A Refined Architecture for Terminological Systems: Terminology = Schema + Views

M. Buchheit, F. M. Donini, W. Nutt, A. Schaerf

RR-95-0
Research Repo

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research Report

RR-95-09

A Refined Architecture for Terminological Systems: Terminology = Schema + Views

M. Buchheit, F. M. Donini, W. Nutt, A. Schaerf

June 1995

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341

Deutsches Forschungszentrum
für
Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, Sema Group, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry of Education, Science, Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct systems with technical knowledge and common sense which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- ☐ Intelligent Engineering Systems
- ☐ Intelligent User Interfaces
- ☐ Computer Linguistics
- ☐ Programming Systems
- ☐ Deduction and Multiagent Systems
- ☐ Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Dr. Dr. D. Ruland

Director

A Refined Architecture for Terminological Systems: Terminology = Schema + Views

M. Buchheit, F. M. Donini, W. Nutt, A. Schaerf

DFKI-RR-95-09

A version of this paper appeared in the Proceedings of the 12th National Conference on Artificial Intelligence, AAAI'94.

This work has been supported by a grant from The Federal Ministry of Education, Science, Research and Technology (FKZ ITWM-9201).

ISSN 0946-008X

A Refined Architecture for Terminological Systems: Terminology = Schema + Views

M. Buchheit[1] F. M. Donini[2] W. Nutt[1] A. Schaerf[2]

1. German Research Center for Artificial Intelligence – DFKI GmbH
Saarbrücken, Germany
{buchheit,nutt}@dfki.uni-sb.de

2. Dipartimento di Informatica e Sistemistica,
Università di Roma "La Sapienza", Roma, Italy
{donini,aschaerf}@dis.uniroma1.it

Abstract

Traditionally, the core of a Terminological Knowledge Representation System (TKRS) consists of a TBox, where concepts are introduced, and an ABox, where facts about individuals are stated in terms of concept memberships. This design has a drawback because in most applications the TBox has to meet two functions at a time: On the one hand—similarly to a database schema—frame-like structures with type information are introduced through primitive concepts and primitive roles; on the other hand, views on the objects in the knowledge base are provided through defined concepts.

We propose to account for this conceptual separation by partitioning the TBox into two components for primitive and defined concepts, which we call the *schema* and the *view* part. We envision the two parts to differ with respect to the language for concepts, the statements allowed, and the semantics.

We argue that this separation achieves more conceptual clarity about the role of primitive and defined concepts and the semantics of terminological cycles. Three case studies show the computational benefits to be gained from the refined architecture.

Contents

1 Introduction

Research on terminological reasoning usually presupposes the following abstract architecture of a knowledge representation system, which quite well reflects the structure of implemented systems. There is a logical representation language that allows for two kinds of statements: In the TBox, or *terminology*, concept descriptions are introduced, and in the ABox, or *world description*, individuals are characterized in terms of concept membership and role relationship. This abstract architecture has been the basis for the design of systems, such as CLASSIC (Borgida, Brachman, McGuinness, & Alperin Resnick, 1989), BACK (Peltason, 1991), LOOM (MacGregor, 1991), and KRIS (Baader & Hollunder, 1991), the development of algorithms (see, *e.g.*, Nebel, 1990a), and the investigation of the computational properties of inferences (see, *e.g.*, Nebel, 1990b; Donini, Lenzerini, Nardi, & Schaerf, 1994).

Given this setting, there are three parameters that characterize a terminological system: (*i*) the language for concept descriptions, (*ii*) the form of the statements allowed, and (*iii*) the semantics given to concepts and statements. Research tried to improve systems by modifying these three parameters. But in all existing systems and almost all theoretical studies language and semantics are supposed to be uniform for all components.[1]

The results of those studies were unsatisfactory in at least two respects. First, it seems that tractable inferences are only possible for languages with little expressivity. Second, no consensus has been reached about the semantics of terminological cycles, although in applications the need to model cyclic dependencies between classes of objects arises constantly (see, *e.g.*, MacGregor, 1992).

Based on experience with applications of terminological systems, we suggest to refine the two-layered architecture consisting of TBox and ABox. Our goal is twofold: On the one hand we want to achieve more conceptual clarity about the role of primitive and defined concepts and the semantics of terminological cycles; on the other hand, we want to improve the tradeoff between expressivity and worst-case complexity. Since our changes are not primarily motivated by mathematical considerations but by the way systems are used, we expect to come up with a more practical system design.

In applications we found that the TBox has to meet two functions at a time. One is to declare frame-like structures by introducing primitive concepts and roles, together with type information like isa-relationships between concepts, or range restrictions and number restrictions of roles. For example, suppose we want to model a company environment. Then we may introduce the concept Employee with slots lives-in of type City, works-for of type Department, salary of type Salary, and boss of type Manager. The slots lives-in and salary have exactly one filler, works-for may have more than one filler. The concept Manager is a specialization of Employee,

[1]In (Lenzerini & Schaerf, 1991) a combination of a weak language for ABoxes and a strong language for queries has been investigated.

having a salary in HighSalary. Such declarations are similar to class declarations in object-oriented systems. For this purpose, a simple language is sufficient. Cycles occur naturally in modeling tasks, *e.g.*, the boss of an Employee is a Manager and therefore also an Employee. These declarations have no definitional import; they just restrict the set of possible interpretations.

The second function of a TBox is to define new concepts in terms of primitive ones by specifying necessary *and* sufficient conditions for concept membership. This can be seen as defining *abstractions* or *views* on the objects in the knowledge base. Defined concepts are important for querying the knowledge base and as left-hand sides of trigger rules. For this purpose we need more expressive languages. If cycles occur in this part they *must* have definitional import.

As an outcome of our analysis we propose to split the TBox into two components: one for declaring frame structures and one for defining views. By analogy to the structure of databases we call the first component the *schema* and the second the *view* part. We envision the two parts to differ with respect to the language, the form of statements, and the semantics of cycles.

The schema consists of a set of primitive concept introductions, formulated in the *schema language*, and the view part consists of a set of concept definitions, formulated in the *view language*. In general, the schema language will be less expressive than the view language. Since the role of statements in the schema is to restrict the interpretations we admit, first order semantics—also called descriptive semantics in this context (see Nebel, 1991)—is adequate for cycles occurring in the schema. For cycles in the view part, we propose to choose a semantics that defines concepts uniquely, *e.g.*, least or greatest fixpoint semantics.

The purpose of this work is not to present the full-fledged design of a new system, but to explore the options that arise from the separation of the TBox into schema and views. Among the benefits to be gained from this refinement are the following three. First, the new architecture has more parameters for improving systems, since language, form of statements, and semantics can be specified differently for schema and views. So we found a combination of schema and view language that allows for polynomial inference procedures whereas merging the two languages into one leads to intractability. Second, we believe that one of the obstacles to a consensus about the semantics of terminological cycles has been precisely the fact that no distinction has been made between primitive and defined concepts. Moreover, intractability of reasoning with cycles mostly refers to inferences with defined concepts. We proved that reasoning with cycles is easier when only primitive concepts are considered. Third, the refined architecture allows for more differentiated complexity measures, as will be shown in the sequel.

In the following section we outline our refined architecture of a TKRS, which comprises *three* parts: the *schema*, the *view taxonomy*, and the *world description*, dealing with primitive concepts, defined concepts and assertions in traditional systems, respectively. In Section 3 we examine the effect of terminological cycles in

Construct Name	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
singleton set	$\{a\}$	$\{a^{\mathcal{I}}\}$
intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
complement	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
universal quantification	$\forall R.C$	$\{d_1 \mid \forall d_2 : (d_1, d_2) \in R^{\mathcal{I}} \rightarrow\ d_2 \in C^{\mathcal{I}}\}$
existential quantification	$\exists R.C$	$\{d_1 \mid \exists d_2 : (d_1, d_2) \in R^{\mathcal{I}} \wedge d_2 \in C^{\mathcal{I}}\}$
existential agreement	$\exists Q \doteq R$	$\{d_1 \mid \exists d_2 : (d_1, d_2) \in Q^{\mathcal{I}} \wedge (d_1, d_2) \in R^{\mathcal{I}}\}$
number restrictions	$(\geq n\, R)$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \geq n\}$
	$(\leq n\, R)$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \leq n\}$

Table 1: Syntax and semantics of concept forming constructs.

Construct Name	Syntax	Semantics
inverse role	P^{-1}	$\{(d_1, d_2) \mid (d_2, d_1) \in P^{\mathcal{I}}\}$
role restriction	$(R{:}\,C)$	$\{(d_1, d_2) \mid (d_1, d_2) \in R^{\mathcal{I}} \wedge d_2 \in C^{\mathcal{I}}\}$
role chain	$Q \circ R$	$\{(d_1, d_3) \mid \exists d_2 : (d_1, d_2) \in Q^{\mathcal{I}} \wedge (d_2, d_3) \in R^{\mathcal{I}}\}$
role conjunction	$Q \sqcap R$	$\{(d_1, d_2) \mid (d_1, d_2) \in Q^{\mathcal{I}} \wedge (d_1, d_2) \in R^{\mathcal{I}}\}$
self	ϵ	$\{(d_1, d_1) \mid d_1 \in \Delta^{\mathcal{I}}\}$

Table 2: Syntax and semantics of role forming constructs.

our architecture and in Section 4, schemas are considered in detail. In Section 5, we show by three case studies that adding a simple schema with cycles to existing systems does not increase the complexity of reasoning. Finally, conclusions are drawn in Section 6.

2 The Refined Architecture

We start this section by a short reminder on concept languages. Then we discuss the form of statements and their semantics in the different components of a TKRS. Finally, we specify the reasoning services provided by each component and introduce different complexity measures for analyzing them.

2.1 Concept Languages

In concept languages, complex concepts (ranged over by C, D) and complex roles (ranged over by Q, R) can be built up from simpler ones using concept and role forming constructs (see Tables 1 and 2 for a set of common constructs). The basic syntactic symbols are (i) *concept names*, which are divided into *schema names* (ranged over by A, B) and *view names* (ranged over by V), (ii) *role names* (ranged over by P), and (iii) *individual names* (ranged over by a, b). An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of the *domain* $\Delta^{\mathcal{I}}$ and the *interpretation function* $\cdot^{\mathcal{I}}$, which maps every concept to a subset of $\Delta^{\mathcal{I}}$, every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and every individual to an element of $\Delta^{\mathcal{I}}$. We assume that different individuals are mapped to different elements of $\Delta^{\mathcal{I}}$, *i.e.*, $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ for $a \neq b$. This restriction is usually called *Unique Name Assumption* (UNA). Complex concepts and roles are interpreted according to the semantics given in Tables 1 and 2, respectively (with $\sharp X$ we denote the cardinality of the set X). We call two concepts C and D *equivalent* (written $C \equiv D$), iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every interpretation $\mathcal{I}$. A *subconcept* of a concept C is a substring of C that is itself a concept.

In our architecture, there are two different concept languages in a TKRS, a *schema language* for expressing schema statements and a *view language* for formulating views and queries to the system. The schema language allows only for schema names whereas the view language allows for both schema and view names. The view and schema languages in the case studies will be defined by restricting the set of concept and role forming constructs to a subset of those in Tables 1 and 2.

2.2 The Three Components

Now we describe the three parts of a TKRS: the schema, the view taxonomy and the world description. We first focus our attention to the schema.

2.2.1 The Schema

The schema introduces concept and role names and states isa-relationships between concepts and elementary type constraints for the roles. Figure 1 shows a part of the concepts and roles that models the company environment. Concepts are represented by ovals, (direct) isa relationships by dotted arrows and roles by normal arrows.

Formally, relationships between concepts and type constraints on roles are stated by *inclusion axioms* having one of the forms:

$$A \sqsubseteq D, \qquad P \sqsubseteq A_1 \times A_2,$$

where A, A_1, A_2 are schema names, P is a role name, and D is a concept of the schema language. Intuitively, the first axiom, called a *concept inclusion*, states that all instances of A are also instances of D. The second axiom, called a *role inclusion*,

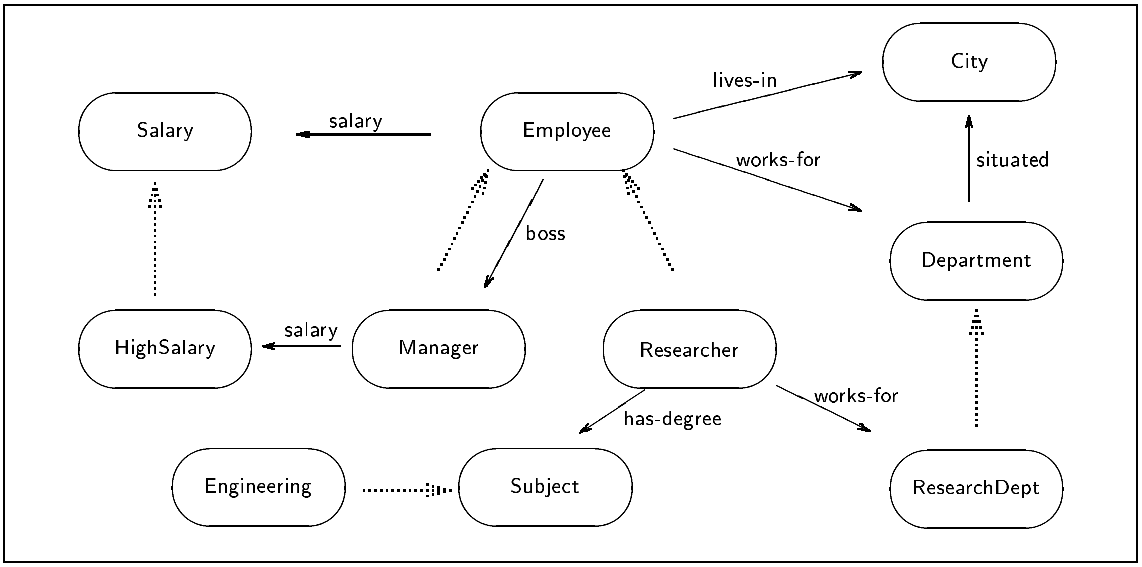

Figure 1: Concepts and roles in the company environment

Employee	$\sqsubseteq$	$(= 1\ \mathsf{salary})$	Engineering	$\sqsubseteq$	Subject
"	$\sqsubseteq$	$(= 1\ \mathsf{lives\text{-}in})$	HighSalary	$\sqsubseteq$	Salary
Manager	$\sqsubseteq$	Employee	salary	$\sqsubseteq$	Employee $\times$ Salary
"	$\sqsubseteq$	$\forall\mathsf{salary}.\mathsf{HighSalary}$	boss	$\sqsubseteq$	Employee $\times$ Manager
Researcher	$\sqsubseteq$	Employee	works-for	$\sqsubseteq$	Employee $\times$ Department
"	$\sqsubseteq$	$\forall\mathsf{works\text{-}for}.\mathsf{ResearchDept}$	lives-in	$\sqsubseteq$	Employee $\times$ City
"	$\sqsubseteq$	$(\geq 1\ \mathsf{has\text{-}degree})$	has-degree	$\sqsubseteq$	Researcher $\times$ Subject
ResearchDept	$\sqsubseteq$	Department	situated	$\sqsubseteq$	Department $\times$ City

Figure 2: Schema axioms for the company environment

states that the role P has domain A_1 and codomain A_2. A *schema* $\mathcal{S}$ consists of a finite set of inclusion axioms. An interpretation $\mathcal{I}$ *satisfies* an axiom $A \sqsubseteq D$ if $A^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it *satisfies* $P \sqsubseteq A_1 \times A_2$ if $P^{\mathcal{I}} \subseteq A_1^{\mathcal{I}} \times A_2^{\mathcal{I}}$. The interpretation $\mathcal{I}$ is a *model* of the schema $\mathcal{S}$ if it satisfies all axioms in $\mathcal{S}$. Given a schema $\mathcal{S}$ and two concepts C, D, we say that C is *$\mathcal{S}$-satisfiable* if there is a model $\mathcal{I}$ of $\mathcal{S}$ such that $C^{\mathcal{I}} \neq \emptyset$, and we say that C is *$\mathcal{S}$-subsumed* by D, written $C \sqsubseteq_{\mathcal{S}} D$ or $\mathcal{S} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{S}$.

In Figure 2 we give the schema axioms for the company example of Figure 1. The fact that the role salary has the domain Employee and the codomain Salary is stated by the axiom salary $\sqsubseteq$ Employee $\times$ Salary. The restriction that an Employee must have exactly one salary is expressed by the two axioms Employee $\sqsubseteq (\geq 1\ \mathsf{salary})$ and Employee $\sqsubseteq (\leq 1\ \mathsf{salary})$[2]. The fact that every Manager is an Employee leads

[2]Two axioms of the form $A \sqsubseteq (\leq 1\, P)$ and $A \sqsubseteq (\geq 1\, P)$ are abbreviated by $A \sqsubseteq (= 1\, P)$.

to the axiom $\mathsf{Manager} \sqsubseteq \mathsf{Employee}$, and that a Manager must have a HighSalary to $\mathsf{Manager} \sqsubseteq \forall \mathsf{salary}.\mathsf{HighSalary}$[3].

Inclusion axioms impose only necessary conditions for being an instance of the schema name on the left-hand side. For example, the axiom "$\mathsf{Manager} \sqsubseteq \mathsf{Employee}$" declares that every manager is an employee, but does not give a sufficient condition for being a manager.[4]

A schema may contain *cycles* through inclusion axioms. So one may state that the bosses of an employee are themselves employees, writing "$\mathsf{Employee} \sqsubseteq \forall \mathsf{boss}.\mathsf{Employee}$." In general, existing systems (such as CLASSIC and KRIS) do not allow for terminological cycles, which is a serious restriction, since cycles are ubiquitous in domain models. One of the main issues related to cycles is to fix their semantics. We argue that axioms in the schema have the role of narrowing down the class of models we consider possible. Therefore, they should be interpreted under so-called descriptive semantics, which takes all models into consideration for reasoning. Nebel (1991) proposes two other kinds of semantics in the presence of cycles, namely least fixpoint and greatest fixpoint semantics, which take into account only models that in some sense are the least or greatest, respectively. We will discuss this issue in more detail in Section 3.

2.2.2 The View Taxonomy

The *view part* contains *view definitions* of the form

$$V \doteq C,$$

where V is a view name and C is a concept in the view language. Views provide abstractions by defining new classes of objects in terms of other views and the concept and role names introduced in the schema. We refer to "$V \doteq C$" as the *definition* of V. The distinction between schema and view names is crucial for our architecture. It ensures the separation between schema and views.

A *view taxonomy* $\mathcal{V}$ is a finite set of view definitions such that (*i*) for each view name there is at most one definition, and (*ii*) each view name occurring on the right hand side of a definition has a definition in $\mathcal{V}$.

Differently from schema axioms, view definitions give necessary *and* sufficient conditions. As an example of a view, using the inverse of boss, one can describe the bosses of the employee Bill as the instances of "$\mathsf{BillsBosses} \doteq \exists \mathsf{boss}^{-1}.\{\mathsf{BILL}\}$."

An interpretation $\mathcal{I}$ *satisfies* the definition $V \doteq C$ if $V^{\mathcal{I}} = C^{\mathcal{I}}$, and it is a *model* for a view taxonomy $\mathcal{V}$ if $\mathcal{I}$ satisfies all definitions in $\mathcal{V}$.

[3]The introduced syntax for defining a schema is well-suited for studying the theoretical properties of the new architecture. However, in a real system one would implement more user-friendly languages as they are known from frame systems and object-oriented databases.

[4]It gives, though, a sufficient condition for being an employee: If an individual is asserted to be a Manager we can deduce that it is an Employee, too.

Whether or not to allow cycles in view definitions is a delicate design decision. Differently from the schema, the role of cycles in the view part is to state recursive definitions. In this case, descriptive semantics is not adequate because it might not determine uniquely the extension of defined concepts from the extension of the other ones. We will discuss this problem in general in the section on terminological cycles. In this paper however, we only deal with cycle-free view taxonomies. Therefore this problem does not arise and descriptive semantics is adequate.

2.2.3 The World Description

A state of affairs in the world is described by *assertions* of the form

$$a\colon C, \qquad aRb,$$

where C and R are concept and role descriptions in the view language. Intuitively, an assertion $a\colon C$ states that a is an instance of the concept C, and aRb states that a is in relation with b through the role R. Assertions of the form $a\colon A$ or aPb, where A and P are names in the schema, resemble basic facts in a database. Assertions involving view names and complex concepts are comparable to view updates.

A *world description* $\mathcal{W}$ is a finite set of assertions. The semantics is as usual: an interpretation $\mathcal{I}$ *satisfies* $a\colon C$ if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ and it *satisfies* aRb if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$; it is a *model* of $\mathcal{W}$ if it satisfies every assertion in $\mathcal{W}$.

Summarizing, a *knowledge base* is a triple $\Sigma = \langle \mathcal{S}, \mathcal{V}, \mathcal{W} \rangle$, where $\mathcal{S}$ is a schema, $\mathcal{V}$ a view taxonomy, and $\mathcal{W}$ a world description. An interpretation $\mathcal{I}$ is a *model* of a knowledge base if it is a model of all three components. A knowledge base is *satisfiable* if there exists a model for it. The concept names that occur on the left-hand side of a view definition are called *defined concepts*, the other ones are called *atomic concepts*. All role names are *atomic roles*, since their are no role definitions.

