I_ν is a ν-model of KB. □

To summarize, metamodeling produces new consequences only if it is possible to derive that two symbols are equal, such as in the example from the beginning of this section. It is unclear whether this minor increase in expressivity is relevant in practice. Hence, it is justified to wonder whether metamodeling at the logical level is worth the trouble, in particular by taking into account potential problems with transitive roles and a more complicated decision procedure. We conjecture that requirements of most applications can be fulfilled by simply allowing the set of atomic concept names, individual names, abstract and simple roles to coincide, and to interpret them contextually, without any support for metamodeling in logic.

However, logical support for metamodeling becomes more interesting if it is combined with non-monotonic features, such as default rules [98] or autoepistemic description logics [36]. However, non-monotonic reasoning currently our of our scope.

9.4 Related Work

The definition of ν-satisfiability given in Section 9.2 is inspired by HiLog, a logic attempting to simulate second-order logic in a first-order framework [30]. HiLog generalizes the syntax of first-order logic by allowing general terms to occur in place of function and predicate symbols. The semantics of HiLog is defined by interpreting each individual as a member of the interpretation domain, and by assigning a functional and a relational interpretation to domain elements. In [30], it was shown that HiLog can be considered "syntactic sugar", since each HiLog formula can be encoded into an equisatisfiable first-order formula by reification. Our Definition 9.2.3 closely resembles to this encoding. Finally, it was shown that a satisfiable first-order formula without equality is also satisfiable under HiLog semantics.

In [87], the RDFS Model Theory was criticized for allowing infinite number of meta-layers. The authors argue that such a non-standard metamodeling semantics is inadequate for the Semantic Web since (*i*) it does not provide adequate support for inferencing, (*ii*) it allows defining classes which contain themselves, thus leading potentially to paradoxes, and (*iii*) by adding classes, one necessarily introduces objects in the interpretation universe. As a reaction to these deficiencies, the authors propose RDFS-FA, a stratified four-level semantics, consisting of the meta-language layer, the language layer, the ontology layer and the instance layer. Each of these layers is considered an instance of the layer above. Since we do not allow redefining modeling primitives in knowledge bases, our proposal follows the principles of RDFS-FA, with the difference that the ontology and the instance layer are combined.

To the best of our knowledge, we are unaware of any work which considered decidability of metamodeling in OWL-Full or which proposed a decidable description logic with metamodeling features.

Chapter 10

Conclusion

In this work we have developed several novel algorithms for reasoning with description logics related to $\mathcal{SHIQ}$. Our work is primarily motivated by the prospects of reusing well-known deductive database techniques to optimize query answering in description logics. However, techniques needed to achieve this goal are themselves new results.

In our approach we focus on the $\mathcal{SHIQ}(\mathbf{D})$ description logic. This logic is closely related to the ontology language for the Semantic Web OWL-DL. The logical underpinning of OWL-DL is actually the $\mathcal{SHOIN}(\mathbf{D})$ description logic, which differs from $\mathcal{SHIQ}(\mathbf{D})$ only in that $\mathcal{SHOIN}(\mathbf{D})$ supports nominals, but it does not provide for qualified number restrictions.

The first algorithm we present is an algorithm for checking satisfiability of $\mathcal{SHIQ}$ knowledge bases. We derived the procedure using basic superposition. The novel aspect is that the procedure allows clauses to contain functional terms of depth two, and relies on basic superposition to block certain undesirable inferences. Furthermore, it relies on subsumption to restrict the term depth, and not just the clause length. Finally, our procedure runs in ExpTime in the size of the knowledge base for unary coding of numbers in the input, which makes it worst-case optimal for the logic considered. It is worth noting that the assumption on unary coding of numbers is standard in practical description logic reasoning systems. Basic superposition alone decides only a slightly weaker logic $\mathcal{SHIQ}^{-}$, so to handle $\mathcal{SHIQ}$, we extend basic superposition with the decomposition rule, for which we show soundness and completeness.