2.3 Reasoning Services

For each component, there is a prototypical reasoning service to which the other services can be reduced.

Schema Validation: Given a schema $\mathcal{S}$, check whether there exists a model of $\mathcal{S}$ that interprets every schema name as a nonempty set.

View Subsumption: Given a schema $\mathcal{S}$, a view taxonomy $\mathcal{V}$, and view names V_1 and V_2, check whether $V_1^{\mathcal{I}} \subseteq V_2^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{S}$ and $\mathcal{V}$. This is written as $\mathcal{S}, \mathcal{V} \models V_1 \sqsubseteq V_2$ or as $V_1 \sqsubseteq_{\mathcal{S},\mathcal{V}} V_2$.

Instance Checking: Given a knowledge base Σ, an individual a, and a view name V, check whether $a^{\mathcal{I}} \in V^{\mathcal{I}}$ holds in every model $\mathcal{I}$ of Σ. This is written as $\Sigma \models a\colon V$.

Schema validation supports the knowledge engineer by checking whether the skeleton of his/her domain model is consistent. Instance checking is the basic operation in querying a knowledge base. View subsumption helps in organizing and optimizing queries (see *e.g.*, Buchheit, Jeusfeld, Nutt, & Staudt, 1994). Note that the schema $\mathcal{S}$ has to be taken into account in all three services and that the view taxonomy $\mathcal{V}$ is relevant not only for view subsumption, but also for instance checking. In systems that forbid cycles, one can get rid of $\mathcal{S}$ and $\mathcal{V}$ by expanding definitions (as shown in Nebel, 1990b). This is not possible when $\mathcal{S}$ and/or $\mathcal{V}$ are cyclic.

2.4 Complexity Measures

The separation of the core of a TKRS into three components allows us to introduce refined complexity measures for analyzing the difficulty of inferences.

The complexity of a problem is generally measured with respect to the size of the whole input. However, with regard to our setting, three different pieces of input are given, namely the schema, the view taxonomy, and the world description. For this reason, different kinds of complexity measures may be defined, similarly to what has been suggested in (Vardi, 1982) for queries over relational databases. We consider the following measures (where $|X|$ denotes the size of X):

Schema Complexity: the complexity as a function of $|\mathcal{S}|$;

View Complexity: the complexity as a function of $|\mathcal{V}|$;

World Description Complexity: the complexity as a function of $|\mathcal{W}|$;

Combined Complexity: the complexity as a function of $|\mathcal{S}| + |\mathcal{V}| + |\mathcal{W}|$.

The combined complexity takes into account the whole input. The other three instead consider only a part of the input, so they are meaningful only when it is reasonable to suppose that the size of the other parts is negligible. For instance, it is sensible to analyze the schema complexity of view subsumption because usually the schema is much bigger than the two views which are compared. Similarly, one might be interested in the world description complexity of instance checking whenever one can expect $\mathcal{W}$ to be much larger than the schema and the view part.

It is worth noticing that for every problem the combined complexity, taking into account the whole input, is at least as high as the other three. For example, if the complexity of a problem is $O(|\mathcal{S}| \cdot |\mathcal{V}| \cdot |\mathcal{W}|)$, the combined complexity is cubic, whereas the other ones are linear. Similarly, if the complexity of a given problem is $O(|\mathcal{S}|^{|\mathcal{V}|})$, the combined complexity and the view complexity are exponential, the schema complexity is polynomial, and the world description complexity is constant.

In this paper, we use combined complexity to compare the complexity of reasoning in our architecture with reasoning in the traditional one. Moreover, we use schema complexity to show how the presence of a large schema affects the complexity of the reasoning services previously defined. View and world description complexity have been investigated (under different names) in (Nebel, 1990b; Baader, 1990) and (Schaerf, 1993; Donini et al., 1994), respectively.

For a general description of the complexity classes we use, see (Johnson, 1990).

3 Terminological Cycles

Terminologies with cycles—so called "terminological cycles"—have been investigated by a number of researchers. There are two main issues related to terminological cycles: The first is to fix the semantics and the second, based on this, to come up with a proper inference procedure. In this section we discuss in detail the problem of semantics. To this end, we first recall some definitions and then summarize the previous work on this topic. Then we examine the different possibilities of a semantics for our formalism. It shows up that our choice, the descriptive semantics, comes off best. The problem of inferences and the influence of the different kinds of cycles to their complexity will be dealt with in Section 4 and 5.

3.1 Semantics for Cycles

Intuitively, a set of inclusions or definitions is cyclic, if a concept name appearing on the left-hand side also appears on the right-hand side. In the following, we will formally define when a terminology, schema or view taxonomy is cyclic. Then we review various kinds of semantics for cycles. For the moment we suppose that a schema consists only of concept inclusions. In Section 4 we will extend this to role inclusions. There we will also distinguish between different types of cycles and their effects on the complexity of inferences for concrete schema languages.

Let $\mathcal{T}$ be a terminology consisting of concept inclusions and view definitions where for each view name there is at most one definition. We define the *dependency graph* $D(\mathcal{T})$ of $\mathcal{T}$ as follows. The nodes are the concept names in $\mathcal{T}$. Let A_1, A_2 be two nodes. There is an edge from A_1 to A_2, iff there is a concept inclusion or a view definition with A_1 on its left-hand side and A_2 appearing on its right-hand side. We say $\mathcal{T}$ is *cyclic*, if $D(\mathcal{T})$ contains a cycle, and *cycle-free* otherwise. Let $\Sigma = \langle \mathcal{S}, \mathcal{V}, \mathcal{W} \rangle$ be a knowledge base. We say $\mathcal{S}$ is cyclic, if $D(\mathcal{S})$ contains a cycle. We say, $\mathcal{V}$ is cyclic, if $D(\mathcal{S} \cup \mathcal{V})$ contains a cycle. Note that, since view names are not allowed in the schema, $D(\mathcal{S} \cup \mathcal{V})$ contains a cycle if and only if $D(\mathcal{V})$ contains one.

To come up with a semantics for a terminology means to define which of its models should be considered for reasoning. This is a problem when cycles are present since an interpretation of the atomic concepts might be extendible to a model of the terminology in more than one way. Therefore, the defined concepts are not uniquely determined by the atomic ones. This is counterintuitive to the idea of a "definition." So one has to restrict the models taken into account. Nebel (1990a) proposes three types of semantics for a terminology in the presence of cycles: *descriptive semantics*, *least fixpoint semantics (lfp-semantics)*, and *greatest fixpoint semantics (gfp-semantics)*. The descriptive semantics takes into account—as usual first-order semantics—all models of a terminology. The lfp- and gfp-semantics take into account only those models that are in some sense minimal or maximal. To make this idea more precise, we need some definitions.

Let $\mathcal{T}$ be a set of concept definitions, $\mathcal{T} := \{A_i \doteq C_i \mid i \in 1..n\}$, where each A_i occurs only once as the left-hand side of a definition, *i.e.*, $A_i \neq A_j$ for $i \neq j$. An *atomic interpretation* $\mathcal{J}$ of $\mathcal{T}$ interprets only the atomic concepts and roles in $\mathcal{T}$. An atomic interpretation $\mathcal{J}$ can be extended to an interpretation of $\mathcal{T}$ by defining the denotation of the A_i's. Note that not every extension of $\mathcal{J}$ is a model of $\mathcal{T}$.

Let $\mathcal{J}$ be an atomic interpretation of $\mathcal{T}$ with domain Δ. Let 2^Δ denote the set of all subsets of Δ and $(2^\Delta)^n$ the n-fold Cartesian product of 2^Δ. We define a mapping $\mathcal{T}_\mathcal{J}\colon (2^\Delta)^n \to (2^\Delta)^n$ by

$$\mathcal{T}_\mathcal{J}(\vec{O}) := (C_1^\mathcal{I}, \ldots, C_n^\mathcal{I}),$$

where $\vec{O} := (O_1, \ldots, O_n)$ and $\mathcal{I}$ is the extension of $\mathcal{J}$ defined by $A_i^\mathcal{I} := O_i$ for $i \in 1..n$.

A *fixpoint* of $\mathcal{T}_\mathcal{J}$ is an $\vec{O} \in (2^\Delta)^n$ such that $\mathcal{T}_\mathcal{J}(\vec{O}) = \vec{O}$. Obviously, the interpretation defined by $\mathcal{J}$ and $\vec{O}$ is a model of $\mathcal{T}$ if and only if $\vec{O}$ is a fixpoint of $\mathcal{T}_\mathcal{J}$.

A mapping $T\colon D \to D$ on a complete lattice $(D, \leq)$ is called *monotonic* if $a \leq b$ implies $T(a) \leq T(b)$ for all $a, b \in D$. Every monotonic mapping on a complete lattice has a fixpoint. Among the fixpoints there is a *greatest fixpoint* and a *least fixpoint* (see *e.g.*, Lloyd, 1987, Chapter 1, Section 5). Let "$\leq$" be the componentwise subset ordering on $(2^\Delta)^n$. Since $((2^\Delta)^n, \leq)$ is a complete lattice, every monotonic mapping $\mathcal{T}_\mathcal{J}$ has a greatest and a least fixpoint. There exist simple syntactic criteria on terminologies which guarantee that, for a given $\mathcal{T}$, all $\mathcal{T}_\mathcal{J}$ are monotonic for all $\mathcal{J}$ (see *e.g.*, Schild, 1994b). We say that a terminology $\mathcal{T}$ is *monotonic* if the $\mathcal{T}_\mathcal{J}$ are monotonic for all $\mathcal{J}$.

For a set of concept definitions $\mathcal{T}$ the gfp-semantics takes into account only those models of $\mathcal{T}$ that are the greatest fixpoint of some mapping $\mathcal{T}_\mathcal{J}$ (gfp-models). The lfp-semantics takes into account only those models of $\mathcal{T}$ that are the least fixpoint of some mapping $\mathcal{T}_\mathcal{J}$ (lfp-models).

3.2 Previous Work

There exists a rich body of research on the semantics of terminological cycles and on algorithms for reasoning in their presence.

In (Baader, 1990) inferences with respect to the three types of semantics for the language $\mathcal{FL}_0$, containing concept conjunction and universal quantification, are characterized as decision problems for finite automata. Baader argues that "as it stands, the gfp-semantics comes off best" (see Baader, 1990, page 626). In (Nebel, 1991) these characterizations are extended to the language $\mathcal{TLN}$, which extends $\mathcal{FL}_0$ by number restrictions. Nebel argues that "the only semantics, which covers our intuitions is the descriptive one" (see Nebel, 1990a, page 135). In both languages, the presence of cycles increases the complexity of reasoning. For example, the complexity of subsumption with respect to a terminology rises from NP-complete to PSPACE-complete for lfp- and gfp-semantics.

Dionne, Mays, and Oles (1992, 1993) base their approach to the semantics of cycles on non-wellfounded set theory. They consider a limited language for which they show that subsumption under their semantics is equivalent to subsumption under gfp-semantics.

Reasoning with respect to descriptive semantics has been considered in (Baader, Bürckert, Hollunder, Nutt, & Siekmann, 1990) for the language $\mathcal{ALC}$ and in (Buchheit, Donini, & Schaerf, 1993) for $\mathcal{ALCNR}$. The language $\mathcal{ALC}$ extends $\mathcal{FL}_0$ by complements of concepts and $\mathcal{ALCNR}$ extends $\mathcal{ALC}$ by role conjunction and number restrictions.[5] $\mathcal{ALCNR}$ is the language of the system KRIS. For both $\mathcal{ALC}$ and $\mathcal{ALCNR}$ subsumption checking with cyclic definitions is EXPTIME-hard, whereas the problem is PSPACE-complete for cycle-free terminologies.

An approach based on the μ-calculus was proposed independently by Schild (1994b) and De Giacomo and Lenzerini (1994). Following this approach it is possible to specify locally in a terminology whether to apply lfp- or gfp-semantics to a particular definition. This offers optimal flexibility but it leaves the burden of choice to the user and not to the designer of the system.

Summarizing one can say that the presence of terminological cycles increases the complexity of reasoning in the examined cases. No consensus has been reached as to which semantics—lfp-, gfp-, or descriptive—should be preferred.

3.3 Inclusions versus Definitions

In order to apply the different kinds of semantics to our schema formalism and to examine the consequences, we have to transform inclusions into definitions, since fixpoint semantics is defined only for sets of definitions. Nebel (1990a) proposes to transform an inclusion $A \sqsubseteq C$ into a definition $A \doteq \bar{A} \sqcap C$ where $\bar{A}$ is a new concept name. Schild (1994b) proposes the transformation into $A \doteq A \sqcap C$. However, both transformations are unsatisfactory or even unnecessary for schema inclusions as we will show in the following.

Let $\mathcal{S} = \{A_i \sqsubseteq C_i \mid i \in 1..n\}$ be a set of inclusion axioms. Without loss of generality, we suppose that each A_i occurs only once on the left-hand side, since

[5] See Section 5.1 for a formal definition of the two languages.

inclusions $A \sqsubseteq D_1, \ldots, A \sqsubseteq D_m$ can be replaced by the single inclusion $A \sqsubseteq D_1 \sqcap \ldots \sqcap D_n$. With $\bar{\mathcal{S}} := \{A_i \doteq \bar{A}_i \sqcap C_i \mid i \in 1..n\}$ we denote the transformation proposed by Nebel, with $\mathcal{S}^{\sqcap} := \{A_i \doteq A_i \sqcap C_i \mid i \in 1..n\}$ the one proposed by Schild, and with $\mathcal{S}^{=} := \{A_i \doteq C_i \mid i \in 1..n\}$ the one that replaces the inclusions by definitions. Obviously, every model of $\bar{\mathcal{S}}$, $\mathcal{S}^{\sqcap}$, or $\mathcal{S}^{=}$ is also a model of $\mathcal{S}$.

Now we consider in turn the different combinations of lfp- and gfp-semantics and the two transformations of Nebel and Schild. Taking lfp-semantics has for both transformations the consequence that naturally arising models are omitted. Obviously, an lfp-model of $\mathcal{S}^{\sqcap}$ interprets each A_i as the empty set, independently of the interpretation of the C_i. In order to examine the transformation $\bar{\mathcal{S}}$, we consider an example. Let $\mathcal{S}$ be the schema $\mathcal{S} = \{A \sqsubseteq \forall P.A\}$. The lfp-models of $\bar{\mathcal{S}} = \{A \doteq \bar{A} \sqcap \forall P.A\}$ can be characterized in terms of P-chains. A P-chain is a sequence of objects where each is a P-filler of its predecessor. An lfp-model of $\bar{\mathcal{S}}$ interprets A as all the instances of $\bar{A}$ for which all the objects reachable by a P-chain are again in $\bar{A}$ and from which no infinite P-chain is issuing (see Baader, 1990). This means that models containing a cyclic P-chain are omitted. For example, with the schema $\mathcal{S} = \{\mathsf{Employee} \sqsubseteq \forall \mathsf{is\text{-}deputy\text{-}of}.\mathsf{Employee}\}$ and the world description where JOE $\mathsf{is\text{-}deputy\text{-}of}$ MARY and MARY $\mathsf{is\text{-}deputy\text{-}of}$ JOE, with the lfp-semantics, JOE and MARY cannot be $\mathsf{Employee}$s. This shows that the approach of taking lfp-semantics is not acceptable.

Before we consider the combinations of gfp-semantics with the two transformations, we have to introduce some notations. Let $T\colon D \to D$ be a mapping on a complete lattice $(D, \leq)$. With $\mathit{gfp}(T)$ we denote the greatest fixpoint of T. Let X be a subset of D. With $\mathit{lub}X$ we denote the least upper bound of X. The next result is a weak form of the Proposition 5.1 in (Lloyd, 1987).

Proposition 3.1 *Let $T\colon D \to D$ be a monotonic mapping on the complete lattice $(D, \leq)$. Then $\mathit{gfp}(T) = \mathit{lub}\{x \mid x \leq T(x)\}$.*

The following proposition, due to Schild (1994a) shows that for a large class of schemas, $\mathcal{S}^{\sqcap}$ and $\mathcal{S}^{=}$ are equivalent under gfp-semantics.

Proposition 3.2 *Let $\mathcal{S}$ be a set of inclusion axioms. Suppose that $\mathcal{S}^{\sqcap}$ and $\mathcal{S}^{=}$ are monotonic. Then an interpretation $\mathcal{I}$ is a gfp-model of $\mathcal{S}^{\sqcap}$ iff $\mathcal{I}$ is a gfp-model of $\mathcal{S}^{=}$.*

Proof. Let $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ be an interpretation and $\mathcal{J}$ the corresponding atomic interpretation, *i.e.*, the restriction of $\mathcal{I}$ to the atomic concepts and roles of $\mathcal{S}$. Remember that $((2^{\Delta})^n, \leq)$ is a complete lattice. With $\vec{C}^{\mathcal{I}}$ we denote the vector $(C_1^{\mathcal{I}}, \ldots, C_n^{\mathcal{I}})$ and with "$\wedge$" the componentwise intersection on $(2^{\Delta})^n$. Then the following holds:

$$\begin{aligned} \mathit{gfp}(\mathcal{S}^{\sqcap}_{\mathcal{J}}) &= \mathit{lub}\{\vec{O} \mid \vec{O} \leq \mathcal{S}^{\sqcap}_{\mathcal{J}}(\vec{O})\} && (1) \\ &= \mathit{lub}\{\vec{O} \mid \vec{O} \leq \vec{O} \wedge \vec{C}^{\mathcal{I}}\} && (2) \end{aligned}$$

$$
\begin{aligned}
&= \; lub\{\vec{O} \mid \vec{O} \leq \vec{C}^{\mathcal{I}}\} && (3)\\
&= \; lub\{\vec{O} \mid \vec{O} \leq \mathcal{S}^{\equiv}_{\mathcal{J}}(\vec{O})\} && (4)\\
&= \; gfp(\mathcal{S}^{\equiv}_{\mathcal{J}}) && (5)
\end{aligned}
$$

Equations 1 and 5 follow from Proposition 3.1, 2 and 4 by definition of the mappings $\mathcal{S}^{\sqcap}_{\mathcal{J}}$ and $\mathcal{S}^{\equiv}_{\mathcal{J}}$, respectively, and 3 is based on a well known result from set theory, *i.e.*, $A \subseteq B$ if and only if $A \subseteq A \cap B$. □

As a consequence of the preceding proposition, taking the transformation of Schild together with gfp-semantics forces all schema concepts with the same frame-like structure to be identical. For example, if the schema is $\mathcal{S} = \{\mathsf{City} \sqsubseteq \forall\mathsf{name}.\mathsf{String}, \mathsf{Employee} \sqsubseteq \forall\mathsf{name}.\mathsf{String}\}$, cities and employees would be equivalent under gfp-semantics.

Next we consider the transformation $\bar{\mathcal{S}}$. We show that the descriptive models of $\mathcal{S}$ and the gfp-models of $\bar{\mathcal{S}}$ correspond to each other in the sense that (1) every gfp-model of $\bar{\mathcal{S}}$ is a descriptive model of $\mathcal{S}$ and (2) every descriptive model of $\mathcal{S}$ can be turned into a gfp-model of $\bar{\mathcal{S}}$ by choosing the denotation of the additional atomic concepts $\bar{A}_i$ in a suitable manner. The first point is obvious. To see the second point, for an interpretation $\mathcal{I}$ of $\mathcal{S}$ let $\bar{\mathcal{I}}$ denote the interpretation of $\bar{\mathcal{S}}$ defined by $A^{\bar{\mathcal{I}}} := A^{\mathcal{I}}$ and $P^{\bar{\mathcal{I}}} := P^{\mathcal{I}}$ for every concept name A and role name P appearing in $\mathcal{S}$ and $\bar{A}_i^{\bar{\mathcal{I}}} := A_i^{\mathcal{I}}$ for $i \in 1..n$. Then the following holds.

Proposition 3.3 *Let $\mathcal{I}$ be a model of $\mathcal{S}$. Then $\bar{\mathcal{I}}$ is a gfp-model of $\bar{\mathcal{S}}$.*

Proof. Let $\bar{\mathcal{J}}$ denote the atomic interpretation corresponding to $\bar{\mathcal{I}}$. We first show that $(A_1^{\bar{\mathcal{I}}}, \ldots, A_n^{\bar{\mathcal{I}}})$ is a fixpoint of $\bar{\mathcal{S}}_{\bar{\mathcal{J}}}$. To this end we have to show that $A_i^{\bar{\mathcal{I}}} = (\bar{A}_i \sqcap C_i)^{\bar{\mathcal{I}}}$ for $i \in 1..n$. By definition of $\bar{\mathcal{I}}$ this is equivalent to $A_i^{\mathcal{I}} = A_i^{\mathcal{I}} \cap C_i^{\mathcal{I}}$ for $i \in 1..n$. The inclusions $A_i^{\mathcal{I}} \supseteq A_i^{\mathcal{I}} \cap C_i^{\mathcal{I}}$ hold trivially. For the inclusions $A_i^{\mathcal{I}} \subseteq A_i^{\mathcal{I}} \cap C_i^{\mathcal{I}}$ it remains to show that $A_i^{\mathcal{I}} \subseteq C_i^{\mathcal{I}}$ for $i \in 1..n$. But this follows from the fact that $\mathcal{I}$ is a model of $\mathcal{S} = \{A_i \sqsubseteq C_i \mid i \in 1..n\}$.