An important aspect of OWL-DL is that it provides constructs for representing concrete data, such as strings or integers. Generally, such capabilities are integrated into description logics by the so-called concrete domain approach. Until now, reasoning with a concrete domain has been studied predominantly in the context of tableaux and automata calculi. These existing approaches to reasoning with a concrete domain are not directly applicable to clausal saturation calculi, such as basic superposition. Therefore, we have devised algorithms for reasoning with a concrete domain in the resolution framework. This approach is general and is applicable to any clausal calculus whose completeness proof is based on the model generation method. Furthermore, we apply this extension and derive a decision procedure for $\mathcal{SHIQ}(\mathbf{D})$. For unary coding

of numbers, assuming an upper bound on the arity of concrete predicates and an exponential decision procedure for checking satisfiability of concrete predicates, we show that adding a concrete domain does not increase the complexity of reasoning, i.e. our algorithm still runs in worst-case exponential time.

We next apply the decision procedure for $\mathcal{SHIQ}(\mathbf{D})$ to obtain the algorithm for reducing a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base to a disjunctive datalog program. This reduction is possible because we can safely remove from the saturated set of clauses produced by our procedure all clauses containing functional terms of depth greater than one, and simulate the ground functional terms of depth one with fresh constants. We believe that our approach will enable efficient ABox reasoning primarily because reduction to disjunctive datalog allows us to use various optimizations, such as join order optimizations or the magic sets transformation. The latter has been show to dramatically improve the evaluation of disjunctive datalog programs, as it reduces the number of models of the disjunctive program.

Based on this reduction, we showing that data complexity of checking satisfiability for $\mathcal{SHIQ}(\mathbf{D})$ knowledge bases is NP-complete. This is a new and somewhat surprising result, since this is much better than the combined complexity. We also defined the Horn fragment of $\mathcal{SHIQ}(\mathbf{D})$, where capability for modeling disjunctive information is traded for polynomial data complexity. We believe that this fragment is very relevant for practice, since it is still expressive, but provides hope for a tractable implementation.

We extend our algorithms with a simple, but useful feature. Namely, we show that certain, so-called DL-safe disjunctive rules can freely be appended to the obtained disjunctive program. The DL-safe rules are characterized by the restriction that each variable occurring in a rule occurs in a non-DL-atom in the rule body. This restriction limits the applicability of rules only to individuals introduced explicitly in the ABox. Since the number of such individuals is finite, adding DL-safe rules to $\mathcal{SHIQ}(\mathbf{D})$ does not cause termination problems.

We show that basic superposition can be used to answer conjunctive queries over $\mathcal{SHIQ}(\mathbf{D})$ knowledge bases. We also show that conjunctive query containment can be reduced to query answering. Conjunctive queries provide an expressive query languages for description logic.

Finally, we have considered extending $\mathcal{SHIQ}(\mathbf{D})$ with metamodeling. We have shown that metamodeling, as employed in the Semantic Web standard OWL-Full, is undecidable. Therefore, we have proposed an alternative semantics for metamodeling, for which we give a decision procedure for $\mathcal{ALCHIQ}(\mathbf{D})$.

For our future work, we see three theoretical and one practical challenge. The theoretical challenges are: extending the logic with nominals, providing a decision procedure running in ExpTime regardless of the coding of numbers and providing a decision procedure capable of dealing with transitivity directly, without encoding transitivity axioms. From the practical point of view, we are currently implementing a new description logic inference system, for which we shall perform a detailed performance comparison.

Bibliography

[1] A. Abecker, D. Drollinger, and P. Hanschke. TAXON: A Concept Language with Concrete Domains. In H. Boley and M. M. Richter, editors, *Proc. of the Int. Workshop on Processing Declarative Knowledge (PDK'91)*, pages 411–413. Springer, Kaiserslautern, Germany, 1991.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] G. Alsaç and C. Baral. Reasoning in description logics using declarative logic programming. Technical report, Arizona State University, Arizona, USA, 2002. www.public.asu.edu/ cbaral/papers/descr-logic-aaai2.pdf.