In order to see that $\bar{\mathcal{I}}$ is a gfp-model observe that for every fixpoint model $\mathcal{I}^*$ extending $\bar{\mathcal{J}}$ it holds that $A_i^{\mathcal{I}^*} = \bar{A}_i^{\bar{\mathcal{J}}} \cap C^{\mathcal{I}^*}$ and therefore $A_i^{\mathcal{I}^*} \subseteq \bar{A}_i^{\bar{\mathcal{J}}}$. But by definition of $\bar{\mathcal{I}}$ we have $\bar{A}_i^{\bar{\mathcal{J}}} = \bar{A}_i^{\bar{\mathcal{I}}} = A_i^{\mathcal{I}} = A_i^{\bar{\mathcal{I}}}$. That is, $A_i^{\mathcal{I}^*} \subseteq A_i^{\bar{\mathcal{I}}}$. Hence, $\mathcal{I}^*$ is a smaller fixpoint than $\bar{\mathcal{I}}$. □

Hence taking the transformation of Nebel has the consequence that descriptive semantics and gfp-semantics coincide, *i.e.*, every conclusion with respect to descriptive semantics is also a conclusion with respect to gfp-semantics and vice versa. But this means that making that transformation and then providing a mechanism for reasoning with respect to gfp-semantics is just a detour of reasoning with respect to descriptive semantics.

Summarizing one can say that adopting lfp- or gfp-semantics for our schema formalism leads either to unacceptable results or is equivalent to descriptive semantics.

This gives additional evidence for our choice to take descriptive semantics for the schema.

3.4 Schema Cycles versus View Cycles

We feel that much of the confusion about the semantics of terminological cycles and many computational problems stem from the mixing of inclusions and definitions. Therefore we propose to make a distinction between the schema, containing only inclusions, and the view taxonomy containing only definitions. These two parts also differ with respect to the concept language and the type of semantics. The axioms in the schema have the role of narrowing down the class of models we consider possible. Therefore, they should be interpreted under descriptive semantics. Also the results presented in this section support this choice.

Differently from the schema, the role of cycles in the view part is to state recursive definitions. For example, if we want to describe the group of individuals that are above Bill in the hierarchy of bosses we can use the definitions "$\mathsf{BillsBosses} \doteq \exists \mathsf{boss}^{-1}.\{\mathsf{BILL}\}$" and "$\mathsf{BillsSuperBosses} \doteq \mathsf{BillsBosses} \sqcup \exists \mathsf{boss}^{-1}.\mathsf{BillsSuperBosses}$." But as argued before, in general this does not yield a definition if we assume descriptive semantics. For a fixed interpretation of BILL and the role $\mathsf{subordinate}$ there may be several ways to interpret $\mathsf{BillsSuperBosses}$ in such a way that the above equality holds. In this example, we only obtain the intended meaning if we assume lfp-semantics. Unfortunately, algorithms for subsumption of views under such semantics are known only for fragments of the concept language defined in Tables 1 and 2.

In this paper, we only deal with cycle-free view taxonomies. In this case all the three types of semantics coincide.

4 Schemas

The schema introduces the concepts and roles of the domain to be modeled and describes their relationships. In this section we first introduce the concept language $\mathcal{SL}$. In $\mathcal{SL}$, we can express the statements most frequent occurring in the declaration of primitive concepts in terminological systems and in the static parts of object-oriented database schemas. Then we investigate two extensions of $\mathcal{SL}$: the language $\mathcal{SL}_{\text{dis}}$, where one can state that two classes are disjoint, and $\mathcal{SL}_{\text{inv}}$, which allows for statements about inverse attributes. We show that reasoning about $\mathcal{SL}$-schemas is easy, while it is hard for the two extensions. The language $\mathcal{SL}$ will also be used in Section 5 as the schema language in our case studies.

4.1 $\mathcal{SL}$-schemas

A schema does not contain definitions, but imposes only necessary conditions on concepts and roles, which are expressed by inclusion axioms.

Basic schema information can be captured if we choose the concept language $\mathcal{SL}$, introduced in (Buchheit et al., 1994), which is defined by the syntax rule

$$D \longrightarrow \ A \mid \forall P.A \mid (\geq 1\, P) \mid (\leq 1\, P).$$

As shown in Section 2, by such schemas we can express elementary type information like domain and codomain of roles, inclusion relationships, and restrictions of the codomain of a role due to restrictions of its domain. Moreover, we can specify a role as necessary (at least one value) or single valued (at most one value). An *$\mathcal{SL}$-schema* is a set of inclusion axioms where all concepts are from $\mathcal{SL}$.

The basic reasoning task for schemas is to determine validity. For $\mathcal{SL}$-schemas, this is trivial.

Proposition 4.1 *Every $\mathcal{SL}$-schema is valid.*

Proof. For a given $\mathcal{SL}$-schema $\mathcal{S}$ we construct an interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ as follows. Let $\Delta^\mathcal{I}$ be the set of individual names in our language (we assume that there is at least one). For any concept name A, role name P and individual a we define $A^\mathcal{I} := \Delta^\mathcal{I}$, $P^\mathcal{I} := \{(a,a) \mid a \in \Delta^\mathcal{I}\}$, and $a^\mathcal{I} := a$. It is easy to check that $\mathcal{I}$ satisfies every axiom in $\mathcal{S}$ and that $A^\mathcal{I} \neq \emptyset$ for every concept name A. □

It is also interesting to determine the subsumption relations between schema names that are entailed by a schema. An $\mathcal{SL}$-schema may entail non-obvious subsumptions. For example, from the schema

$$\{\mathsf{salary} \sqsubseteq \mathsf{Person} \times \mathsf{Salary},\ \mathsf{Employee} \sqsubseteq (\geq 1\, \mathsf{salary})\}$$

it follows that every employee is a person. We will call a schema $\mathcal{S}$ *isa-complete* if all implicit subsumption relations of this kind also appear explicitly, *i.e.*, if $\mathcal{S}$ contains the axiom $A_1 \sqsubseteq A_2$ whenever there are in $\mathcal{S}$ axioms $P \sqsubseteq A_2 \times B$ and $A_1 \sqsubseteq (\geq 1\, P)$.

General Assumption. *In the rest of the section we assume that all schemas are isa-complete.*

For a schema $\mathcal{S}$, we write $A \preceq_\mathcal{S} B$ if there are schema names $A = A_0, A_1 \ldots, A_n = B$ such that $\mathcal{S}$ contains the axioms $A_{i-1} \sqsubseteq A_i$ for $i \in 1..n$. In other words, "$\preceq_\mathcal{S}$" is the transitive, reflexive closure of the explicit subsumption statements in $\mathcal{S}$.

Proposition 4.2 *Let $\mathcal{S}$ be an $\mathcal{SL}$-schema and A, B be schema names. Then $\mathcal{A} \sqsubseteq_\mathcal{S} B$ if and only if $A \preceq_\mathcal{S} B$.*

Proof. This is a consequence of Proposition 4.15 on page 25. □

We conclude that subsumption of schema names w.r.t. an $\mathcal{SL}$-schema can be computed in polynomial time.

4.2 Schemas with Disjointness Axioms

In many modeling tasks one would like to state that certain classes are disjoint. Considering the company environment in Figure 2, one might want to require employees, cities, departments *etc.* not to have common instances. This can be achieved by *disjointness axioms* of the form

$$A \sqsubseteq \neg B.$$

The schema language obtained from $\mathcal{SL}$ by adding negation of concept names $\neg B$ is called $\mathcal{SL}_{\mathrm{dis}}$.

In contrast to $\mathcal{SL}$, not every $\mathcal{SL}_{\mathrm{dis}}$-schema is valid. We say that a schema $\mathcal{S}$ is *locally valid* if every schema name is interpreted as a nonempty set by some model of $\mathcal{S}$. The following proposition says that validity of $\mathcal{SL}_{\mathrm{dis}}$-schemas can be decided by considering one concept at a time.

Proposition 4.3 *An $\mathcal{SL}_{\mathrm{dis}}$-schema is valid if and only if it is locally valid.*

Proof. (See Appendix) □

4.2.1 Validity of $\mathcal{SL}_{\mathrm{dis}}$-schemas is co-NP-hard

We show that deciding the validity of $\mathcal{SL}_{\mathrm{dis}}$-schemas is co-NP-hard. The proof consists in a reduction of the satisfiability problem for concepts of the language $\mathcal{ALE}$ (see Schmidt-Schauß & Smolka, 1991), which is defined by the syntax rule

$$C, C' \longrightarrow \ \bot \mid \top \mid A \mid \neg A \mid C \sqcap C' \mid \forall P.C \mid \exists P.C.$$

In (Donini, Hollunder, Lenzerini, Spaccamela, Nardi, & Nutt, 1992) it has been shown that deciding satisfiability of $\mathcal{ALE}$-concepts is co-NP-complete. The intuitive reason for this result is that for an unsatisfiable concept there always exists an unsatisfiability proof of polynomial length. However, the interaction of universal and existential quantifiers may generate an exponential number of Skolem constants, which results in an exponential number of deductions that have to be considered during the search for a proof.

The proof in (Donini et al., 1992) reveals, more specifically, that satisfiability is still co-NP-complete for *restricted* $\mathcal{ALE}$-concepts C, which satisfy the following properties:

1. only one role symbol occurs in C;
2. no concept name other than $\top$ and $\bot$ occurs in C;
3. there is exactly one occurrence of $\bot$ in C;

4. every proper subconcept of C distinct from $\bot$ is satisfiable.

A subconcept is proper if it is a proper substring. The condition that no proper subconcept other than $\bot$ is unsatisfiable implies that a restricted concept has no subconcept of the form $\exists P.\bot$ or $\bot \sqcap D$.

Our proof consists in associating to every restricted $\mathcal{ALE}$-concept C an $\mathcal{SL}_{\mathrm{dis}}$-schema $\mathcal{S}_C$ such that $\mathcal{S}_C$ is valid if and only if C is satisfiable.

Construction 4.4 Let C be a restricted $\mathcal{ALE}$-concept whose only role symbol is Q. Without loss of generality, we assume that $C \neq \bot$. The assumptions imply that C has exactly one subconcept of the form $\forall Q.\bot$. We choose for each subconcept $D \neq \bot$ of C a concept name A_D and for every subconcept $D = \exists Q.D'$ a role symbol P_D. Let $\mathcal{P}_C$ be the set of all such role symbols. Let A^+, A^- be two additional concept names. For every subconcept D of C we enter the following axioms into the schema $\mathcal{S}_C$:

1. $A_D \sqsubseteq A_{D'}$, $A_D \sqsubseteq A_{D''}$, if $D = D' \sqcap D''$;
2. $A_D \sqsubseteq (\geq 1\ P_D)$, $A_D \sqsubseteq \forall P_D.A_{D'}$, if $D = \exists Q.D'$;
3. $A_D \sqsubseteq \forall P.A_{D'}$ for all $P \in \mathcal{P}_C$, if $D = \forall Q.D'$ with $D' \neq \bot$;
4. $A_D \sqsubseteq \forall P.A^+$, $A_D \sqsubseteq \forall P.A^-$, for all $P \in \mathcal{P}_C$, and $A^+ \sqsubseteq \neg A^-$ if $D = \forall Q.\bot$.

The idea underlying our reduction is to "unfold" the concept C into a set of axioms. In this process, conceptually the role Q is imitated by the union of all $P \in \mathcal{P}_C$, universally quantified subconcepts of C are translated into universal quantification over all roles $P \in \mathcal{P}_C$, and existentially quantified subconcepts D are translated into an existential quantification over the role P_D. Thus, the reduction shows that, as in reasoning about $\mathcal{ALE}$-concepts, the interplay between universal and existential quantifiers makes inferences about $\mathcal{SL}_{\mathrm{dis}}$-schemas difficult.

Lemma 4.5 *Let C be a restricted $\mathcal{ALE}$-concept. Then $\mathcal{S}_C$ is valid if and only if C is satisfiable.*

Proof. (See Appendix) □

Theorem 4.6 *Validity of $\mathcal{SL}_{\mathrm{dis}}$-schemas is* co-NP-*hard.*

4.2.2 An Algorithm for Reasoning about $\mathcal{SL}_{\mathrm{dis}}$-schemas

Next we describe an algorithm for deciding the local validity of an $\mathcal{SL}_{\mathrm{dis}}$-schema $\mathcal{S}$. Actually, it is a method to check whether a finite conjunction of schema names is $\mathcal{S}$-satisfiable. From it we can derive as an upper complexity bound that validity can be decided with polynomial space for arbitrary schemas.

Our method consists in constructing for every schema $\mathcal{S}$ a labeled directed graph $\mathcal{G}_{\mathcal{S}}$ such that the validity of $\mathcal{S}$ can be decided by traversing $\mathcal{G}_{\mathcal{S}}$. The size of $\mathcal{G}_{\mathcal{S}}$ is exponential in the size of $\mathcal{S}$, and the portion of $\mathcal{G}_{\mathcal{S}}$ to be explored might also be exponential in the size of $\mathcal{S}$. We obtain our PSPACE result by keeping only a small portion of $\mathcal{G}_{\mathcal{S}}$ in memory at a time.

Let P be a role symbol. We say that P *is necessary on* A if there is an A' with $A \preceq_{\mathcal{S}} A'$ and $A' \sqsubseteq (\geq 1\ P) \in \mathcal{S}$. If P is necessary on A then in any model of $\mathcal{S}$ every instance of A has a P-filler.

We say that $\mathcal{S}$ contains a *P-transition from A to B* (written $A \xrightarrow{P}_{\mathcal{S}} B$) if there is an A' with $A \preceq_{\mathcal{S}} A'$ and $A' \sqsubseteq \forall P.B \in \mathcal{S}$ or if there is a role inclusion $P \sqsubseteq A'' \times B \in \mathcal{S}$. Note that if P is necessary on A then, since $\mathcal{S}$ is isa-complete, it holds that $A \preceq_{\mathcal{S}} A''$. If there is a P-transition from A to B, then in any model of $\mathcal{S}$ every P-filler of an instance of A is an instance of B.

If $\mathcal{C}$ is a set of concept names occurring in $\mathcal{S}$ we define the *range of P on $\mathcal{C}$* as the set

$$\mathit{range}(P, \mathcal{C}) := \{B \mid A \xrightarrow{P}_{\mathcal{S}} B \text{ for some } A \in \mathcal{C}\}.$$

Construction 4.7 For an $\mathcal{SL}_{\mathrm{dis}}$-schema $\mathcal{S}$ the *schema graph* $\mathcal{G}_{\mathcal{S}}$ is defined as follows:

- every set $\mathcal{C}$ of concept names occurring in $\mathcal{S}$ is a *node* of $\mathcal{G}_{\mathcal{S}}$;
- there is an *edge* with label P from $\mathcal{C}$ to $\mathcal{C}'$ if
  - P is necessary on A for some $A \in \mathcal{C}$, and
  - $\mathcal{C}' = \mathit{range}(P, \mathcal{C})$.

A node $\mathcal{C}$ is a *conflict node* if there are A, $B \in \mathcal{C}$ such that $A' \sqsubseteq \neg B' \in \mathcal{S}$ for some A', B' with $A \preceq_{\mathcal{S}} A'$ and $B \preceq_{\mathcal{S}} B'$.

Intuitively, a node $\mathcal{C} = \{A_1, \ldots, A_m\}$ represents the assumption that $A_1, \ldots, A_m$ have a common instance. A conflict node stands for an assumption that contradicts some disjointness axiom in $\mathcal{S}$. If there is an edge with label P from $\mathcal{C}$ to $\mathcal{C}' = \{B_1, \ldots, B_n\}$, then every common instance of $A_1, \ldots, A_m$ has a P-filler (because P is necessary on some A_i), which is a common instance of $B_1, \ldots, B_n$ (because $\mathcal{C}'$ is the range of P on $\mathcal{C}$). The set $\mathcal{C}'$ might be the empty set. But then there is no edge going out of $\mathcal{C}'$, since a role P can be necessary only on concepts. The graph $\mathcal{G}_{\mathcal{S}}$ will be used to check whether the assumption that $A_1, \ldots, A_m$ have a common instance leads to a contradiction.

Lemma 4.8 *Let $\mathcal{S}$ be an $\mathcal{SL}_{\rm dis}$-schema and $\mathcal{C} = \{A_1, \ldots, A_m\}$. Then $A_1 \sqcap \ldots \sqcap A_m$ is $\mathcal{S}$-unsatisfiable if and only if there is no path in $\mathcal{G}_\mathcal{S}$ from $\mathcal{C}$ to a conflict node.*

Proof. (See Appendix) □

By Lemma 4.8, $A_1 \sqcap \ldots \sqcap A_m$ is not $\mathcal{S}$-satisfiable if and only if there is a path in $\mathcal{G}_\mathcal{S}$ from $\mathcal{C} = \{A_1, \ldots, A_m\}$ to some conflict node $\mathcal{C}'$. Such a path can be detected nondeterministically as follows: for a given node we construct a sequence of successor nodes until we have reached a conflict node. A successor node can be computed if the current node and the schema are known. Both can be stored using no more than polynomial space. Thus, there exists a nondeterministic polynomial space algorithm. By Savitch's Theorem (see Hopcroft & Ullman, 1969), it can be transformed into a deterministic polynomial space algorithm. This proves the following theorem:

Theorem 4.9 *There is a* PSPACE *algorithm that decides for an $\mathcal{SL}_{\rm dis}$-schema $\mathcal{S}$ and schema names $A_1, \ldots, A_m$ whether the conjunction $A_1 \sqcap \ldots \sqcap A_m$ is $\mathcal{S}$-satisfiable.*

Combining Theorem 4.9 with the preceding hardness result leads to the following complexity bounds.

Corollary 4.10 *The validity problem for $\mathcal{SL}_{\rm dis}$-schemas is in* PSPACE *and* co-NP-*hard.*

4.2.3 Cycles in $\mathcal{SL}_{\rm dis}$-Schemas

In Section 3 we introduced a general notion of terminological cycles for arbitrary schemas without role inclusions. In this section we refine this notion for $\mathcal{SL}_{\rm dis}$-schemas and adopt it also to role inclusions. Then we identify a class of cycles that increases the complexity of reasoning about $\mathcal{SL}_{\rm dis}$-schemas. To this end, we extend the dependency graph in two directions. First, we add edges coming from role inclusions and second, we mark the edges in order to classify the cycles.

Role inclusions may give rise to terminological cycles. To see this, note that an axiom of the form $P \sqsubseteq A_1 \times A_2$ is equivalent to the two axioms $(\geq 1\, P) \sqsubseteq A_1, \top \sqsubseteq \forall P.A_2$.[6] Thus, a role inclusion $P \sqsubseteq A_1 \times A_2$ leads to two kinds of additional edges. There is an edge from A to A_2 for every concept name A, since $A \sqsubseteq \top$ and $\top \sqsubseteq \forall P.A_2$ hold. There is also an edge from A to A_1 for every axiom $A \sqsubseteq (\geq 1\, P)$, since $(\geq 1\, P) \sqsubseteq A_1$ holds.

[6]Two sets of axioms are equivalent if they have the same models. An interpretation $\mathcal{I}$ satisfies an inclusion $C \sqsubseteq D$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$.

We want to distinguish between different classes of cycles and clarify their influence on the complexity of inferences. Some cycles are computationally harmless. For example, the schema $\mathcal{S} = \{A_1 \sqsubseteq A_2, A_2 \sqsubseteq A_1\}$ is cyclic, but in every model of $\mathcal{S}$, A_1 and A_2 denote the same set. One can get rid of say A_1 while keeping essentially the same meaning. We extend the dependency graph definition by using labeled edges. The label indicates the kind of axiom the edge comes from.

Let $\mathcal{S}$ be an $\mathcal{SL}_{\text{dis}}$-schema. We redefine the *dependency graph* $D(\mathcal{S})$ of $\mathcal{S}$ as follows. The nodes are the concept names in $\mathcal{S}$. Let A_1, A_2 be two nodes. There is

- an *isa-edge* from A_1 to A_2 if there is an axiom $A_1 \sqsubseteq A_2$ in $\mathcal{S}$;
- a *some-edge* from A_1 to A_2 if there are axioms $A_1 \sqsubseteq (\geq 1\, P)$ and $P \sqsubseteq A_2 \times A_3$ in $\mathcal{S}$;
- an *all-edge* from A_1 to A_2 if there is an axiom $A_1 \sqsubseteq \forall P.A_2$ in $\mathcal{S}$ or if there is an axiom $P \sqsubseteq A \times A_2$ in $\mathcal{S}$;
- a *neg-edge* from A_1 to A_2 if there is an axiom $A_1 \sqsubseteq \neg A_2$ in $\mathcal{S}$.

Since schemas are assumed to be isa-complete, there is always an isa-edge from A_1 to A_2 if there is a some-edge from A_1 to A_2. We say $\mathcal{S}$ is *cyclic*, if $D(\mathcal{S})$ contains a cycle, and *cycle-free* otherwise. An *all-cycle* is a cycle which contains at least one all-edge and no neg-edge. A schema $\mathcal{S}$ is *all-cycle-free*, if $D(\mathcal{S})$ contains no all-cycle.

So the all-cycle-free schemas are a subset of all schemas and the cycle-free schemas are a subset of the all-cycle-free schemas. Now we want to determine the complexity of reasoning for these subclasses.

Notice that the schema built by Construction 4.4 is always cycle-free. This leads to the following lower bound for validity checking.

Theorem 4.11 *Validity of cycle-free $\mathcal{SL}_{\text{dis}}$-schemas is* co-NP*-hard.*

Now we turn to the upper bound. First notice the correspondence between all-cycles and cyclic chains of P-transitions.

Proposition 4.12 *A schema $\mathcal{S}$ contains an all-cycle iff there is a sequence of transitions $A_0 \xrightarrow{P_0}_{\mathcal{S}} A_1, \ldots, A_k \xrightarrow{P_k}_{\mathcal{S}} A_0$.*

Thus, if $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_n$ is a path in the schema graph $\mathcal{G}_{\mathcal{S}}$ of an all-cycle-free schema $\mathcal{S}$, then any two distinct sets $\mathcal{C}_i$, $\mathcal{C}_j$ on the path are disjoint. Therefore, the length of paths in $\mathcal{G}_{\mathcal{S}}$ is bounded linearly by the number of names occurring in $\mathcal{S}$. Thus, the nondeterministic algorithm of Section 4.2.2 that follows a path issuing from $\{A_1, \ldots, A_m\}$ until it reaches a conflict node can be run in polynomial time. This gives the following result.

Theorem 4.13 *Let $\mathcal{S}$ be an all-cycle-free $\mathcal{SL}_{\mathrm{dis}}$-schema. Then deciding whether a conjunction $A_1 \sqcap \ldots \sqcap A_m$ of schema names is $\mathcal{S}$-satisfiable can be done in nondeterministic polynomial time.*

Combining this theorem with the hardness result of Theorem 4.6 leads to the following complexity bound.

Corollary 4.14 *The validity problem for all-cycle-free $\mathcal{SL}_{\mathrm{dis}}$-schemas is* co-NP-*complete.*

4.2.4 Subsumption in $\mathcal{SL}_{\mathrm{dis}}$

Deciding subsumption of schema names with respect to an $\mathcal{SL}_{\mathrm{dis}}$-schema $\mathcal{S}$ cannot be easier than checking satisfiability: $A_1 \sqcap \ldots \sqcap A_m$ is $\mathcal{S}$-unsatisfiable iff for any name B not occurring in $\mathcal{S}$ we have $A_1 \sqcap \ldots \sqcap A_m \sqsubseteq_{\mathcal{S}} B$. The following proposition shows that the difficulty of subsumption checking stems solely from the difficulty of checking satisfiability and that for satisfiable concepts $\mathcal{S}$-subsumption is captured completely by the relation "$\preceq_{\mathcal{S}}$".

Proposition 4.15 *Let $\mathcal{S}$ be an $\mathcal{SL}_{\mathrm{dis}}$-schema and A, $A_1, \ldots, A_m$, be schema names. Suppose that $A_1 \sqcap \ldots \sqcap A_m$ is $\mathcal{S}$-satisfiable. Then $A_1 \sqcap \ldots \sqcap A_m \sqsubseteq_{\mathcal{S}} A$ if and only if there is an A_i such that $A_i \preceq_{\mathcal{S}} A$.*

Proof. Obviously, if $A_i \preceq_{\mathcal{S}} A$, then $A_i \sqsubseteq_{\mathcal{S}} A$ and thus $A_1 \sqcap \ldots \sqcap A_m \sqsubseteq_{\mathcal{S}} A$.

If $A_1 \sqcap \ldots \sqcap A_m$ is $\mathcal{S}$-satisfiable, then the interpretation $\mathcal{I}$ constructed in the proof of Lemma 4.8 is a model of $\mathcal{S}$ with $\mathcal{C} := \{A_1, \ldots, A_m\} \in A_1^{\mathcal{I}} \cap \ldots \cap A_m^{\mathcal{I}}$. If there is no A_i with $A_i \preceq_{\mathcal{S}} A$, then $\mathcal{C} \notin A^{\mathcal{I}}$. Hence, $A_1 \sqcap \ldots \sqcap A_m$ is not $\mathcal{S}$-subsumed by A. □

4.2.5 Dichotomic Schemas

We now investigate a restricted class of $\mathcal{SL}_{\mathrm{dis}}$-schemas that allow for polynomial time reasoning. We facilitate our presentation by assuming that schemas come in a normal form.

A schema $\mathcal{S}$ is *normal* if for every P occurring in $\mathcal{S}$ we have:

- $\mathcal{S}$ contains exactly one axiom of the form $P \sqsubseteq A \times B$;
- if $A' \sqsubseteq (\geq 1\, P) \in \mathcal{S}$, $A' \sqsubseteq (\leq 1\, P) \in \mathcal{S}$, or $A' \sqsubseteq \forall P.B' \in \mathcal{S}$, then $A' \preceq_{\mathcal{S}} A$ and $B' \preceq_{\mathcal{S}} B$.

In normal schemas, the domain and codomain of a role P have a unique name in the schema. We denote them as $dom(P)$ and $cod(P)$, respectively. Moreover, statements about P only involve concepts that are $\mathcal{S}$-subsumed by the domain or codomain of P.

A normal $\mathcal{SL}_{\text{dis}}$-schema $\mathcal{S}$ is *dichotomic* if for every role P we have that $\mathcal{S}$ contains at most one axiom of the form $A \sqsubseteq (\geq 1\ P)$, and if so, then $A = dom(P)$. Dichotomic schemas owe their name to the fact that a role is either necessary on its entire domain or it is not necessary for any concept. Thus, in such a schema, the interaction between universal and existential quantification over roles is limited.

Practical schemas are mostly normal and often also dichotomic. For example, schemas of object-oriented databases usually enforce implicitly this property by distinguishing between set-valued and other attributes. For a set-valued attribute, the set of fillers may be empty, while other attributes always have exactly one filler. The latter correspond to necessary, the former to non-necessary roles.

We will show that for dichotomic schemas validity can be decided in polynomial time. For any dichotomic schema $\mathcal{S}$ we construct a directed graph $\mathcal{D}_\mathcal{S}$ such that it suffices to inspect $\mathcal{D}_\mathcal{S}$ in order to decide the satisfiability of concepts. In contrast to $\mathcal{G}_\mathcal{S}$, the size of $\mathcal{D}_\mathcal{S}$ is polynomial in the size of $\mathcal{S}$.

Construction 4.16 For every $\mathcal{SL}_{\text{dis}}$-schema S the *dichotomic schema graph* $\mathcal{D}_\mathcal{S}$ is defined as follows:

- every nonempty finite set $\mathcal{C}$ of concept names with $|\mathcal{C}| \leq 2$ is a *node* of $\mathcal{D}_\mathcal{S}$;
- there is an *edge* with label P from $\{A, B\}$ to $\{A', B'\}$ if
  - P is necessary on $dom(P)$, and
  - $A \xrightarrow{P}_{\mathcal{S}} B$ and $A' \xrightarrow{P}_{\mathcal{S}} B'$.

A node $\{A, B\}$ is a *conflict node* if there are A', B' with $A \preceq_\mathcal{S} A'$, $B \preceq_\mathcal{S} B'$ such that $A' \sqsubseteq \neg B' \in \mathcal{S}$.

The intuition underlying $\mathcal{D}_\mathcal{S}$ is similar to the one that led to $\mathcal{G}_\mathcal{S}$. For arbitrary $\mathcal{SL}_{\text{dis}}$-schemas, however, we had to take into account arbitrarily big sets of schema names, while for dichotomic schemas we can concentrate on sets with at most two elements.

Lemma 4.17 *Let $\mathcal{S}$ be a dichotomic schema and $A_1, \ldots, A_m$ be concept names. A conflict node in $\mathcal{G}_\mathcal{S}$ is reachable from $\{A_1, \ldots, A_m\}$ if and only if there are A_i, A_j such that a conflict node in $\mathcal{D}_\mathcal{S}$ is reachable from $\{A_i, A_j\}$.*

Proof. (See Appendix) □

Corollary 4.18 *Let $\mathcal{S}$ be a dichotomic schema and $A_1, \ldots, A_m$ be concept names. Then the following are equivalent:*

1. $A_1 \sqcap \ldots \sqcap A_m$ *is not $\mathcal{S}$-satisfiable;*
2. *there are A_i, A_j such that $A_i \sqcap A_j$ is not $\mathcal{S}$-satisfiable;*
3. *there are A_i, A_j such that a conflict node in $\mathcal{D}_\mathcal{S}$ is reachable from $\{A_i, A_j\}$.*

Corollary 4.19 *For dichotomic schemas, satisfiability and subsumption of conjunctions of concept names can be decided in polynomial time.*

4.3 Schemas with Inverse Roles

Often, it would be convenient to make statements about inverses of roles in a schema. For instance, let the role $\mathsf{employs}$ be a shorthand for $\mathsf{works\text{-}for}^{-1}$. Then with the axiom $\mathsf{ResearchDept} \sqsubseteq \forall\mathsf{employs}.\mathsf{Researcher}$, one can express that only researchers are working for a research department.

As seen before, subsumption relations between names occurring in an $\mathcal{SL}$-schema $\mathcal{S}$ are obvious in the sense that $A \sqsubseteq_\mathcal{S} B$ iff $A \preceq_\mathcal{S} B$ (Proposition 4.2), while the difficulty of subsumption w.r.t. $\mathcal{SL}_{\text{dis}}$-schemas stems only from the difficulty of satisfiability checking (Proposition 4.15). However, if we allow for inverse roles in a schema, this may give rise also to implicit subsumption relationships between satisfiable concepts, as we illustrate by an example. Consider the following fragment of the company schema:

$$\begin{array}{rcrcl} \mathcal{S} & = & \{ \quad \mathsf{Researcher} & \sqsubseteq & (\geq 1\, \mathsf{works\text{-}for}), \\ & & \mathsf{Researcher} & \sqsubseteq & \forall\mathsf{works\text{-}for}.\mathsf{ResearchDept}, \\ & & \mathsf{ResearchDept} & \sqsubseteq & \forall\mathsf{employs}.\mathsf{Employee}\}. \end{array}$$

Although the schema is isa-complete and $\mathsf{Researcher} \preceq_\mathcal{S} \mathsf{Employee}$ does *not* hold, it entails that $\mathsf{Researcher}$ is subsumed by $\mathsf{Employee}$. Suppose that JOE is an arbitrary researcher. Then JOE works for some research department, say $\mathsf{D007}$. Since research departments only employ employees, every individual employed by $\mathsf{D007}$ is an employee. Hence, JOE is an employee.

Detecting such implicit subsumption relations might be complex. Let us call $\mathcal{SL}_{\text{inv}}$ the language obtained from $\mathcal{SL}$ by allowing for inverse roles, *i.e.*, $\mathcal{SL}_{\text{inv}}$ contains also concepts of the form $\forall P^{-1}.A$, $(\geq 1\, P^{-1})$ and $(\leq 1\, P^{-1})$. In this subsection, we prove that subsumption of concept names w.r.t. $\mathcal{SL}_{\text{inv}}$-schemas is NP-hard. Moreover, we show that for $\mathcal{SL}_{\text{inv}}$-schemas there is a difference between reasoning w.r.t. all models and reasoning w.r.t. all *finite* models.

4.3.1 Subsumption with respect to $\mathcal{SL}_{\mathrm{inv}}$-schemas is NP-hard

We construct for every restricted $\mathcal{ALE}$-concept C an $\mathcal{SL}_{\mathrm{inv}}$-schema $\bar{\mathcal{S}}_C$ containing two concept names A and A' such that $\bar{\mathcal{S}}_C \models A \sqsubseteq A'$ if and only if C is unsatisfiable.

To specify the construction we inductively define a function $\lambda_C(D)$ that associates to each subconcept D of C the level at which D occurs in C: the concept C itself occurs at depth 0; if $D = D_1 \sqcap D_2$, then $\lambda_C(D_i) := \lambda_C(D)$; if $D = \exists R.D'$ or if $D = \forall R.D'$ then $\lambda_C(D') := \lambda_C(D) + 1$. The level gives us the number of quantifiers in the scope of which D is located.

We obtain $\bar{\mathcal{S}}_C$ by modifying the construction of $\mathcal{S}_C$ in 4.2.1. We do not need the names A^+, A^-, but choose concept names $A_0, \ldots, A_k$, where $k = \lambda_C(\bot)$. Steps 1.) to 3.) remain exactly as they are for $\mathcal{S}_C$. However, instead of the axioms added in step 4.), we enter the following axioms into $\bar{\mathcal{S}}_C$:

4′. $A_D \sqsubseteq \forall P.A_k$ for all $P \in \mathcal{P}_C$, if $D = \forall Q.\bot$;

5′. $A_k \sqsubseteq \forall P^{-1}.A_{k-1}, \ldots, A_1 \sqsubseteq \forall P^{-1}.A_0$ for all $P \in \mathcal{P}_C$.

To explain the underlying intuition, we need some definitions. If $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is an interpretation, we say that a sequence $d_0, \ldots, d_n$ of elements of $\Delta^{\mathcal{I}}$ is a *chain* of length n from d_0 to d_n if there are roles $P_1, \ldots, P_n \in \mathcal{P}_C$ such that $(d_{i-1}, d_i) \in P_i^{\mathcal{I}}$ for $i \in 1..n$. We say that d_n is *reachable* from d_0 if there is a chain from d_0 to d_n.

In Section 4.2.1, for an interpretation $\mathcal{I}$ to be an $\mathcal{S}_C$-model, it is crucial that elements of $A_D^{\mathcal{I}}$, $D = \forall Q.\bot$, do not have P-fillers for any $P \in \mathcal{P}_C$. Now, $\bar{\mathcal{S}}_C$ is defined in such a way that for $\mathcal{I}$ to be an $\bar{\mathcal{S}}_C$-model where A_C is not interpreted as a subset of A_0, $\mathcal{I}$ has to satisfy two properties: (i) there is some element $d \in A_C^{\mathcal{I}}$ (since otherwise $A_C^{\mathcal{I}} = \emptyset$ is a subset of any set), and (ii) no element $d' \in A_D^{\mathcal{I}}$ which is reachable from d by a chain of length $k-1$ has a P-filler for any $P \in \mathcal{P}_C$ (since otherwise the axioms in 4′.) and 5′.) force d to be an element of $A_0^{\mathcal{I}}$). Thus, in both cases it is important that elements of $A_D^{\mathcal{I}}$ do not have any P-fillers.

Lemma 4.20 *Let C be a restricted $\mathcal{ALE}$-concept. Then $\bar{\mathcal{S}}_C \models A_C \sqsubseteq A_0$ if and only if C is unsatisfiable.*

Proof. (See Appendix) □

Theorem 4.21 *Subsumption of concept names with respect to $\mathcal{SL}_{\mathrm{inv}}$-schemas is* NP-*hard.*

Proof. The claim follows by the preceding lemma because unsatisfiability of restricted $\mathcal{ALE}$-concepts is NP-hard (see 4.2.1). □

4.3.2 Finite Model Reasoning

For $\mathcal{SL}_{\text{dis}}$-schemas, it does not make a difference if we define satisfiability or subsumption of concept names with respect to all interpretations or with respect to *finite interpretations*, *i.e.*, interpretations with finite domains.

However, in an $\mathcal{SL}_{\text{inv}}$-schema $\mathcal{S}$ there may be concepts A, B such that $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ for all finite models of $\mathcal{S}$, but not for all models. To see this, observe that $\mathcal{S}$ may require every model to interpret A as a set of cardinality at least as great as the cardinality of B. For example, consider the schema

$$\begin{aligned} \mathcal{S} = \{ \quad \mathsf{Manager} &\sqsubseteq (\geq 1\, \mathsf{boss}^{-1}), \\ \mathsf{Manager} &\sqsubseteq \forall \mathsf{boss}^{-1}.\mathsf{Employee}, \\ \mathsf{Employee} &\sqsubseteq (\leq 1\, \mathsf{boss})\}, \end{aligned}$$

saying that every manager is the boss of at least one person, and that all persons a manager is the boss of are employees. Moreover, every employee has at most one boss. As a consequence, in any model one can map injectively every manager to some employee. Thus, in any finite model, the number of managers does not exceed the number of employees. If we add the axiom $\mathsf{Employee} \sqsubseteq \mathsf{Manager}$, then for any finite model $\mathcal{I}$ we have $\mathsf{Employee}^{\mathcal{I}} = \mathsf{Manager}^{\mathcal{I}}$. This need not be true in infinite models. Consequently, in every finite model $\mathcal{I}$ of $\mathcal{S}' := \mathcal{S} \cup \{\mathsf{Employee} \sqsubseteq \mathsf{Manager}\}$ we have $\mathsf{Manager}^{\mathcal{I}} \subseteq \mathsf{Employee}^{\mathcal{I}}$, which need not hold in an infinite model. Reasoning about schemas w.r.t. finite models has been investigated in (Calvanese & Lenzerini, 1994; Calvanese, Lenzerini, & Nardi, 1994). We will not study finite model reasoning in this paper, since this requires different techniques.

5 Case Studies

In this section, we study some illustrative examples that show the advantages of the architecture we propose. We extend three systems by the language $\mathcal{SL}$ for cyclic schemas. The view languages are derived from three implemented systems described in the literature, namely KRIS (Baader & Hollunder, 1991), CLASSIC (Borgida et al., 1989) and CONCEPTBASE (Jarke, Gallersdoerfer, Jeusfeld, Staudt, & Eherer, 1995).

For the extended systems, we study the complexity of the reasoning services, where, in particular, we obtain the following results:

- combined complexity is not increased by the presence of terminological cycles in the schema;
- reasoning with respect to schema complexity is always tractable.

The second result can intuitively be interpreted as stating that in all cases the complexity of inferences is due to the view language alone.

In this section, we assume that the view taxonomy is cycle-free. We also assume that no view names occur in the right-hand sides of view definitions or in the world description. In fact, this can be achieved by iteratively substituting every view name with its definition, which is possible because of our acyclicity assumption (see Nebel, (1990b) for a discussion of this substitution and its complexity). In practice, this is equivalent to assuming that the view taxonomy is empty. Therefore, from this point on we do not take into account the view taxonomy, and we assume the knowledge base Σ to be simply a pair $\langle \mathcal{S}, \mathcal{W} \rangle$.

The three systems stand for three different design paradigms (see Baader and Nutt, (1992)). Thus each case study emphasizes a different aspect of the benefits that can be gained from our proposal.

The system KRIS is built at DFKI and used in several applications as the knowledge representation component (see *e.g.*, Wahlster, André, Finkler, Profitlich, and Rist, (1993)). The designers wanted to provide complete reasoning for a language which is so rich that no polynomial inference procedures are feasible (if P $\neq$ NP). The concept language of KRIS is closed under propositional connectives and it provides universal and existential quantification over roles. For this reason, subsumption and instance checking are PSPACE-hard (Baader & Hollunder, 1991). Since KRIS also provides number restrictions on roles, it is a proper extension of $\mathcal{SL}$. Hence, the aspect in which our architecture goes beyond that of KRIS is that it allows for cycles going through schema concepts. We show that, for this extension, both view subsumption and instance checking remain in PSPACE. As a byproduct, we give for the first time a proof that instance checking in KRIS (without cyclic schemas) is in PSPACE.

The CLASSIC system has been developed at AT&T Bell Laboratories, where it is applied in a number of projects (see *e.g.*, Wright, Weixelbaum, Vesonder, Brown, Palmer, Berman, and Moore, (1993)). Its language has been designed with the goal to be as expressive as possible while still admitting polynomial time inferences. So it provides intersection of concepts but no union, universal but not existential quantification over roles, and number restrictions over roles but no intersection of roles, since each of these combinations is known to make reasoning NP-hard (Donini, Lenzerini, Nardi, & Nutt, 1991). Similarly to KRIS, the concept language of CLASSIC extends $\mathcal{SL}$, so that the novelty of our approach is in the cycles in the schema. Here we show that in the extended architecture view subsumption and instance checking can be computed in polynomial time. As a special case we give a proof for the polynomiality of CLASSIC that uses a technique different from the one in (Borgida & Patel-Schneider, 1994).

ConceptBase is a deductive object-oriented database system, which is under development at the University of Aachen. In ConceptBase there is a distinction between classes in the schema and classes that define queries. The former correspond to schema concepts and the latter to view concepts in our framework. Class descriptions in ConceptBase consist of two parts: a structural part, where essentially

$$
\begin{array}{lll}
V_1 & = & \mathsf{Researcher} \sqcap \forall \mathsf{has\text{-}degree.Engineering} \\
V_2 & = & \mathsf{Employee} \sqcap \exists \mathsf{has\text{-}degree.Engineering}
\end{array}
$$

Figure 3: $\mathcal{ALCNR}$-Views

isa-relationships and restrictions on attributes are expressed, and a nonstructural part where additional membership conditions can be expressed with first-order formulas. The language in which the structural part of schema classes is specified coincides with $\mathcal{SL}$. The view language we consider has been proposed in (Buchheit et al., 1994) as an extension of the structural part of query classes. In this case study the view language is not an extension of the schema language as in the previous cases. Instead, each of the two offers constructs that do not occur in the other. The design is such that all inferences are polynomial while combining the constructs in the schema and the view language would make reasoning intractable. Therefore, this case study illustrates that with our architecture one can reach a better compromise between expressivity and tractability than with the homogeneous traditional one.