[4] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27:217–274, 1998.

[5] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, January 2003.

[6] F. Baader and P. Hanschke. A Scheme for Integrating Concrete Domains into Concept Languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pages 452–457, Sydney, Australia, 1991.

[7] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[8] F. Baader and W. Snyder. Unification Theory. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 8, pages 445–532. Elsevier Science, 2001.

[9] M. Baaz, U. Egly, and A. Leitsch. Normal Form Transformations. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 5, pages 273–333. Elsevier Science, 2001.

[10] L. Bachmair and H. Ganzinger. Rewrite-based Equational Theorem Proving with Selection and Simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.

[11] L. Bachmair and H. Ganzinger. Equational Reasoning in Saturation-Based Theorem Proving. In W. Bibel and P. H. Schmidt, editors, *Automated Deduction: A Basis for Applications*, volume I, Foundations: Calculi and Methods, chapter 11. Kluwer Academic Publishers, Dordrecht, 1998.

[12] L. Bachmair and H. Ganzinger. Ordered Chaining Calculi for First-Order Theories of Transitive Relations. *Journal of the ACM*, 45(6):1007–1049, November 1998.

[13] L. Bachmair and H. Ganzinger. Strict Basic Superposition. In *Automated Deduction—CADE-15*, volume 1421 of *Lecture Notes in Computer Science*, pages 160–174, Lindau, Germany, July 1998. Springer.

[14] L. Bachmair and H. Ganzinger. Resolution Theorem Proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 2, pages 19–99. Elsevier Science, 2001.

[15] L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic Paramodulation. *Information and Computation*, 121(2):172–192, 1995.

[16] P. Baumgartner. An Ordered Theory Resolution Calculus. In A. Voronkov, editor, *Proc. of the 1st Int. Conf. on Logic Programming and Automated Reasoning (LPAR'92)*, volume 624, pages 119–130, St. Petersburg, Russia, July 1992. Springer.

[17] C. Beeri and R. Ramakrishnan. On the power of magic. In *Proc. of the Sixth ACM Symposium on Principles of Database Systems*, pages 269–293, San Diego, CA, March 1987.

[18] S. Bergamaschi, S. Castano, D. Beneventano, and M. Vincini. Semantic Integration of Heterogeneous Information Sources. *Special Issue on Intelligent Information Integration, Data and Knowledge Engineering*, 36(1):215–249, 2001.

[19] R. Berger. The undecidability of the dominoe problem. *Memoirs of the American Mathematical Society*, 66, 1966.

[20] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: a structural data model for objects. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 58–67, Portland, Oregon, United States, 1989. ACM Press.

[21] Alexander Borgida. On the Relative Expressiveness of Description Logics and Predicate Logics. *Artificial Intelligence*, 82(1-2):353–367, 1996.

[22] R. Boyer. *Locking: A Restriction of Resolution.* PhD thesis, University of Texas at Austin, Texas, USA, 1971.

[23] R. J. Brachman and J. G. Schmolze. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science*, 9(2):171–216, 1985.

[24] S. Brass and U. W. Lipeck. Generalized Bottom-Up Query Evaluation. In A. Pirotte, C. Delobel, and G. Gottlob, editors, *Advances in Database Technology - Proceedings EDBT'92*, pages 88–103, Berlin, 1992. Springer.

[25] C.-L. Chang and R. C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, Inc., 1997.

[26] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the Decidability of Query Containment under Constraints. In *Proc. of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 149–158. ACM Press, 1998.

[27] D. Calvanese, M. Lenzerini, and D. Nardi. Unifying Class-Based Representation Formalisms. *Journal of Artificial Intelligence Research*, 11:199–240, 1999.

[28] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 386–391. AAAI Press, 2000.

[29] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proc. of the 9th annual ACM Symposium on Theory of Computing*, pages 77–90, Boulder, Colorado, USA, 1977. ACM Press.

[30] W. Chen, M. Kifer, and D. S. Warren. HILOG: a foundation for higher-order logic programming. *Journal of Logic Programming*, 15(3):187–230, 1993.