5.1 The Language of KRIS as View Language

The system KRIS provides as its basic language $\mathcal{ALCNR}$, which is defined by the following syntax rules:

$$
\begin{array}{rcl}
C, D & \longrightarrow & A \mid \top \mid \bot \mid C \sqcap D \mid C \sqcup D \mid \neg C \mid \forall R.C \mid \exists R.C \mid (\geq n\, R) \mid (\leq n\, R) \\
R & \longrightarrow & P_1 \sqcap \cdots \sqcap P_k
\end{array}
$$

The language $\mathcal{ALCNR}$, first introduced in (Hollunder, Nutt, & Schmidt-Schauß, 1990), allows one to express intersection, union, and complement of concepts, universal and existential quantification on roles, number restrictions and role conjunction. Figure 3 contains some examples of $\mathcal{ALCNR}$-views. View V_1 denotes the researchers only having degrees in engineering. View V_2 denotes the employees who have a degree in engineering. Without any schema information there is no subsumption relationship between V_1 and V_2. But given the schema of Figure 2, (1) every researcher is an employee, and (2) every researcher has some degree. Hence, view V_1 is subsumed by V_2. An $\mathcal{ALCNR}$-knowledge base is a pair $\langle \mathcal{S}, \mathcal{W} \rangle$, where $\mathcal{S}$ is an $\mathcal{SL}$-schema and $\mathcal{W}$ is an $\mathcal{ALCNR}$-world description, respectively. Throughout Section 5.1, by knowledge base we always mean $\mathcal{ALCNR}$-knowledge base.

We study the complexity of reasoning for both view subsumption $C \sqsubseteq_{\mathcal{S}} D$ and instance checking $\langle \mathcal{S}, \mathcal{W} \rangle \models a\colon D$, where C, D are $\mathcal{ALCNR}$-concepts. For the complexity analysis, we assume that numbers in number restrictions are represented with unary encoding (*i.e.*, a number n is represented as a string of n equal symbols). Alternatively, the analysis holds also if numbers cannot exceed a constant bound.

Reasoning in $\mathcal{ALCNR}$-knowledge bases can be done using a calculus similar to the tableaux calculus with equality in first-order logic. Schmidt-Schauß and Smolka (1991) first used such a calculus for the language $\mathcal{ALC}$ which is a sublanguage of $\mathcal{ALCNR}$ that allows neither to express number restrictions nor role conjunction. In the next subsection we introduce the calculus for $\mathcal{ALCNR}$, and in the following one we study the complexity of reasoning by means of the calculus.

5.1.1 Completion Rules of the $\mathcal{ALCNR}$-Calculus

Our calculus operates on knowledge bases; it starts from the given knowledge base, called the *initial* knowledge base, and adds assertions to the world description by suitable *completion rules*. Before describing how assertions are added, we need to expand the syntax and the definitions in a suitable way.

We assume that there exists an alphabet of new individuals, which are denoted by the letters x, y, z, and w, possibly with subscript. Individuals initially present in the knowledge base are called *old individuals*. We use the term *individual* for old and new individuals, and use s, t, u to denote individuals. Unlike the old individuals, which are always interpreted as different elements (recall the Unique Name Assumption in Section 2.1), two (or more) new individuals might be interpreted as the same element; to enforce a different interpretation for two individuals s and t, we add the following new type of assertion in the world description:

$$s \neq t$$

Formally, let $\mathcal{I}$ be an interpretation: We say that $\mathcal{I}$ *satisfies* the assertion $s \neq t$ if $s^{\mathcal{I}} \neq t^{\mathcal{I}}$. The definition of a model remains the same.

To make the interpretation of old and new individuals homogeneous, we drop the UNA from the definition of interpretation of old individuals, and we assume that a world description contains the assertion $a \neq b$ for every pair a, b of old individuals appearing in $\mathcal{W}$.

The following proposition is an immediate consequence of the above definitions.

Proposition 5.1 *Let C, D be $\mathcal{ALCNR}$-concepts, let $\langle \mathcal{S}, \mathcal{W} \rangle$ be an $\mathcal{ALCNR}$-knowledge base, x a new individual and a an old one. Then:*

1. *$C \sqsubseteq_{\mathcal{S}} D$ if and only if the knowledge base $\langle \mathcal{S}, \{x\colon C \sqcap \neg D\} \rangle$ is unsatisfiable.*
2. *$\langle \mathcal{S}, \mathcal{W} \rangle \models a\colon D$ if and only if the knowledge base $\langle \mathcal{S}, \mathcal{W} \cup \{a\colon \neg D\} \rangle$ is unsatisfiable.*

S1: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{s\colon A,\ t\colon B\} \cup \mathcal{W} \rangle$

if 1. sPt is in $\mathcal{W}$, and
2. $P \sqsubseteq A \times B$ is in $\mathcal{S}$

S2: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{sPy\} \cup \mathcal{W} \rangle$

if 1. $s\colon \forall P.C$ is in $\mathcal{W}$,
2. $s\colon A$ is in $\mathcal{W}$,
3. $A \sqsubseteq (\geq 1\, P)$ is in $\mathcal{S}$, and
4. y is a new individual, and
5. there is no t such that sPt is in $\mathcal{W}$

S3: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{t\colon B\} \cup \mathcal{W} \rangle$

if 1. $s\colon A$ is in $\mathcal{W}$,
2. sPt is in $\mathcal{W}$, and
3. $A \sqsubseteq \forall P.B$ is in $\mathcal{S}$

S4: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{s\colon B\} \cup \mathcal{W} \rangle$

if 1. $s\colon A$ is in $\mathcal{W}$, and
2. $A \sqsubseteq B$ is in $\mathcal{S}$

S5: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{s\colon (\leq 1\, P)\} \cup \mathcal{W} \rangle$

if 1. $s\colon A$ is in $\mathcal{W}$, and
2. $A \sqsubseteq (\leq 1\, P)$ is in $\mathcal{S}$

Figure 4: The schema rules for $\mathcal{ALCNR}$

Concepts are assumed to be in *negation normal form*, *i.e.*, the only complements they contain are of the form $\neg A$, where A is a concept name. Arbitrary $\mathcal{ALCNR}$-concepts can be rewritten in linear time into equivalent concepts in negation normal form (Donini et al., 1991).

The $\mathcal{ALCNR}$*-calculus* is described by a set of $\mathcal{ALCNR}$*-completion rules*, which are divided into two subsets, the *schema rules* and the *view rules*. If it is clear from the context, we omit the prefix $\mathcal{ALCNR}$. Completion rules add assertions to a world description $\mathcal{W}$ of a knowledge base $\langle \mathcal{S}, \mathcal{W} \rangle$ until either a contradiction is generated or the knowledge base is recognized to be satisfiable.

The schema rules are presented in Figure 4. A completion rule is said to be *applicable* to a knowledge base Σ if Σ satisfies the conditions associated with the rule and if Σ is altered when transformed according to the rule. The second requirement is needed to ensure termination of our calculus. As an example, Rule S1 is applicable to $\langle \mathcal{S}, \mathcal{W} \rangle$ if sPt is in $\mathcal{W}$, $P \sqsubseteq A \times B$ is in $\mathcal{S}$, *and* if $s\colon A$ and $t\colon B$ are not both in $\mathcal{W}$.

Note that the schema rules treat the axioms of the form $A \sqsubseteq (\geq 1\ P)$ and $A \sqsubseteq \forall P.C$ differently from the others: The corresponding rules (S2 and S3) do not add the right-hand side of the axiom to $\mathcal{W}$, but only the logical consequences of the axiom. In this way, schema rules never add to a world description assertions of the form $s\colon \forall P.C$ or $s\colon (\geq\ 1\ P)$; this is done for termination and complexity

V1: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{s\colon C_1,\ s\colon C_2\} \cup \mathcal{W} \rangle$

if 1. $s\colon C_1 \sqcap C_2$ is in $\mathcal{W}$

V2: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{s\colon D\} \cup \mathcal{W} \rangle$

if 1. $s\colon C_1 \sqcup C_2$ is in $\mathcal{W}$,
2. neither $s\colon C_1$ nor $s\colon C_2$ is in $\mathcal{W}$, and
3. $D = C_1$ or $D = C_2$

V3: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{t\colon C\} \cup \mathcal{W} \rangle$

if 1. $s\colon \forall R.C$ is in $\mathcal{W}$, and
2. t is an R-successor of s

V4: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{sP_1y, \ldots, sP_ky,\ t\colon C\} \cup \mathcal{W} \rangle$

if 1. $s\colon \exists R.C$ is in $\mathcal{W}$,
2. $R = P_1 \sqcap \ldots \sqcap P_k$, and
3. y is a new individual,
4. there is no t such that t is an R-successor of s in $\mathcal{W}$ and $t\colon C$ is in $\mathcal{W}$

V5: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{sP_1y_i, \ldots, sP_ky_i \mid i \in 1..n\} \cup \{y_i \neq y_j \mid i, j \in 1..n, i \neq j\} \cup \mathcal{W} \rangle$

if 1. $s\colon (\geq n\, R)$ is in $\mathcal{W}$,
2. $R = P_1 \sqcap \ldots \sqcap P_k$, and
3. $y_1, \ldots, y_n$ are new individuals,
4. there do not exist n R-successors of s in $\mathcal{W}$

V6: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \mathcal{W}[y/t] \rangle$

if 1. $s\colon (\leq n\, R)$ is in $\mathcal{W}$,
2. s has more than n R-successors in $\mathcal{W}$, and
3. y, t are two R-successors of s which are not separated

Figure 5: The view rules for $\mathcal{ALCNR}$

considerations (see Section 5.1.2).

Before providing the view rules, we need some additional definitions, which are related to considering individuals in a world description as nodes in a graph, and constraints sPt as labeled arcs in this graph. Let $\mathcal{W}$ be a world description and $R = P_1 \sqcap \ldots \sqcap P_k$ $(k \geq 1)$ be a role. We say that t is an *R-successor of s in $\mathcal{W}$* if $sP_1t, \ldots, sP_kt$ are in $\mathcal{W}$. We say that t is a *direct successor of s in $\mathcal{W}$* if for some role R, the individual t is an R-successor of s. If $\mathcal{W}$ is clear from the context we simply say that t is an R-successor or a direct successor of t. Moreover, we call *successor* the transitive closure of the relation "direct successor".

We say that s and t are *separated in $\mathcal{W}$* if the assertion $s \neq t$ is in $\mathcal{W}$.

Let $\mathcal{W}$ be a world description, x a new individual, and s an individual; with $\mathcal{W}[x/s]$ we denote the world description obtained by replacing each occurrence of x in $\mathcal{W}$ by s (observe that we never replace an old individual).

The *view rules* are presented in Figure 5. We call the rules V2 and V6 *non-deterministic* rules, since they can be applied in different ways to the same world description. All the other rules are called *deterministic* rules. Moreover, we call the rules S2, V4 and V5 *generating* rules, since they introduce new individuals into the world description. All other rules are called *nongenerating*.

If Σ, Σ' are two $\mathcal{ALCNR}$-knowledge bases, then Σ' is said to be *directly derived* from Σ if it is obtained from Σ by the application of an $\mathcal{ALCNR}$-completion rule, and Σ' is said to be *derived from* Σ if it is obtained from Σ by a sequence of applications. A knowledge base is *complete* if no completion rule applies to it. Any complete knowledge base derived from Σ is called a *completion of* Σ.

We now prove some properties of the knowledge bases obtained by the completion rules. It can be proved by induction that the "successor" relation restricted to new individuals forms a tree. More formally:

Proposition 5.2 *In any knowledge base derived from an initial one by the completion rules, no new individual is a direct successor of two different individuals.*

It can be shown that both schema and view rules do not add unnecessary contradictions; that is, starting from a satisfiable knowledge base there is always a way of applying the rules which leads to a satisfiable knowledge base again (multiple ways of applying rules are possible, since the rules V2 and V6 are nondeterministic).

Theorem 5.3 (Invariance) *Let Σ be an $\mathcal{ALCNR}$-knowledge base.*

1. *Let Σ' be directly derived from Σ. If Σ' is satisfiable then Σ is satisfiable.*
2. *Conversely, if Σ is satisfiable and a rule is applicable to Σ, then there exists a satisfiable knowledge base Σ' directly derived from Σ using that rule.*

The proof is mainly a rephrasing of the soundness of tableaux rules in first-order logic. A similar theorem was proved in (Buchheit et al., 1993) with $\mathcal{ALCNR}$ as a language for expressing schema axioms between concepts (*i.e.*, statements of the form $C \sqsubseteq D$). The only kind of schema statements not considered in the cited paper is $P \sqsubseteq A \times B$, whose corresponding rule is obviously sound.

We call *clash* a set of assertions of the following form:

- $\{s\colon \bot\}$
- $\{s\colon A,\ s\colon \neg A\}$, where A is a concept name.
- $\{s\colon (\leq n\, R)\} \cup \{sP_1t_i, \ldots, sP_kt_i \mid i \in 1..n+1\}$
  $\cup \{t_i \neq t_j \mid i, j \in 1..n+1, i \neq j\}$,
  where $R = P_1 \sqcap \ldots \sqcap P_k$.

A clash is evidently an unsatisfiable set of assertions, hence any world description containing a clash is obviously unsatisfiable. The last case represents the situation in which it is asserted that an individual has at most n R-successors, and at the same time it has more than n R-successors, none of which can be identified with another, because the successors are pairwise separated.

A knowledge base $\Sigma = \langle \mathcal{S}, \mathcal{W} \rangle$ contains a clash if $\mathcal{W}$ contains a clash. Whenever a knowledge base Σ contains a clash, it is obviously unsatisfiable. From Theorem 5.3, we know that if Σ is satisfiable then there exists a complete knowledge base derived from Σ which contains no clash. We prove now that a complete clash-free knowledge base is always satisfiable. To this end, we define the following particular interpretation.

Given a complete knowledge base $\Sigma = \langle \mathcal{S}, \mathcal{W} \rangle$, we define the *canonical interpretation* $\mathcal{I}_\Sigma$ as follows:

$$
\begin{array}{rcl}
\Delta^{\mathcal{I}_\Sigma} & := & \{s \mid s \text{ is an individual in } \mathcal{W}\} \cup \{u\} \\
s^{\mathcal{I}_\Sigma} & := & s \\
A^{\mathcal{I}_\Sigma} & := & \{s \mid s\colon A \text{ is in } \mathcal{W}\} \cup \{u\} \\
P^{\mathcal{I}_\Sigma} & := & \{(s,t) \mid sPt \text{ is in } \mathcal{W}\} \cup \{(u,u)\} \\
 & & \cup \{(s,u) \mid \text{there is no } sPt \text{ in } \mathcal{W}, \text{ but for some } A, \\
 & & \qquad\qquad s\colon A \text{ is in } \mathcal{W} \text{ and } A \sqsubseteq (\geq 1\, P) \text{ is in } \mathcal{S}\},
\end{array}
$$

where u is a new individual not appearing in $\mathcal{W}$. Note that the canonical interpretation uses all the individuals of the knowledge base, plus the special individual u which appears in the interpretation of every primitive concept and is related to itself by every role P. The purpose of this special individual is to satisfy all axioms $A \sqsubseteq (\geq 1 P)$ for those individuals s such that $s\colon A$ is in $\mathcal{W}$, but having no P-successor in $\mathcal{W}$.

Proposition 5.4 *A complete, clash-free $\mathcal{ALCNR}$-knowledge base is satisfiable.*

Proof. (See Appendix) $\square$

The above theorem shows that if the calculus reaches a complete knowledge base Σ without clashes, then Σ is satisfiable, and hence also the initial knowledge base (contained in its completion) is satisfiable. However, to prove that completions are actually reached one should prove that the calculus can be applied so as to terminate. Instead of proving termination by itself, we prove the stronger result that there is a way of applying completion rules such that they always terminate and use just polynomial space.

5.1.2 Termination and Complexity of the $\mathcal{ALCNR}$-Calculus

In this section we will show, that with $\mathcal{SL}$ as schema language and $\mathcal{ALCNR}$ as view language, view subsumption and instance checking are

- PSPACE-complete problems w.r.t. combined complexity and
- PTIME problems w.r.t. schema complexity.

In order to prove this, we have to provide the adequate machinery. The calculus proposed in the previous section requires to compute all the completions of an initial knowledge base Σ. Unfortunately, such completions may be of exponential size w.r.t. the size of Σ, hence that nondeterministic calculus requires exponential space.

To obtain a polynomial-space calculus, it is therefore crucial not to keep an entire complete world description in memory, but to store only small portions at a time. We modify the previous completion rules, so that they build up only a portion of a complete knowledge base and we call the modified rules trace rules.

The *trace rules* consist of the rules presented above, but adding to the application conditions of the generating rules S2, V4, V5 the following further condition:

- For all assertions $tP'z$ in $\mathcal{W}$, either t is a predecessor of s or $s = t$

We label S2$'$, V4$'$, V5$'$ these modified rules.

Let T be a knowledge base obtained from Σ by application of the trace rules. We call T a *trace* of Σ if no trace rule applies to T.

A preliminary definition: Completion rules and trace rules are always applied to a knowledge base $\langle \mathcal{S}, \mathcal{W} \rangle$ because of the presence in $\mathcal{W}$ of a given assertion $s\colon C$, or sPt (condition 1 of all rules). We exploit this property to say that a rule is *applied to* the assertion $s\colon C$, or *applied to* the individual s (instead of saying that it is applied to the knowledge base $\langle \mathcal{S}, \mathcal{W} \rangle$).

The trace rules exhibit the following behavior: Given an individual s, if at least one generating rule is applicable to s, all of s's successors $y_1, \ldots, y_n$ are introduced. Then, after nongenerating rules are applied to s, one new individual y_i is (non-deterministically) chosen, and all successors of y_i are introduced. Unlike normal completion rules, no successor is introduced for any individual different from y_i. Then, one individual is chosen among the successors of y_i, only its successors are added to the world description, and so on.

The reason why we introduce all the successors of the "chosen" individual is the following: For every chosen individual s all direct successors of s must be present simultaneously at some stage of the computation, since the number restrictions force us to identify certain successors. This is important because, when identifying individuals, the constraints imposed on them are combined, which may lead to clashes that otherwise would not have occurred.

Trace rules for $\mathcal{ALC}$ were defined in (Schmidt-Schauß & Smolka, 1991), and were extended to more expressive languages in (Hollunder et al., 1990; Hollunder & Nutt, 1990; Donini et al., 1991). A polynomial-space algorithm that checks the satisfiability of an $\mathcal{ALCNR}$-concept C by generating all complete world descriptions derivable from an initial world description $\{x\colon C\}$, while keeping only one trace in memory at a time, was given in (Donini et al., 1991).

We now adapt those previous results to the presence of a schema. The union of two traces $T_1 = \langle \mathcal{S}, \mathcal{W}_1 \rangle, T_2 = \langle \mathcal{S}, \mathcal{W}_2 \rangle$ is defined as $T_1 \cup T_2 = \langle \mathcal{S}, \mathcal{W}_1 \cup \mathcal{W}_2 \rangle$.

We call *depth* of a concept D, written $depth(D)$, the maximal sequence of nested quantifiers in D (including also number restrictions as quantifiers). More precisely, $depth(A) := depth(\bot) := depth(\top) := 0$, where A is a concept name. Furthermore, $depth(\geq n\, R) := depth(\leq n\, R) := 1$, and $depth(\neg C) := depth(C)$. If D is of one of the forms $D = D_1 \sqcap D_2$ or $D = D_1 \sqcup D_2$ then $depth(D) := \max(depth(D_i))$. If $D = \exists R.D_1$ or $D = \forall R.D_1$ then $depth(D) := depth(D_1) + 1$.

The following proposition collects a number of properties that will be used in the next proposition to state the complexity of view subsumption.

Proposition 5.5 *Let C be an $\mathcal{ALCNR}$-concept, x a new individual, and $\mathcal{W} = \{x\colon C\}$ the corresponding world description; let $\mathcal{S}$ be an $\mathcal{SL}$-schema, and $\Sigma = \langle \mathcal{S}, \mathcal{W} \rangle$ the corresponding knowledge base. Then:*

1. *For every chain of direct successors $x, y_1, \ldots, y_h$ in a knowledge base derived from Σ, if $y_i\colon D$ is in $\mathcal{W}$, and D is a subconcept of C then $depth(D) \leq depth(C) - i$.*

2. *For every chain of direct successors $x, y_1, \ldots, y_h$ in a knowledge base derived from Σ, the length of the chain h is bounded by $|C|$ (the size of C).*

3. *Let N be the maximal number of direct successors of an individual in a trace. Then N is bounded by $|C|$.*

4. *The size of a trace issuing from Σ is polynomially bounded by $|C|$ and linearly bounded by $|\mathcal{S}|$.*

5. *Every completion of Σ can be obtained as the union of finitely many traces.*

6. *Suppose $\Sigma' = \langle \mathcal{S}, \mathcal{W}' \rangle$ is a complete knowledge base derived from Σ and $\mathcal{T}$ is a finite set of traces such that $\Sigma = \bigcup_{T \in \mathcal{T}} T$. Then Σ' contains a clash if and only if some $T \in \mathcal{T}$ contains a clash.*

Proof. (See Appendix) □

Proposition 5.6 *Let $\mathcal{S}$ be an $\mathcal{SL}$-schema, and C, D be $\mathcal{ALCNR}$-concepts. Then:*

1. *Checking $C \sqsubseteq_{\mathcal{S}} D$ can be done in polynomial space w.r.t. $|\mathcal{S}|$, $|C|$, and $|D|$;*

2. *Checking $C \sqsubseteq_{\mathcal{S}} D$ can be done in polynomial time w.r.t. $|\mathcal{S}|$.*

Proof. Combining Proposition 5.5 with Point 1 of Proposition 5.1, one directly proves that view subsumption can be checked in nondeterministic polynomial space w.r.t. combined complexity. In fact, in order to check that $\langle \mathcal{S}, \{x\colon C \sqcap \neg D\}\rangle$ is satisfiable, one generates a clash-free completion keeping in memory only one trace at a time. A deterministic check keeps in memory also the choice points for possible backtracking. Since these points are as many as the assertions of the form $s\colon C_1 \sqcup C_2$ and $s\colon (\leq n\, R)$ (polynomially many in each trace), the entire method is in PSPACE. This proves Point 1 of Proposition 5.6.

The result on schema complexity of view subsumption (Point 2 of Proposition 5.6) is proved in two steps:

First, we prove that both the number of traces in a completion of $\langle \mathcal{S}, \{x\colon C \sqcap \neg D\}\rangle$, and the number of completions depend only on $|C|$ and $|D|$. Observe that the number of traces in a completion depends on the number of applications of generating rules, while the number of different completions depends on the number of choices of applications of nondeterministic rules. All these rules require (condition 1 of all mentioned rules) the presence of assertions which are not added by schema rules, except for an assertion of the form $s\colon (\leq 1\, P)$, which can be introduced by Rule 5. However, this assertion leaves no choice to Rule V6 but leads it to identify all direct P-successors of s. Hence the presence of this assertion does not lead to multiple completions. Moreover, the number of different applications of rule V6 depends on the number of direct successors of an individual. Hence, both the number of traces in a completion and the number of possible completions depend on the number of individuals generated. Since the successor relation restricted to new individuals forms a tree (see Proposition 5.2), the number of individuals can be estimated by N^h, where N is its branching factor—the number of direct successors of an individual—and h is its depth. From Points 2 and 3 in Proposition 5.5, both h and N are bounded by $|C| + |D|$, which proves the claim.

Second, observe that, since the number of traces in a completion and the number of completions depend only on $|C|$ and $|D|$, schema complexity can be computed from the maximal size of a single trace. This size is linear in $|\mathcal{S}|$, as proved in Point 4 of Proposition 5.5. Therefore, schema complexity is in PTIME, and more precisely in $O(|\mathcal{S}|)$. □

We now turn to instance checking. Traces developed so far deal only with satisfiability of concepts (and hence subsumption), and not with instance checking.

The trace algorithm for subsumption of (Schmidt-Schauß & Smolka, 1991) in $\mathcal{ALC}$ was extended by (Baader & Hollunder, 1991) to solve instance checking in $\mathcal{ALC}$-world descriptions. Following similar ideas (see also Donini et al., (1994)), we reformulate the trace calculus given above to instance checking.

A knowledge base is said to be a *precompletion* of another knowledge base Σ if it is obtained from Σ by applying the completion rules only to old individuals, as far as possible.[7] Intuitively, only properties of old individuals are made explicit in precompletions.

Proposition 5.7 *Every precompletion of a knowledge base $\Sigma = \langle \mathcal{S}, \mathcal{W} \rangle$ has polynomial size w.r.t. Σ, and the number of individuals in it does not depend on $|\mathcal{S}|$.*

Proof. (See Appendix) $\square$

Let $\langle \mathcal{S}, \mathcal{W} \rangle$ be a precompletion and x a new individual in $\mathcal{W}$. The *projection of* $\mathcal{W}$ *along* x, denoted as $\mathcal{W}_x$, is the world description formed by all assertions $x\colon C$ that are in $\mathcal{W}$. In other words, $\mathcal{W}_x$ represents all the information about the concepts which x is an instance of, according to $\mathcal{W}$.

Recall that to perform instance checking one has to verify whether a knowledge base Σ is unsatisfiable (Proposition 5.1). The notions of precompletion and projection are useful to perform the latter task. In fact, one can examine each clash-free precompletion $\Sigma' = \langle \mathcal{S}, \mathcal{W}' \rangle$ of Σ, extract the various world descriptions $\mathcal{W}'_x$, and independently check them for unsatisfiability. The correctness of this method follows from the following propositions.

Proposition 5.8 *A knowledge base $\Sigma = \langle \mathcal{S}, \mathcal{W} \rangle$ is satisfiable if and only if there exists a precompletion $\Sigma' = \langle \mathcal{S}, \mathcal{W}' \rangle$ of Σ that is satisfiable.*

Proof. (See Appendix) $\square$

Intuitively, the above proposition proves that one can always build a clash-free completion by first computing a precompletion, and then applying rules to new individuals. The next proposition shows that rules can be applied to new individuals independently for each individual x.

Proposition 5.9 *A precompletion $\Sigma' = \langle \mathcal{S}, \mathcal{W}' \rangle$ of Σ is satisfiable if and only if it is clash-free, and for each new individual x in $\mathcal{W}'$, the knowledge base $\langle \mathcal{S}, \mathcal{W}'_x \rangle$ has a clash-free completion.*

Proof. (See Appendix) $\square$

Proposition 5.10 *Let $\mathcal{S}$ be an $\mathcal{SL}$-schema, $\mathcal{W}$ an $\mathcal{ALCNR}$-world description, a an individual, and D an $\mathcal{ALCNR}$-concept. Then:*

1. *Checking $\langle \mathcal{S}, \mathcal{W} \rangle \models a\colon D$ can be done in polynomial space w.r.t. $|\mathcal{S}|$, $|\mathcal{W}|$, and $|D|$;*
2. *Checking $\langle \mathcal{S}, \mathcal{W} \rangle \models a\colon D$ can be done in polynomial time w.r.t. $|\mathcal{S}|$.*

[7] Notice that this notion of precompletion is different from the one given in (Donini et al., 1994).

Proof. It is now easy to prove Point 1 of Proposition 5.10: To check whether $\langle \mathcal{S}, \mathcal{W} \rangle \models a{:}D$, compute (nondeterministically) a clash-free precompletion Σ' of $\langle \mathcal{S}, \mathcal{W} \cup \{a{:}\neg D\} \rangle$ (this needs polynomial space by Proposition 5.7); then, for each new individual x in Σ', check whether there is a clash-free completion of $\langle \mathcal{S}, \mathcal{W}'_x \rangle$ using the trace calculus developed for satisfiability and subsumption (again, this needs polynomial space); if the nondeterministic method fails, return true, otherwise return false. The deterministic version just keeps track of all backtracking points in applications of nondeterministic rules.

We now turn to the last point of Proposition 5.10, namely, PTIME schema complexity of instance checking. Let $\Sigma = \langle \mathcal{S}, \mathcal{W} \rangle$ be a knowledge base. We prove the point in four steps.

Step 1: The number of individuals in a precompletion does not depend on $|\mathcal{S}|$, by Proposition 5.7. Call this number I.

Step 2: For each assertion of the form $s{:}C_1 \sqcup C_2$, there are two different applications of rule V2 to the assertion; hence there are at most 2^I different applications, for each concept $C_1 \sqcup C_2$ in $\mathcal{W}$. Therefore, the total number of different applications of rule V2 is $O(|\mathcal{W}| \cdot 2^I)$, which does not depend on $|\mathcal{S}|$. Similarly, the number of different applications of rule V6 to the assertion $s{:}(\leq n\, R)$ is bounded by I and n (by a binomial coefficient), and the total number of different applications of rule V6 does not depend on $|\mathcal{S}|$.

Step 3: Since the number of possible precompletions depends only on the number of different applications of nondeterministic rules, such a number is $O(1)$ w.r.t. $|\mathcal{S}|$.

Step 4: The schema complexity of the entire method is the product of the following factors:

- maximal number of precompletions (a constant w.r.t. $|\mathcal{S}|$)
- time to compute a precompletion (linear in $|\mathcal{S}|$ from Proposition 5.7)
- number of new individuals in a precompletion (I, a constant w.r.t. $\mathcal{S}$),
- schema complexity of the trace calculus applied to $\langle \mathcal{S}, \mathcal{W}'_x \rangle$ (again, linear in $|\mathcal{S}|$)

Therefore, the schema complexity of instance checking is in $O(|\mathcal{S}|^2)$. □

We conclude the section by summarizing and commenting the main result.

Theorem 5.11 *With $\mathcal{SL}$ as schema language and $\mathcal{ALCNR}$ as view language, view subsumption and instance checking are PSPACE-complete problems w.r.t. combined complexity and PTIME problems w.r.t. schema complexity.*

Compare this result with the fact that using $\mathcal{ALCNR}$ also as the schema language, combined complexity is EXPTIME-hard (Schild, 1994a), while subsumption between $\mathcal{ALCNR}$-concepts (without any schema) is already PSPACE-complete. Hence, we can conclude that simple inclusion axioms with cycles can be added to systems like KRIS without changing substantially the complexity of reasoning services, whereas adding full cyclic definitions increases significantly the complexity.

It is important to note that the results on schema complexity can be extended to other languages (*e.g.*, $\mathcal{ALC}$ plus inverse roles). In fact, the schema rules are valid independently of the view rules and they can be applied a polynomial number of times with respect to the size of the schema, still independently of the view rules. The key point is that schema rules create new individuals only if an assertion of the form $x: \forall R.C$ is present, and schema rules themselves never add such assertions to a world description. Hence, the number of applications of the schema rules is fixed by the size of the knowledge base generated by the view rules and by the number of assertions of the form $x: \forall R.C$ the view rules can generate. This is a constant with respect to the size of the schema (unless the view contains some constructors that can trigger infinite applications of the rules, like the transitive closure construct).

5.2 The Language of CLASSIC as View Language

The view language we study in this section is the concept language of the CLASSIC system.[8] Here, we only consider the constructs that can be given a declarative semantics while we ignore those which allow one to make use of the host programming language.

The CLASSIC language has several constructs that are not contained in Tables 1 and 2. First, there is the construct ONE-OF$(a_1, \ldots, a_k)$, which intuitively stands for the set of individuals $a_1, \ldots, a_k$. Second, CLASSIC distinguishes two kinds of roles: usual roles (denoted as P) and functional roles, called *attributes* (denoted as F). Both kinds of roles can be employed in expressions of the form FILLS(P, a) and FILLS(F, a), which intuitively describe the set of objects having the individual a as a filler of the role P or the attribute F, respectively. Finally, attributes can be combined into chains $F_1 \circ \cdots \circ F_n$ (denoted as p, q). Such chains can appear in concepts of the form SAME-AS(p, q), which are interpreted as the set of objects such that the chain p leads to the same object as the chain q.[9]

The declarative core of CLASSIC's concept language can be captured by the following syntax rules:

$$\begin{array}{rcl} C, D & \longrightarrow & A \mid \text{ONE-OF}(a_1, \ldots, a_k) \mid \text{FILLS}(P, a) \mid \text{FILLS}(F, a) \mid \\ & & C \sqcap D \mid \forall P.C \mid \forall F.C \mid (\leq n\, P) \mid (\geq n\, P) \mid \text{SAME-AS}(p, q) \end{array}$$

[8] For a description of CLASSIC see (Borgida et al., 1989; Borgida & Patel-Schneider, 1994).

[9] Only chains of attributes are allowed as arguments to SAME-AS(.,.) in order to keep reasoning in CLASSIC decidable.

V_1	$=$	$\mathsf{works\text{-}for}\colon \{\mathsf{SALES}\} \sqcap (\leq 1\, \mathsf{works\text{-}for})$
V_2	$=$	$\mathsf{Employee} \sqcap \forall \mathsf{works\text{-}for}.\{\mathsf{SALES}, \mathsf{MARKETING}, \mathsf{PERSONNEL}\}$

Figure 6: $\mathcal{CL}$-Views

$$p, q \quad \longrightarrow \quad F \mid p \circ q.$$

The semantics is as one would expect, except for the constructs ONE-OF and FILLS, that allow one to refer to individuals. Individuals appearing in these expressions have a semantics different from individuals in first order logic. They are interpreted as primitive disjoint concepts (see Borgida & Patel-Schneider, 1994), *i.e.*, as subsets of the domain, instead of as single elements of it.

In order to capture correctly how CLASSIC treats individuals, we use the following syntax and conventions. Each occurrence of an individual a appearing in a concept expression is replaced with an *individual concept* I_a. Individual concepts are pairwise disjoint concept names (*i.e.*, for every interpretation $\mathcal{I}$, we have $I_a^{\mathcal{I}} \cap I_b^{\mathcal{I}} = \emptyset$ for $a \neq b$). Individual concepts can appear neither in the schema nor on the left-hand side of a definition. Furthermore, the assertion $a\colon I_a$ is added to the world description for each a appearing in the knowledge base.

An expression ONE-OF$(a_1, \ldots, a_k)$ is represented as a set of individual concepts $\{I_{a_1}, \ldots, I_{a_k}\}$, written for simplicity $\{I_1, \ldots, I_k\}$. We interpret it as the disjunction of individual concepts $I_1 \sqcup \cdots \sqcup I_k$. For sets of individual concepts we use the operations of intersection and union with their usual meaning.

The FILLS(P, a) construct is now a particular case of existential quantification, written as $P\colon I_a$ or $P\colon I$ for simplicity, where I is an individual concept, and interpreted similarly as $\exists P.\{I\}$.

We capture attributes by usual roles for which we enforce functionality by the two $\mathcal{SL}$-schema axioms $P \sqsubseteq A_P \times A'_P$ and $A_P \sqsubseteq (\leq 1\, P)$, where A_P, A'_P are concept names appearing only in these two axioms.

In addition, the SAME-AS(p, q) construct is expressed by the existential agreement of role chains $\exists Q \doteq R$, where we assume that every chain consists of roles whose functionality has been stated in the schema.

We call $\mathcal{CL}$ the resulting language, whose syntax is the following:

$$\begin{aligned} C, D \quad &\longrightarrow \quad A \mid S \mid P\colon I \mid C \sqcap D \mid \forall P.C \mid (\leq n\, P) \mid (\geq n\, P) \mid \exists Q \doteq R \\ Q, R \quad &\longrightarrow \quad P \mid R \circ Q, \end{aligned}$$

where S denotes a concept of the form $\{I_1, \ldots, I_k\}$.

Examples of $\mathcal{CL}$-views are given in Figure 6. View V_1 denotes people working only in the SALES department. View V_2 denotes the employees working in one of the three listed departments. Given the domain restriction of role $\mathsf{works\text{-}for}$ in the schema of Figure 2, view V_1 is subsumed by V_2.

A $\mathcal{CL}$-knowledge base is a pair $\langle \mathcal{S}, \mathcal{W} \rangle$, such that $\mathcal{S}$ is an $\mathcal{SL}$-schema, $\mathcal{W}$ is a $\mathcal{CL}$-world description and for each role P occurring in an existential agreement $\exists Q \doteq R$ occurring in $\mathcal{W}$, there is a pair of axioms in $\mathcal{S}$ stating that P is functional.

We now prove that the combined complexity of reasoning in our architecture is PTIME, and therefore that a limited form of cycles can be added to CLASSIC without endangering the tractability, which was one of the main concerns of the CLASSIC designers (see Borgida et al., 1989). In the next subsection we introduce a calculus for reasoning about $\mathcal{CL}$-knowledge bases, and in the following one we study the complexity of reasoning by means of the calculus.

5.2.1 Completion Rules of the $\mathcal{CL}$-Calculus

Since the original algorithm for subsumption between $\mathcal{CL}$-concepts is based on a normal-form transformation to so-called description graphs (Borgida & Patel-Schneider, 1994), it is not easily extensible so as to deal with schema axioms. Therefore, we employ again a tableaux-like calculus for reasoning with $\mathcal{CL}$-concepts and $\mathcal{SL}$-schemata. However, some optimizations are needed in order to keep the reasoning process tractable. Among others, a control structure is needed for the treatment of the construct $P : I$. In fact, being similar to a qualified existential quantification of the form $\exists P.C$, a non-optimized calculus would create world descriptions of exponential size (as shown in Donini et al., 1992, see also Subsection 4.2.1).

As a side result of our work, we have a tableaux-like algorithm for reasoning in pure CLASSIC.

We consider both view subsumption $C \sqsubseteq_{\mathcal{S}} D$ and instance checking $\langle \mathcal{S}, \mathcal{W} \rangle \models a \colon D$, where C, D are $\mathcal{CL}$-concepts and $\langle \mathcal{S}, \mathcal{W} \rangle$ is a $\mathcal{CL}$-knowledge base.

Let C, D be $\mathcal{CL}$-concepts. Using the equivalence $(\forall P.C_1) \sqcap (\forall P.C_2) \equiv \forall P.(C_1 \sqcap C_2)$, every concept D can be transformed into an equivalent concept $D^1 \sqcap \cdots \sqcap D^n$, where each D^i is a conjunction-free concept. Then, $C \sqsubseteq_{\mathcal{S}} D$ if and only if for every conjunct D^i we have $C \sqsubseteq_{\mathcal{S}} D^i$. Similarly, $\langle \mathcal{S}, \mathcal{W} \rangle \models a \colon D$ iff for every conjunct D^i we have $\langle \mathcal{S}, \mathcal{W} \rangle \models a \colon D^i$. Thus, without loss of generality, from this point on we assume D to be conjunction-free.

Proposition 5.1 holds for $\mathcal{CL}$ accordingly. Therefore, both view subsumption and instance checking are reduced to the satisfiability of a knowledge base that contains a concept of the form $\neg D$.

As in the previous section, we rewrite the concept $\neg D$ into negation normal form by "pushing" the negation inside the concept. In particular, the concept $\neg(P : I)$, being equivalent to $\neg\exists P.\{I\}$, is rewritten as $\forall P.\neg\{I\}$. However, notice that the result of rewriting $\neg D$ is not always a $\mathcal{CL}$-concept. For example, the negation of universal quantification introduces qualified existential quantification, which is not a $\mathcal{CL}$ construct.

Therefore, our calculus must cope not only with the constructs of $\mathcal{CL}$, but also with the constructs obtained by rewriting negated conjunction-free $\mathcal{CL}$-concepts.

Looking at the syntax of $\mathcal{CL}$, we see that the negation of a conjunction-free $\mathcal{CL}$-concept has always the following form

$$\exists P_1.\exists P_2.\cdots\exists P_n.E,$$

where $n \geq 0$ and E is the (rewritten) negation of a conjunction-free $\mathcal{CL}$-concept without universal quantification. Hence, it is a concept of the form $\neg A$, $\neg S$, $\forall P.\neg\{I\}$, $(\leq n\,P)$, $(\geq n\,P)$, or $\neg\exists Q \doteq R$.

As in the previous section, the $\mathcal{CL}$-*calculus* is specified by a set of $\mathcal{CL}$-*completion rules*, which are divided into *schema rules* and *view rules*. Since the schema language $\mathcal{SL}$ is the same as in the previous subsection, the schema rules are the same as in Section 5.1. The view rules for $\mathcal{CL}$ are presented in Figure 7.

Rule V3 deals with the assertion $s\colon\exists P_1.\exists P_2.\cdots\exists P_n.C$, which is present in the initial knowledge base. The repeated applications of rule V3 to the assertion of that form would generate (among others) a set of assertions of the form

$$\{sP_1x_1, x_1P_2x_2, \ldots, x_{n-1}P_nx_n, x_n\colon C\}.$$

We call such a set of assertions a *thread*, the individuals $s, x_1, x_2, \ldots, x_n$ the *thread individuals*, and n the *size of the thread*. Notice that, since the concept D is conjunction-free, there exists at most one thread in a knowledge base.

Observe also that rule V7 introduces a concept $\{I\}$, with a singleton set of individual concepts. This is to provide a correct interaction between the individual concept I appearing in $P : I$ and individual concepts present in concepts of the form $\{I_1, \ldots, I_n\}$. In fact, remember that individual concepts are interpreted as disjoint sets and therefore no object can be in the interpretation of two different individual concepts. For example, the calculus must detect that the world description $\{a\colon\forall P.\{I_1, I_2\}, a\colon(P\colon I_3)\}$ is unsatisfiable.

The notions of *directly derived*, *derived*, *complete*, and *completion* for the $\mathcal{CL}$-calculus are defined analogously to the corresponding definitions for the $\mathcal{ALCNR}$-calculus in Section 5.1.