[31] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[32] H. de Nivelle. A Resolution Decision Procedure for the Guarded Fragment. In *Proc. of the 15th Int. Conf. on Automated Deduction*, pages 191–204. Springer, 1998.

[33] H. de Nivelle. Splitting through new proposition symbols. In R. Nieuwenhuis and A. Voronkov, editors, *Proc. of the 8th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2001)*, volume 2250 of *LNAI*, pages 172–185, Havana, Cuba, December 2001. Springer.

[34] N. Dershowitz and D.A. Plaisted. Rewriting. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 9, pages 535–610. Elsevier Science, 2001.

[35] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. AL-log: Integrating Datalog and Description Logics. *J. of Intelligent Information Systems*, 10(3):227–252, 1998.

[36] F. M. Donini, D. Nardi, and R. Rosati. Description Logics of Minimal Knowledge and Negation as Failure. *ACM Transactions on Computational Logic*, 3(2):177–225, 2002.

[37] J. Edelmann and B. Owsnicki. Data Models in Knowledge Representation Systems: A Case Study. In C.-R. Rollinger and W. Horn, editors, *Proc. of the 10th German Workshop on Artificial Intelligence (GWAI'86) and the 2nd Austrian Symposium on Artificial Intelligence (ÖGAI'86)*, pages 69–74. Springer, Ottenstein/Niederösterreich, September 1986.

[38] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive Datalog. *ACM Transactions on Database Systems*, 22(3):364–418, 1997.

[39] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. A Deductive System for Non-Monotonic Reasoning. In J. Dix, U. Furbach, and A. Nerode, editors, *Proceedings of the 4th International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR'97)*, volume 1265 of *Lecture Notes in Artificial Intelligence*, pages 364–375, Dagstuhl, Germany, July 1997. Springer.

[40] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining Answer Set Programming with Description Logics for the Semantic Web. In *Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR2004)*. AAAI Press, 2004. To appear.

[41] C. Fermuller, T. Tammet, N. Zamov, and A. Leitsch. *Resolution Methods for the Decision Problem*. Springer, 1993.

[42] M. Fitting. *First-Order Logic and Automated Theorem Proving, 2nd Edition*. Springer, 1996.

[43] H. Ganzinger and H. de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Proceedings of the 14th IEEE Symposium on Logic in Computer Science*, pages 295–305. IEEE Computer Society Press, 1999.

[44] H. Ganzinger, U. Hustadt, C. Meyer, and R. A. Schmidt. A Resolution-Based Decision Procedure for Extensions of K4. In M. Zakharyaschev, K. Segerberg, M. de Rijke, and H. Wansing, editors, *Advances in Modal Logic, Volume 2*, volume 119 of *Lecture Notes*, pages 225–246. CSLI Publications, Stanford, USA, 2001.

[45] François Goasdoué and M.-C. Rousset. Answering Queries using Views: a KRDB Perspective for the Semantic Web. *ACM Journal - Transactions on Internet Technology (TOIT)*, 2004. To appear.

[46] G. Gottlob and A. Leitsch. On the efficiency of subsumption algorithms. *Journal of the ACM*, 32(2):280–295, 1985.

[47] E. Grädel, M. Otto, and E. Rosen. Two-Variable Logic with Counting is Decidable. In *Proc. of 12th IEEE Symposium on Logic in Computer Science LICS '97*, Warsaw, Poland, 1997.

[48] S. Greco. Binding Propagation Techniques for the Optimization of Bound Disjunctive Queries. *IEEE Transactions on Knowledge and Data Engineering*, 15(2):717–736, March/April 2003.

[49] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description Logic Programs: Combining Logic Programs with Description Logic. In *Proc. of the Twelfth Int. World Wide Web Conf. (WWW 2003)*, pages 48–57. ACM, 2003.

[50] V. Haarslev and R. Möller. RACER System Description. In *1st Int. Joint Conf. on Automated Reasoning (IJCAR-01)*, pages 701–706. Springer, 2001.