Theorem 5.3 (Invariance) holds accordingly for the $\mathcal{CL}$-calculus.

As in the previous section, the calculus is nondeterministic. In fact, rule V4 is nondeterministic, and therefore there can be more than one knowledge base directly derived from a given knowledge base. This is dealt with in Point 2 of the Invariance Theorem.

A clash is a world description of one of the following forms:

- $\{a\colon(\leq n\,P)\} \cup \{aPb_1, \ldots, aPb_{n+1}\}$, where $a, b_1, \ldots, b_{n+1}$ are old individuals.
- $\{s\colon(\geq m\,P), s\colon(\leq n\,P)\}$ with $m > n$
- $\{sPt, s\colon(\leq 0\,P)\}$

V1: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{s\colon C_1,\ s\colon C_2\} \cup \mathcal{W} \rangle$
if 1. $s\colon C_1 \sqcap C_2$ is in $\mathcal{W}$

V2: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{t\colon C\} \cup \mathcal{W} \rangle$
if 1. $s\colon \forall P.C$ is in $\mathcal{W}$, and
2. t is a P-successor of s

V3: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{sPy,\ y\colon C\} \cup \mathcal{W} \rangle$
if 1. $s\colon \exists P.C$ is in $\mathcal{W}$, and
2. y is a new individual

V4: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \mathcal{W}[y/t] \rangle$
if 1. $s\colon (\leq n\, P)$ is in $\mathcal{W}$, and
2. s has more than n P-successors in $\mathcal{W}$,
3. y, t are two P-successors of s that are not separated

V5: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{s\colon S\} \cup \mathcal{W} \rangle$
if 1. $s\colon S_1, \ldots, s\colon S_n$ are in $\mathcal{W}$ and $S = S_1 \cap \cdots \cap S_n$

V6: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{sPy_i \mid i \in 1..n\} \cup \{y_i \not\doteq y_j \mid i, j \in 1..n, i \neq j\} \cup \mathcal{W} \rangle$
if 1. $s\colon (\geq n\, P)$ is in $\mathcal{W}$, and
2. $y_1, \ldots, y_n$ are new individuals,
3. there do not exist n pairwise separated P-successors of s in $\mathcal{W}$

V7: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{sPy, y\colon \{I\}\} \cup \mathcal{W} \rangle$
if 1. $s\colon (P : I) \in \mathcal{W}$

V8: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{sP_1y_1, \ldots, y_{n-1}P_nw\} \cup \{sQ_1z_1, \ldots, z_{k-1}Q_kw\} \cup \mathcal{W} \rangle$
if 1. $s\colon \exists R \doteq Q$ is in $\mathcal{W}$, and
2. $R = P_1 \circ \cdots \circ P_n$, and $Q = Q_1 \circ \cdots \circ Q_k$,
3. $y_1, \ldots, y_{n-1}, z_1, \ldots, z_{n-1}, w$ are new individuals

Figure 7: The view rules for $\mathcal{CL}$

- $\{s\colon A, s\colon \neg A\}$
- $\{s\colon S_1, s\colon \neg S_2\}$ with $S_1 \subseteq S_2$
- $\{s\colon \emptyset\}$
- $\{s\colon \neg\exists R \doteq Q, sP_1t_1, \ldots, t_{n-1}P_nv,\ sQ_1u_1, \ldots, u_{k-1}Q_kv\}$ with $R = P_1 \circ \cdots \circ P_n$, and $Q = Q_1 \circ \cdots \circ Q_k$.

Note that clashes involving number restrictions and new individuals are treated differently in this setting than in Section 5.1: Dealing only with atomic roles, in order to detect a clash involving an assertion of the form $s\colon (\leq n\, P)$ it is not necessary to look for sets of pairwise separated P-successors of s; instead, we look only for an assertion of the form $s\colon (\geq m\, P)$ with $m > n$.

Conversely, the clashes involving old individuals are treated as in Section 5.1.

This is done so as to detect contradictions explicitly present in the original knowledge base, *e.g.*, $\{a\colon (\leq 1\, P), aPb, aPc\}$.

As usual, we show that the completion rules always detect a clash in an unsatisfiable world description, by proving the converse: A complete, clash-free world description has a model, hence is satisfiable.

Let $\Sigma = \langle \mathcal{S}, \mathcal{W} \rangle$ be a $\mathcal{CL}$-knowledge base. We define the *canonical interpretation* $\mathcal{I}_\Sigma$ in the same way we did in Section 5.1 (we add the interpretation of individual concepts):

$$
\begin{array}{rcl}
\Delta^{\mathcal{I}_\Sigma} & := & \{s \mid s \text{ is an individual in } \mathcal{W}\} \cup \{u\} \\
s^{\mathcal{I}_\Sigma} & := & s \\
A^{\mathcal{I}_\Sigma} & := & \{s \mid s\colon A \text{ is in } \mathcal{W}\} \cup \{u\} \\
I^{\mathcal{I}_\Sigma} & := & \{s \mid s\colon I \text{ is in } \mathcal{W}\} \\
P^{\mathcal{I}_\Sigma} & := & \{(s,t) \mid sPt \text{ is in } \mathcal{W}\} \cup \{(u,u)\} \\
 & & \cup \{(s,u) \mid \text{there is no } sPt \text{ in } \mathcal{W}, \text{ but for some } A, \\
 & & \qquad\qquad s\colon A \text{ is in } \mathcal{W} \text{ and } A \sqsubseteq (\geq 1\, P) \text{ is in } \mathcal{S}\}.
\end{array}
$$

Theorem 5.12 *A complete clash-free $\mathcal{CL}$-knowledge base is satisfiable.*

Proof. (See Appendix) □

5.2.2 Termination and Complexity of the $\mathcal{CL}$-Calculus

In this section we will show that with $\mathcal{SL}$ as schema language and $\mathcal{CL}$ as view language, the combined complexity of view subsumption and of instance checking is PTIME. The above correct and complete calculus can be turned into an actual procedure for view subsumption and instance checking. However, it may produce several completions, each one of exponential size. For example, if the knowledge base contains an assertion of the form

$$
\begin{array}{l}
a\colon (P\colon I_1) \sqcap (P\colon I_2) \sqcap \\
\qquad\qquad \forall P.((P\colon I_3) \sqcap (P\colon I_4) \sqcap \\
\qquad\qquad\qquad\qquad \forall P.(\cdots (P\colon I_{2n-1}) \sqcap (P\colon I_{2n}) \sqcap \forall P.A) \cdots)
\end{array}
$$

the corresponding completion would have $O(2^n)$ new variables and $O(2^n)$ assertions on such variables. In order to have one completion of polynomial size, we modify rules and add a suitable strategy.

Given a new individual x, we say that y is a *sibling* of x in a world description $\mathcal{W}$ if there exists a role P and an individual s such that both sPx and sPy are in $\mathcal{W}$.

The rules responsible for the exponential size of a completion are the generating rules V6 and V7. Rule V3, instead, is applied only a number of times equal to the

S2′ $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{sPy\} \cup \mathcal{W} \rangle$
if 1. s: $\forall P.C$ is in $\mathcal{W}$,
2. s: A is in $\mathcal{W}$,
3. $A \sqsubseteq (\geq 1\, P)$ is in $\mathcal{S}$,
4. y is a new individual and
5. neither s nor any sibling of s has a P-successor in $\mathcal{W}$

V6′ $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{sPy\} \cup \mathcal{W} \rangle$
if 1. s: $(\geq n\, P)$ is in $\mathcal{W}$,
2. y is a new individual, and
3. neither s nor any sibling of s has a P-successor in $\mathcal{W}$

V7′ $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{sPy, y\colon\{I\}\} \cup \mathcal{W} \rangle$
if 1. s: $(P : I) \in \mathcal{W}$,
2. there is no t such that both $sPt, t\colon\{I\}$ are in $\mathcal{W}$, and
3. if s is a new individual, no sibling of s has a P-successor in $\mathcal{W}$

Figure 8: The modified schema and view rules for $\mathcal{CL}$-queries.

size of the thread of the knowledge base. Hence, we modify rules V6 and V7 as in Figure 8. For uniformity reasons, we modify in the same way also Rule S2. All the other rules are left unmodified.

The completion rules we obtain this way are called *quasi-completion rules*. A knowledge base to which no quasi-completion rule is applicable, is a *quasi-completion*.

The basic idea of rules V6′ and V7′ is that when we have two siblings we generate the successors of only one of them. This is possible because (as proved later) the successors of the second sibling would have exactly the same properties as the corresponding successors of the first one. Therefore, their creation is useless, in the sense that the clashes they lead to would be detected anyway.

Observe that the additional condition about siblings is just a condition that possibly prevents the application of the rule. Also, observe that the assertion added by Rule V6′ is one of those added by Rule V6. Hence, a world description obtained applying quasi-completion rules is always a subset of a world description obtained applying $\mathcal{CL}$-completion rules. Therefore, the Invariance Theorem still holds for the quasi-completion rules.

The mechanism of quasi-completions is similar to the mechanism of traces introduced in the previous section. In fact, they are both meant to reduce the complexity by blocking the application of some generating rules. However, there are two main differences between the two mechanism. First, trace rules create one piece of the world description at a time, and so they are used only to save working space and not computing time. Conversely, quasi-completions completely disable the application of certain rules, and thus they allow for polynomial computation time. Second, quasi-completions deal with each role separately, in the sense that the presence of an assertion on a role does not affect the applicability of a rule involving a different role. The trace rules instead allow for the generation of the R-successor of a new individual for just one R at a time.

Observe that there can be several different quasi-completions, due to alternative applications of Rule V4. To obtain only one quasi-completion, we suitably drive the application of this rule. We impose the following strategy:

1. Rules applied to thread individuals have precedence over any other rule application.
2. If not applied to thread individuals, Rules S2′ and V6′ have lower priority than any other rule.
3. When alternative applications of Rule V4 are possible, making different substitutions between the direct P-successors of an individual s; if there is a thread individual t, then first substitute another individual with t; if, in addition, $t: \neg S \in \mathcal{W}$, and if there is an individual z such that the assertions $s: (P : I)$, sPz and $z: \{I\}$ are in $\mathcal{W}$, and $I \notin S$, then substitute z with t; otherwise, make any substitution.

From this point on, we assume that the quasi-completion rules are applied according to the above strategy. The following proposition states a key property of quasi-completions, which clarifies the role of the condition about siblings in the modified rules of Figure 8. The intuition of the proposition is that non-thread individuals have the same membership assertions (*i.e.*, assertions of the form $x: C$) as their siblings except for assertions of the form $x: \{I\}$, which come from the construct FILLS.

Proposition 5.13 *Let $\Sigma_0 = \langle \mathcal{S}, \mathcal{W} \rangle$ be a quasi-completion, let x be an individual in $\mathcal{W}$ that is not a thread individual, and let C be a concept that is not of the form $C = S$. If the assertion $x: C$ is in $\mathcal{W}$ and if there is a sibling y of x, then $y: C$ is in $\mathcal{W}$.*

Proof. (See Appendix) □

Notice that for a single knowledge base we can still have several quasi-completions. However, the next proposition ensures that the alternative possible applications all lead to the same result.

Proposition 5.14 *Let Σ be a knowledge base and let Σ' be a knowledge base directly derived from Σ according to the above strategy. Then Σ is satisfiable if and only if Σ' is satisfiable.*

Proof. (See Appendix) □

We now prove that quasi-completion can detect unsatisfiable knowledge bases, by showing that a clash-free quasi-completion can always be turned into a clash-free completion, and then by exploiting Theorem 5.12.

Proposition 5.15 *Let Σ be a clash-free quasi-completion. Then there exists a clash-free completion that extends Σ.*

Proof. (See Appendix) $\square$

We now prove that the size of the quasi-completion is polynomial w.r.t. the size of the initial knowledge base—that is, $\langle \mathcal{S}, \{x\colon C \sqcap \neg D\}\rangle$ for view subsumption, and $\langle \mathcal{S}, \mathcal{W} \cup \{a\colon \neg D\}\rangle$ for instance checking.

First we observe a simple property: For every role P let N_P be the number of concepts of the form $(P : I)$ occuring in the initial knowledge base, and let $N := \max(N_P)$. Then for every new individual x in a quasi-completion, and for every role P, there are at most N assertions of the form xPy_i. Obviously, N is greater or equal than the number of assertions of the form $x\colon (P : I)$, plus one possible assertion of the form $x\colon \exists P.C$. Observe that we can ignore assertions of the form $x\colon (\geq n\, P)$, since all these assertions create no successors if there is at least one assertion of the form $x\colon (P : I)$ or $x\colon \exists P.C$, and one successor (in total) otherwise. We can also ignore the assertions of the form $x\colon \exists R \doteq Q$, since in this case the roles are functional, and so only one successor is possible.

This means that there are at most N direct P-successors of x in the quasi-completion. Moreover, only one of these N direct successors has itself successors. This is obvious from the application conditions of generating rules in Figure 8, and from the strategy.

Similarly to the previous subsection, denote with $depth(D)$ the depth of a concept D, *i.e.*, the maximum sequence of nested quantifiers in D, including also number restrictions and $P\colon I$ as quantifiers, and letting the depth of a concept $\exists R \doteq Q$ be the length of the longest (between R and Q) chain of roles. Consider any chain of direct successors $s, y_1, \ldots, y_h$ in a quasi-completion. By induction on the application of rules, it can be proved that if $y_i\colon D$ is in $\mathcal{W}$, and D is a subconcept of a concept C occuring in the world description,then $depth(D) \leq depth(C) - i$. Then, using the fact that $depth(C) \leq |C|$, one can prove that the number h is bounded by $|C|$, that is, the length of any chain of direct successors in a quasi-completion is bounded by the size of the $\mathcal{CL}$-concepts involved, hence by the size of the world description. In addition, due to the condition on siblings, we see that for every concept (including subconcepts) appearing in the initial knowledge base, we have that at most N individuals are instances of it in the quasi-completion, where N is the upper bound on the number of successors of an individual introduced above. Since N is bounded by the size of the initial world description, we conclude that the number of new individuals of a quasi-completion is polynomial in the size of the initial knowledge base.

We now estimate the size of the quasi-completion of an initial knowledge base $\langle \mathcal{S}, \mathcal{W}\rangle$. The number of assertions of the form $a\colon C$, where a is an old individual, is bounded by the number of old individuals times the number of subconcepts in $\mathcal{S} \cup \mathcal{W}$, hence is polynomial. Also the number of assertions of the form aPb is obviously polynomial in $\mathcal{W}$. The number of assertions of the form $x: C$, where x is a new individual is polynomial, since the number of new individuals is polynomial, and C is a concept appearing in $\mathcal{S}$ or in $\mathcal{W}$. Similarly, the number of assertions of the form sPy is polynomial. Therefore, the size of the quasi-completion of $\langle \mathcal{S}, \mathcal{W} \rangle$ is polynomial w.r.t. $|\mathcal{S}| + |\mathcal{W}|$.

The time spent to build a quasi-completion is polynomially related to its size. In fact, the application of a rule takes polynomial time. In addition, all rules adding assertions cannot be applied more times than the size of the quasi-completion itself. Rule V4 does not add assertions, but it cannot be applied more times than the number of possible direct successors of each individual, hence a polynomial number of times.

Finally, clash detection in a quasi-completion can be done in polynomial time w.r.t. the size of the quasi-completion. This is obvious for the usual clashes, which can be detected in linear time. For the clash involving two sets of individual concepts, observe that it can be detected by testing set containment, which again can be done in polynomial time.

Therefore, we have proved the following theorem.

Theorem 5.16 *With $\mathcal{SL}$ as schema language and $\mathcal{CL}$ as view language, view subsumption and instance checking are problems in PTIME w.r.t. combined complexity.*

We conclude that adding (possibly cyclic) schema information does not change the complexity of reasoning with CLASSIC.

Note that adding the SAME-AS construct to $\mathcal{SL}$ would make view subsumption undecidable (Nebel, 1991).

5.3 The Language of CONCEPTBASE as View Language

In (Buchheit et al., 1994) the query language $\mathcal{QL}$ was defined, which is derived from the CONCEPTBASE system. In $\mathcal{QL}$, roles are formed with all the constructs of Table 2 on page 7. That is, roles can be primitive roles P or inverses P^{-1} of primitive roles. Furthermore, there are *role restrictions*, written $(R: C)$, where R is a role and C is a $\mathcal{QL}$-concept. Intuitively, $(R: C)$ restricts the pairs related by R to those whose second component satisfies C. Roles can be composed to so-called *paths*: $R_1 \circ R_2 \circ \cdots \circ R_n$. In $\mathcal{QL}$, concepts are formed according to the rule:

$$C, D \longrightarrow A \mid \top \mid \{a\} \mid C \sqcap D \mid \exists R.C \mid \exists Q \doteq R.$$

Observe that concepts and roles can be arbitrarily nested through role restrictions. All concepts in $\mathcal{QL}$ correspond to existentially quantified formulas. We feel that many practical queries are of this form and do not involve universal quantification.

Figure 9 contains some examples of $\mathcal{QL}$ queries. Suppose we are given the schema of Figure 2. Query Q_1 denotes all the managers and Q_2 all the employees

Q_1	$=$	Manager
Q_2	$=$	Employee $\sqcap$ $\exists$salary.HighSalary
Q_3	$=$	Researcher $\sqcap$ $\exists$lives-in $\doteq$ works-for $\circ$ situated
Q_4	$=$	Employee $\sqcap$ $\exists$(works-for: ResearchDept) $\doteq$ lives-in $\circ$ hosts

Figure 9: $\mathcal{QL}$ Queries

that get a high salary. Then query Q_1 is subsumed by Q_2 since every manager is an employee and salaries of managers must be high salaries. Query Q_3 denotes all the researchers that live in the town in which the department they are working for is situated. Query Q_4 denotes all the employees that work for a research department that the city they are living in is hosting. With hosts being the inverse of situated, query Q_3 is subsumed by Q_4. This is because every researcher is an employee and any department he works for is a research department.

For the combination of $\mathcal{SL}$ and $\mathcal{QL}$ in our architecture, we have the following results:

Theorem 5.17 *With $\mathcal{SL}$ as schema language and $\mathcal{QL}$ as view language, view subsumption and instance checking are in PTIME w.r.t. combined complexity.*

The result on instance checking is an easy consequence of the one on view subsumption observing that, by means of singletons, a world description can be completely described by means of concepts so that instance checking can then be reduced to subsumption checking (as shown in Schaerf, (1994)). Intuitively, the assertion $a\colon C$ corresponds to the concept $\{a\}\sqcap C$ and the assertion aRb to the concept $\{a\}\sqcap\exists R.\{b\}$. More precisely, the transformation Φ of a world description into a concept is defined as follows. Let $\mathcal{W}$ be a world description, C a concept, and a, b two individuals, then:

$$\begin{aligned}
\Phi(\mathcal{W}) &:= \sqcap_{(\alpha\in\mathcal{W})}\Phi(\alpha) \\
\Phi(a\colon C) &:= \exists Q.(\{a\}\sqcap C) \\
\Phi(aRb) &:= \exists Q.(\{a\}\sqcap\exists R.\{b\}),
\end{aligned}$$

where Q does not appear in $\mathcal{W}$. Intuitively, Φ "encodes" the world description $\mathcal{W}$ in the implicit assertions of the concept $\Phi(\mathcal{W})$. The following proposition states the relation between the $\mathcal{W}$ and $\Phi(\mathcal{W})$.