[51] V. Haarslev and R. Möller. Optimization Strategies for Instance Retrieval. In I. Horrocks, S. Tessaris, and J. Z. Pan, editors, *Proc. Int. Workshop on Description Logics (DL-2002)*, Toulouse, France, April 2002.

[52] V. Haarslev and R. Möller. Incremental Query Answering for Implementing Document Retrieval Services. In D. Calvanese, G. de Giacomo, and F. Franconi, editors, *Proc. Int. Workshop on Description Logics (DL-2003)*, pages 85–94, Rome, Italy, September 2003.

[53] V. Haarslev, R. Möller, and M. Wessel. The Description Logic ALCNHR+ Extended with Concrete Domains: A Practically Motivated Approach. In T. Nipkow R. Gore, A. Leitsch, editor, *Proc. of Int. Joint Conf. on Automated Reasoning, IJCAR 2001*, pages 29–44, Siena, Italy, June 18-23 2001. Springer.

[54] S. Heymans and D. Vermeir. Integrating Semantic Web Reasoning and Answer Set Programming. In M. De Vos and A. Provetti, editors, *Answer Set Programming, Advances in Theory and Implementation, Proc. of the 2nd Int. ASP'03 Workshop*, Volume 78 of CEUR Proceedings, pages 194–208, Messina, Sicily, September 2003.

[55] I. Hodkinson. Loosely guarded fragment of first-order logic has the finite model property. *Studia Logica*, 70:217–274, 2002.

[56] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.

[57] I. Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *Proc. 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, Trento, Italy, June 1998. Morgan Kaufmann Publishers.

[58] I. Horrocks and P. F. Patel-Schneider. A Proposal for an OWL Rules Language. In *Proc. of the Thirteenth Int. World Wide Web Conf.(WWW 2004)*. ACM, 2004.

[59] I. Horrocks and U. Sattler. Ontology Reasoning in the $\mathcal{SHOQ}(\mathbf{D})$ Description Logic. In B. Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204. Morgan Kaufmann, 2001.

[60] I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies. How to decide Query Containment under Constraints using a Description Logic. In *Proc. of the 7th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR 2000)*, Lecture Notes in Artificial Intelligence. Springer, 2000.

[61] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–263, 2000.

[62] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In D. MacAllester, editor, *Proc. of the 17th Int. Conf. on Automated Deduction (CADE 2000)*, number 1831 in Lecture Notes in Artificial Intelligence, pages 482–496. Springer, 2000.

[63] I. Horrocks and S. Tessaris. Querying the Semantic Web: a Formal Approach. In I. Horrocks and J. Hendler, editors, *Proc. of the 13th Int. Semantic Web Conf. (ISWC 2002)*, number 2342 in Lecture Notes in Computer Science, pages 177–191. Springer, 2002.

[64] U. Hustadt. *Resolution-Based Decision Procedures for Subclasses of First-Order Logic*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, November 1999.

[65] U. Hustadt, B. Motik, and U. Sattler. Reasoning in Description Logics with a Concrete Domain in the Framework of Resolution. In R. López de Mántaras and L. Saitta, editors, *Proc. of the 16th European Conf. on Artificial Intelligence (ECAI 2004)*, pages 353–357, Valencia, Spain, August 2004.

[66] U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog Programs. In D. Dubois, C. Welty, and M.-A. Williams, editors, *Proc. of the 9th Int. Conf. on Knowledge Representation and Reasoning (KR2004)*, pages 152–162, Menlo Park, California, USA, June 2004. AAAI Press.

[67] U. Hustadt and R. A. Schmidt. Issues of Decidability for Description Logics in the Framework of Resolution. In R. Caferra and G. Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *LNAI*, pages 192–206. Springer, 1999.

[68] W. H. Joyner Jr. Resolution Strategies as Decision Procedures. *Journal of the ACM*, 23(3):398–417, 1976.

[69] B. Kallick. A decision procedure based on the resolution method. In A. J. H. Morrell, editor, *Proc. of the IFIP Congress*, pages 269–275. Horth-Holland, 1968. Volume 1 - Mathematics, Software.

[70] Y. Kazakov and H. de Nivelle. A Resolution decision procedure for the guarded fragment with transitive guards. In *Proc. of 2nd Int. Joint Conference on Automated Reasoning (IJCAR 2004)*, Cork, Ireland, 2004. To appear.

[71] A. Leitsch. Deciding clause classes by semantic clash resolution. *Fundamenta Informaticae*, 18:63–182, 1993.

[72] A. Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1-2):165–209, 1998.

[73] A. Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems - Special Issue on Networked Information Discovery and Retrieval*, 5(2):121–143, 1995.

[74] C. Lutz. *The Complexity of Reasoning with Concrete Domains*. PhD thesis, Teaching and Research Area for Theoretical Computer Science, RWTH Aachen, 2002.

[75] C. Lutz. NExpTime-complete Description Logics with Concrete Domains. *ACM Transactions on Computational Logic*, 2003. To appear.

[76] C. Lutz and U. Sattler. The Complexity of Reasoning with Boolean Modal Logics. In F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyaschev, editors, *Advances in Modal Logics*, volume 3. CSLI Publications, Stanford, 2001.

[77] R. M. MacGregor. Inside the LOOM description classifier. *SIGART Bull.*, 2(3):88–92, 1991.

[78] William McCune. Solution of the Robbins Problem. *Journal of Automated Reasoning*, 19(3):263–276, 1997.

[79] M. Minsky. A Framework for Representing Knowledge. In J. Haugeland, editor, *Mind Design: Philosophy, Psychology, Artificial Intelligence*, pages 95–128. MIT Press, Cambridge, MA, 1981.

[80] B. Motik, A. Maedche, and R. Volz. Optimizing Query Answering in Description Logics using Disjunctive Deductive Databases. In *10th International Workshop on Knowledge Representation meets Databases (KRDB-2003)*, Hamburg, Germany, September 15-16 2003.

[81] B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with Rules. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *Proc. of the 3rd Int. Semantic Web Conf. (ISWC 2004)*, volume 3298 of *Lecture Notes in Computer Science*, pages 549–563, Hiroshima, Japan, 2004. Springer.

[82] B. Nebel. Terminological Cycles: Semantics and Computational Properties. In J. F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 331–361. Morgan Kaufmann Publishers, San Mateo (CA), USA, 1991.

[83] R. Nieuwenhuis and A. Rubio. Theorem Proving with Ordering and Equality Constrained Clauses. *Journal of Logic and Computation*, 19(4):312–351, April 1995.

[84] R. Nieuwenhuis and A. Rubio. Paramodulation-Based Theorem Proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 7, pages 371–443. Elsevier Science, 2001.

[85] H. De Nivelle, R. A. Schmidt, and U. Hustadt. Resolution-Based Methods for Modal Logics. *Logic Journal of the IGPL*, 8(3):265–292, May 2000.

[86] A. Nonnengart and C. Weidenbach. Computing Small Clause Normal Forms. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 6, pages 335–367. Elsevier Science, 2001.

[87] J. Pan and I. Horrocks. RDFS(FA) and RDF MT: Two Semantics for RDFS. In D. Fensel, K. Sycara, and J. Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 30–46. Springer, 2003.

[88] J. Pan and I. Horrocks. Web Ontology Reasoning with Datatype Groups. In D. Fensel, K. Sycara, and J. Mylopoulos, editors, *Proc. of the 2nd International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 47–63. Springer, 2003.

[89] J.Z. Pan and I. Horrocks. Extending Datatype Support in Web Ontology Reasoning. In *Proc. of the 2002 Int. Conf. on Ontologies, Databases and Applications of SEmantics (ODBASE 2002)*, volume 2519 of *Lecture Notes in Computer Science*, pages 1067–1081. Springer, 2002.

[90] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1993.

[91] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language; Semantics and Abstract Syntax. http://www.w3.org/TR/owl-semantics/, November 2002.

[92] N. Peltier. On the decidability of the PVD class with equality. *Logic Journal of the IGPL*, 9(4):601–624, 2001.

[93] D. A. Plaisted and S. Greenbaum. A Structure-preserving Clause Form Transformation. *Journal of Symbolic Logic and Computation*, 2(3):293–304, 1986.

[94] D. Poole, A. Mackworth, and R. Goebel. *Computational Intelligence: A Logical Approach*. Oxford University Press, 1997.

[95] I. Pratt-Hartmann. Counting Quantifiers and the Stellar Fragment. Technical report, University of Manchester, UK, 2003. submitted for publishing.

[96] M. R. Quillian. Word concepts: A theorey and simulation of some basic capabilities. *Behavioral Science*, 12:410–430, 1967.

[97] R. Reiter. Two Results on Ordering for Resolution with Merging and Linear Format. *Journal of the ACM*, 18(4):630–646, 1971.

[98] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.

[99] A. Riazanov and A. Voronkov. Splitting Without Backtracking. In B. Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 611–617, Seattle, WA, USA, August 2001. Morgan Kaufmann.

[100] J. A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, 1965.

[101] J. A. Robinson. Automatic deduction with hyper-resolution. *International Journal of Computational Mathematics*, 1(3):227–234, 1965.

[102] A. Schaerf. *Query Answering in Concept-Based Knowledge Representation Systems: Algorithms, Complexity, and Semantic Issues*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1994.

[103] K. Schild. A correspondence theory for terminological logics: preliminary report. In *Proc. of 12th Int. Joint Conference on Artificial Intelligence (IJCAI '91)*, pages 466–471, Sidney, AU, 1991.

[104] R. A. Schmidt and U. Hustadt. A Resolution Decision Procedure for Fluted Logic. In D. McAllester, editor, *Proc. of the 17th Int. Conf. on Automated Deduction (CADE-17)*, volume 1831 of *Lecture Notes in Artificial Intelligence*, pages 433–448. Springer, June 17–20 2000.

[105] R. A. Schmidt and U. Hustadt. A Principle for Incorporating Axioms into the First-Order Translation of Modal Formulae. In F. Baader, editor, *Automated Deduction—CADE-19*, volume 2741 of *Lecture Notes in Artificial Intelligence*, pages 412–426. Springer, 2003.

[106] M. Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Proc. of the 1st Inl't Conf. on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 421–431, Los Altos, 1989. Morgan Kaufmann Publishers.

[107] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[108] G. Schreiber. The Web is not well-formed. *IEEE Intelligent Systems*, 17(2):79–80, March/April 2002. Contribution to the section "Trends & Controversies: Ontologies KISSES in Standardization", edited by S. Staab.

[109] O. Shmueli. Decidability and expressiveness aspects of logic queries. In *Proc. of the 6th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 237–249. ACM Press, 1987.

[110] O. Shmueli. Equivalence of datalog queries is undecidable. *Journal of Logic Programming*, 15(3):231–241, 1993.

[111] M. E. Stickel. Automated Deduction by Theory Resolution. *Journal of Automated Reasoning*, 1(4):333–355, 1985.

[112] T. Tammet. *Resolution Methods for Decision Problems and Finite-Model Building*. PhD thesis, Chalmers University of Technology / University of Göteborg, 1992.

[113] S. Tessaris. *Questions and answers: reasoning and querying in Description Logic*. PhD thesis, University of Manchester, UK, 2001.

[114] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, Germany, 2001.

[115] M. Vardi. Why is modal logic so robustly decidable? In N. Immerman and P. Kolaitis, editors, *Descriptive Complexity and Finite Models*, volume 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 149–184. AMS, 1997.

[116] R. Volz. *Web Ontology Reasoning With Logic Databases*. PhD thesis, Universität Fridericiana zu Karlsruhe (TH), Germany, 2004.

[117] C. Welty and D. Ferrucci. What's in an instance? Technical Report 94-18, Max-Planck-Institut, 1994. RPI Computer Science.