Proposition 5.18 *Given a schema $\mathcal{S}$, a world description $\mathcal{W}$, an individual a, and a concept C then:*

(*i*) *$\mathcal{W}$ is satisfiable iff $\Phi(\mathcal{W})$ is satisfiable,*

(*ii*) *$\langle\mathcal{S},\mathcal{W}\rangle\models C(a)$ iff $\Phi(\mathcal{W})\sqcap\{a\}\sqsubseteq_{\mathcal{S}} C$.*

D1: $\langle \mathcal{S}, \mathcal{F} \rangle \langle \mathcal{G} \rangle \;\rightarrow\; \langle \mathcal{S}, \{s\colon C,\ s\colon D\} \cup \mathcal{F} \rangle \langle \mathcal{G} \rangle$

if 1. $s\colon C \sqcap D$ is in $\mathcal{F}$

D2: $\langle \mathcal{S}, \mathcal{F} \rangle \langle \mathcal{G} \rangle \;\rightarrow\; \langle \mathcal{S}, \{sPy, y\colon C\} \cup \mathcal{F} \rangle \langle \mathcal{G} \rangle$

if 1. $s\colon \exists P.C$ is in $\mathcal{F}$,
2. there is no t such that sPt and $t\colon C$ are in $\mathcal{F}$, and
3. y is a fresh individual

S1: $\langle \mathcal{S}, \mathcal{F} \rangle \langle \mathcal{G} \rangle \;\rightarrow\; \langle \mathcal{S}, \{t\colon A_2\} \cup \mathcal{F} \rangle \langle \mathcal{G} \rangle$

if 1. $s\colon A_1$ and sPt are in $\mathcal{F}$, and
2. $A_1 \sqsubseteq \forall P.A_2$ is in $\mathcal{S}$

S2: $\langle \mathcal{S}, \mathcal{F} \rangle \langle \mathcal{G} \rangle \;\rightarrow\; \langle \mathcal{S}, \{sPy\} \cup \mathcal{F} \rangle \langle \mathcal{G} \rangle$

if 1. there is an A such that $s\colon A$ is in $\mathcal{F}$,
2. $A \sqsubseteq (\geq 1\, P)$ is in $\mathcal{S}$,
3. there is no t such that sPt is in $\mathcal{F}$, and $s\colon \exists P.C$ is in $\mathcal{G}$, and
4. y is a fresh individual

G1: $\langle \mathcal{S}, \mathcal{F} \rangle \langle \mathcal{G} \rangle \;\rightarrow\; \langle \mathcal{S}, \mathcal{F} \rangle \langle \mathcal{G} \cup \{s\colon C, s\colon D\} \rangle$

if 1. $s\colon C \sqcap D$ is in $\mathcal{G}$

G2: $\langle \mathcal{S}, \mathcal{F} \rangle \langle \mathcal{G} \rangle \;\rightarrow\; \langle \mathcal{S}, \mathcal{F} \rangle \langle \mathcal{G} \cup \{t\colon C\} \rangle$

if 1. $s\colon \exists P.C$ is in $\mathcal{G}$, and
2. sPt is in $\mathcal{F}$

C1: $\langle \mathcal{S}, \mathcal{F} \rangle \langle \mathcal{G} \rangle \;\rightarrow\; \langle \mathcal{S}, \{s\colon C \sqcap D\} \cup \mathcal{F} \rangle \langle \mathcal{G} \rangle$

if 1. $s\colon C$ and $s\colon D$ are in $\mathcal{F}$, and
2. $s\colon C \sqcap D$ is in $\mathcal{G}$

C2: $\langle \mathcal{S}, \mathcal{F} \rangle \langle \mathcal{G} \rangle \;\rightarrow\; \langle \mathcal{S}, \{s\colon \exists P.C\} \cup \mathcal{F} \rangle \langle \mathcal{G} \rangle$

if 1. there is a t such that sPt and $t\colon C$ are in $\mathcal{F}$, and
2. $s\colon \exists P.C$ is in $\mathcal{G}$

Figure 10: The decomposition, schema, goal, and composition rules

Proposition 5.18 can be proved analogously to Lemma 6.6 of (Schaerf, 1994).

A detailed proof of the view subsumption part of Theorem 5.17 can be found in (Buchheit et al., 1994). But, since the proof requires techniques quite different from the ones used in the preceding case studies, we will demonstrate the main characteristics of these techniques for a restricted schema and query language. The restricted query language $\mathcal{SL}^-$ is defined by the rule

$$D \longrightarrow\ \forall P.A \mid (\geq 1\ P).$$

An $\mathcal{SL}^-$-schema contains only inclusions of the form $A \sqsubseteq D$. In the restricted query language $\mathcal{QL}^-$ there are no role forming operators and concepts are formed according to the following syntax rule:

$$C, D \longrightarrow\ A \mid C \sqcap D \mid \exists P.C.$$

The basic idea for deciding subsumption between views C and D is as follows. We take an object o and transform C into a prototypical knowledge base where o is an instance of C. We do so by generating objects, entering them into concepts, and relating them through roles. Then we evaluate D over this knowledge base. If o belongs to the instances of D then C is subsumed by D. If not, we have an interpretation where an object is in C but not in D and therefore C is not subsumed by D. The next proposition gives the formal justification for this idea.

Proposition 5.19 *Let $\mathcal{S}$ be an $\mathcal{SL}^-$-schema, C, D be $\mathcal{QL}^-$-concepts, and o be an individual. Then*

$$C \sqsubseteq_{\mathcal{S}} D \quad \textit{iff} \quad \langle \mathcal{S}, \{o\colon C\}\rangle \models o\colon D.$$

The transformation and evaluation process is specified by a calculus, the *$\mathcal{QL}$-calculus* that features four kinds of rules: decomposition, schema, goal, and composition rules. The rules work on a knowledge base that consists of the schema $\mathcal{S}$ and a world description $\mathcal{F}$—called the *facts*—and on a second world description $\mathcal{G}$ called the *goals*. The knowledge base and the goals together are called a *pair* $\langle \mathcal{S}, \mathcal{F}\rangle\langle \mathcal{G}\rangle$. In order to decide whether $C \sqsubseteq_{\mathcal{S}} D$, we take an individual o and start with the knowledge base $\langle \mathcal{S}, \{o\colon C\}\rangle$ and the goal $\{o\colon D\}$. Applying the rules, we add more facts and goals until no more rule is applicable. Intuitively, C is subsumed by D iff the final knowledge base contains the fact $o\colon D$. This is a difference to the refutation style calculus of the first two case studies, where we start with the knowledge base $\langle \mathcal{S}, \{o\colon C, o\colon \neg D\}\rangle$ and check the completions for clashes. In the case of $\mathcal{QL}$ as view language this would lead to an exponential number of possible completions. All rules of this calculus exploit the hierarchical structure of concepts, which is the basic reason for the polynomiality of the procedure. The rules are presented in Figure 10. A rule is applicable to a pair if it satisfies the conditions associated with the rule and if it is altered when transformed according to the rule. The second requirement is needed to ensure termination of our calculus. As an example, Rule D1 is applicable to a pair $\langle \mathcal{S}, \mathcal{F}\rangle\langle \mathcal{G}\rangle$ if $\mathcal{F}$ contains a fact $s\colon C \sqcap D$ *and* if $s\colon C$ and $s\colon D$ are not both in $\mathcal{F}$.

The *decomposition rules* (D1, D2) work on facts. They break up the initial fact $o\colon C$ into facts involving only primitive concepts and primitive roles.

The *schema rules* (S1, S2) also work on facts. They add information derivable from the schema and the current facts. The first rule is simple. It adds membership assertions for individuals in $\mathcal{F}$. Rule S2, however, which might create a new individual, is subject to a tricky control that limits the number of new individuals: it is only applicable if it creates a role filler that is required by a goal. This control is comparable to the control of the corresponding rules in the preceding case studies. There the application is restricted to universally constrained individuals (see rule S2 in Figure 4). Note that an existential quantification in a goal would give rise to a universal quantification in the refutation style calculus. Without this control, an exponential number of individuals could be introduced in the worst case.

The *goal rules* (G1, G2) work on goals. They guide the evaluation of the concept D by deriving subgoals from the original goal $o\colon D$. The interesting rule is G2, since it relates goals to facts: if the goal is to find $s\colon \exists P.C$, then only individuals t are tested which are explicitly mentioned as P-fillers of s in the facts.

The *composition rules* (C1, C2) compose complex facts from simpler ones directed by the goals. This can be understood as a bottom up evaluation of concept D over $\mathcal{F}$.

Both the decomposition rule D2 and the schema rule S2 can introduce individuals. Since the individuals introduced by D2 carry more specific information than the ones created by S2, decomposition rules receive priority, *i.e.*, a schema rule can be applied only if no decomposition rule is applicable. This *strategy* contributes to keeping the whole procedure polynomial.

In (Buchheit et al., 1994) one can find the full calculus and a proof that for $\mathcal{QL}$-concepts C, D and an $\mathcal{SL}$-Schema $\mathcal{S}$, we have that $C \sqsubseteq_{\mathcal{S}} D$ if and only if $o\colon D$ is in the completed facts.

The complexity result is based on the observation that the number of individuals in the completion $\langle \mathcal{S}, \mathcal{F}_C \rangle \langle \mathcal{G}_D \rangle$ of $\langle \mathcal{S}, \{o\colon C\} \rangle \langle \{o\colon D\} \rangle$ is polynomially bounded by the size of C and D. For every individual introduced by a decomposition rule, there is an existentially quantified subconcept of C. Hence, the number of individuals generated by decomposition rules is less or equal to the size of C. Let us call these individuals *primary individuals*. Then, since the introduction of individuals by the schema rule S2 is controlled by the structure of D, one can show that for every primary individual the number of nonprimary successors is bounded by the size of D. Summarizing, we get a polynomial upper bound for the number of individuals. One can show that the number of rule applications is polynomially bounded by the number of individuals and the size of the schema $\mathcal{S}$. Thus, the completion of $\langle \mathcal{S}, \{o\colon C\} \rangle \langle \{o\colon D\} \rangle$ can be computed in time polynomial in the size of C, D and $\mathcal{S}$. This yields our claim.

Theorem 5.17 illustrates the benefits of the new architecture because by restricting universal quantification to the schema and existential quantification to views we can have both without losing tractability. Note that in the language $\mathcal{ALE}$ (cf. Subsection 4.2.1) which contains both universal and existential quantification, subsumption checking is NP-hard, even for cycle-free terminologies.

6 Conclusion

We have proposed to replace the traditional TBox in a terminological system by two components: a schema, where primitive concepts describing frame-like structures are introduced, and a view part that contains defined concepts. We feel that this architecture reflects adequately the way terminological systems are used in most applications.

We also think that this distinction can clarify the discussion about the semantics of cycles. Given the different functionalities of the schema and view part, we propose that cycles in the schema are interpreted with descriptive semantics while for cycles in the view part a definitional semantics should be adopted.

In three case studies we have shown that the revised architecture yields a better tradeoff between expressivity and the complexity of reasoning.

The schema language $\mathcal{SL}$ we have introduced might be sufficient in many cases. Sometimes, however, one might want to impose more integrity constraints on primitive concepts than can be expressed in it. We see two solutions to this problem: either we enrich the language and have to pay by a more costly reasoning process, or we treat such constraints in a passive way by only verifying them for the objects in the knowledge base. The second alternative can be given a logical semantics in terms of epistemic operators (see Donini, Lenzerini, Nardi, Nutt, and Schaerf, (1992)).

Appendix

Proof of Proposition 4.3.
Let $\mathcal{S}$ be an $\mathcal{SL}_{\text{dis}}$-schema. Obviously, $\mathcal{S}$ is locally valid if it is valid. To prove the converse, it suffices to show that for any concept names A_1, A_2, given two models $\mathcal{I}_1$ and $\mathcal{I}_2$ of $\mathcal{S}$ with $A_1^{\mathcal{I}_1} \neq \emptyset$ and $A_2^{\mathcal{I}_2} \neq \emptyset$ we can construct a model $\mathcal{I}$ of $\mathcal{S}$ such that $A_1^{\mathcal{I}} \neq \emptyset$ and $A_2^{\mathcal{I}} \neq \emptyset$.

Without loss of generality, we can assume that the domains $\Delta^{\mathcal{I}_1}$ and $\Delta^{\mathcal{I}_2}$ are disjoint. We then define $\mathcal{I}$ on the domain $\Delta^{\mathcal{I}} := \Delta^{\mathcal{I}_1} \cup \Delta^{\mathcal{I}_2}$ by $A^{\mathcal{I}} := A^{\mathcal{I}_1} \cup A^{\mathcal{I}_2}$ for every concept name A, $P^{\mathcal{I}} := P^{\mathcal{I}_1} \cup P^{\mathcal{I}_2}$ for every role name P, and $a^{\mathcal{I}} := a^{\mathcal{I}_1}$ for every individual a.

It is easy to verify that in the language $\mathcal{SL}_{\text{dis}}$ for every concept C we have $C^{\mathcal{I}} = C^{\mathcal{I}_1} \cup C^{\mathcal{I}_2}$. We conclude that an axiom satisfied by $\mathcal{I}_1$ and $\mathcal{I}_2$ is also satisfied by $\mathcal{I}$. Hence, $\mathcal{I}$ is a model of $\mathcal{S}$. By construction, both A_1 and A_2 are interpreted under $\mathcal{I}$ as nonempty sets. $\square$

Proof of Lemma 4.5.
"$\Rightarrow$" Suppose $\mathcal{S}_C$ is valid. There is an interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ such that $A_C^{\mathcal{J}} \neq \emptyset$. We modify $\mathcal{J}$ so as to yield an interpretation $\mathcal{I}$ with $C^{\mathcal{I}} \neq \emptyset$. We define $\mathcal{I}$ as equal to $\mathcal{J}$ for every symbol occurring in $\mathcal{S}_C$ and put $Q^{\mathcal{I}} := \bigcup_{P \in \mathcal{P}_C} P^{\mathcal{J}}$. Since $\mathcal{J}$ is a model of $\mathcal{S}_C$, so is $\mathcal{I}$, and $A_C^{\mathcal{I}} \neq \emptyset$. We will show by induction over the structure of concepts that $A_D^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every subconcept D of C. This implies that $A_C^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ and, since $A_C^{\mathcal{I}} \neq \emptyset$, the claim follows.

Base case: If $D = \top$, then $A_D^{\mathcal{I}} \subseteq \Delta^{\mathcal{J}} = \top^{\mathcal{I}}$. Suppose that $D = \forall Q.\bot$. The schema S_C contains the axiom $A^+ \sqsubseteq \neg A^-$, and for every $P \in \mathcal{P}_C$ the axioms $A_D \sqsubseteq \forall P.A^+$ and $A_D \sqsubseteq \forall P.A^-$. Thus, if $d \in A_D^{\mathcal{I}}$, then d has no filler for any of the roles $P \in \mathcal{P}_C$. Otherwise, such a filler would be an element of $(A^+)^{\mathcal{I}}$ and of $(A^-)^{\mathcal{I}}$, which is impossible, because these sets are disjoint. This proves that $A_D^{\mathcal{I}} \subseteq (\forall Q.\bot)^{\mathcal{I}}$.

Inductive case: If $D = D' \sqcap D''$, then $\mathcal{S}_C$ contains the axioms $A_D \sqsubseteq A_{D'}$ and $A_D \sqsubseteq A_{D''}$. By the induction hypothesis we know that $A_{D'}^{\mathcal{I}} \subseteq D'^{\mathcal{I}}$ and $A_{D''}^{\mathcal{I}} \subseteq D''^{\mathcal{I}}$. Hence, $A_D^{\mathcal{I}} \subseteq A_{D'}^{\mathcal{I}} \cap A_{D''}^{\mathcal{I}} \subseteq D'^{\mathcal{I}} \cap D''^{\mathcal{I}} = D^{\mathcal{I}}$.

If $D = \exists Q.D'$, then $\mathcal{S}_C$ contains the axioms $A_D \sqsubseteq (\geq 1\, P_D)$ and $A_D \sqsubseteq \forall P_D.D'$. This implies that for any $d \in A_D^{\mathcal{I}}$ there is some d' with $(d, d') \in P_D^{\mathcal{I}}$ and $d' \in A_{D'}^{\mathcal{I}}$. Then, by definition of Q, we have $(d, d') \in Q^{\mathcal{I}}$, and by the induction hypothesis we have $A_{D'}^{\mathcal{I}} \subseteq D'^{\mathcal{I}}$. Hence, $d \in (\exists Q.D')^{\mathcal{I}}$. This shows that $A_D^{\mathcal{I}} \subseteq (\exists Q.D')^{\mathcal{I}}$.

If $D = \forall Q.D'$, $D' \neq \bot$, then $\mathcal{S}_C$ contains for every $P \in \mathcal{P}_C$ the axiom $A_D \sqsubseteq \forall P.A_{D'}$. Let $d \in A_D^{\mathcal{I}}$ and $(d, d') \in Q^{\mathcal{I}}$. By definition of Q we have $(d, d') \in P^{\mathcal{I}}$ for some $P \in \mathcal{P}_C$. From the axioms it follows that $d' \in A_{D'}^{\mathcal{I}}$, which together with the induction hypothesis $A_{D'}^{\mathcal{I}} \subseteq D'^{\mathcal{I}}$ implies that $d' \in D'^{\mathcal{I}}$. This shows that $A_D^{\mathcal{I}} \subseteq (\forall Q.D')^{\mathcal{I}}$.

"$\Leftarrow$" Suppose C is satisfiable. We construct an interpretation $\mathcal{I}$ such that $A_C^{\mathcal{I}} \neq \emptyset$.

The concept C has a model $\mathcal{J}$. We extend $\mathcal{J}$ to an interpretation $\mathcal{I}$ by defining $\Delta^{\mathcal{I}} := \Delta^{\mathcal{J}} \cup \{d^+, d^-\}$, where d^+, d^- are two distinct objects that are not elements of $\Delta^{\mathcal{J}}$. The interpretation of the symbols in $\mathcal{S}_C$ is given by $A_D^{\mathcal{I}} := D^{\mathcal{J}}$ for every subconcept D of C, $(A^+)^{\mathcal{I}} := \{d^+\}$, $(A^-)^{\mathcal{I}} := \{d^-\}$, and, for D of the form $\exists Q.D'$, $P_D^{\mathcal{I}} := \{(d, d') \mid d \in A_D^{\mathcal{I}},\ d' \in A_{D'}^{\mathcal{I}},\ (d, d') \in Q^{\mathcal{J}}\}$ for every $P \in \mathcal{P}_C$.

We check that $\mathcal{I}$ satisfies every axiom in $\mathcal{S}_C$. For any $D = D' \sqcap D''$, $\mathcal{S}_C$ contains the axioms $A_D \sqsubseteq A_{D'}$ and $A_D \sqsubseteq A_{D''}$, which are satisfied, since by definition of $\mathcal{I}$, we have $A_D^{\mathcal{I}} = (D' \sqcap D'')^{\mathcal{J}} = D'^{\mathcal{J}} \cap D''^{\mathcal{J}} = A_{D'}^{\mathcal{I}} \cap A_{D''}^{\mathcal{I}}$.

If $D = \exists Q.D'$, then $\mathcal{S}_C$ contains the axioms $A_D \sqsubseteq (\geq 1\, P_D)$ and $A_D \sqsubseteq \forall P_D.D'$. Since $A_D^{\mathcal{I}} = (\exists Q.D')^{\mathcal{J}}$, for every $d \in A_D^{\mathcal{I}} = D^{\mathcal{J}}$ there is some $d' \in D'^{\mathcal{J}}$ such that $(d, d') \in Q^{\mathcal{J}}$, which implies that $(d, d') \in P_D^{\mathcal{I}}$. Thus, the first axiom is satisfied. By definition of $P_D^{\mathcal{I}}$, every filler for P_D is an element of $A_{D'}^{\mathcal{I}}$. Thus, the second axiom is satisfied.

If $D = \forall Q.D'$, then $\mathcal{S}_C$ contains for every $P \in \mathcal{P}_C$ the axiom $A_D \sqsubseteq \forall P.A_{D'}$. By definition, we have $A_D^{\mathcal{I}} = D^{\mathcal{J}}$, $A_{D'}^{\mathcal{I}} = D'^{\mathcal{J}}$, and $P_D^{\mathcal{I}} \subseteq Q^{\mathcal{J}}$. This implies that all such axioms are satisfied.

If $D = \forall Q.\bot$, then there are axioms $A^+ \sqsubseteq \neg A^-$, and $A_D \sqsubseteq \forall P.A^+$, $A_D \sqsubseteq \forall P.A^-$ for every $P \in \mathcal{P}_C$. By construction, $(A^+)^{\mathcal{I}}$ and $(A^-)^{\mathcal{I}}$ are disjoint. Thus, the first axiom is satisfied. Moreover, since $D^{\mathcal{I}} = (\forall Q.\bot)^{\mathcal{J}}$ and $P^{\mathcal{I}} \subseteq Q^{\mathcal{J}}$ for all $P \in \mathcal{P}_C$, it follows that elements of $A_D^{\mathcal{I}}$ do not have a filler for any role $P \in \mathcal{P}_C$. Thus, the latter axioms are satisfied.

This proves that $\mathcal{I}$ is a model of $\mathcal{S}_C$. Also, we have that $A_C^{\mathcal{I}} = C^{\mathcal{J}} \neq \emptyset$. However, it might be the case that $A_D^{\mathcal{I}} = \emptyset$ for some proper subconcept $D \neq \bot$ of C. Since such a subconcept D is satisfiable, it has a model from which we can construct in a similar way as above a model of $\mathcal{S}_C$ that interprets A_D as a nonempty set. This proves that $\mathcal{S}_C$ is locally valid. By Proposition 4.3, $\mathcal{S}_C$ is valid. $\square$

Proof of Lemma 4.8.
"$\Rightarrow$" Suppose there is a path $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_k$ in $\mathcal{G}_{\mathcal{S}}$ from $\mathcal{C} = \mathcal{C}_0$ to some conflict node $\mathcal{C}_k$. Then there are roles $P_1, \ldots, P_k$ such that P_i is necessary on some concept in $\mathcal{C}_{i-1}$, and $\mathcal{C}_i = \mathit{range}(P_i, \mathcal{C}_{i-1})$. Obviously, $\mathcal{C}_i \neq \emptyset$ for every $i \in 0..k$.

Assume that $A_1 \sqcap \cdots \sqcap A_m$ is $\mathcal{S}$-satisfiable. Then there is a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{S}$ with an element $d \in \Delta^{\mathcal{I}}$ such that $d \in A_1^{\mathcal{I}} \cap \ldots \cap A_m^{\mathcal{I}}$. We show by induction that for every $i \in 0..k$ we have $\bigcap_{A \in \mathcal{C}_i} A^{\mathcal{I}} \neq \emptyset$. The claim for $i = 0$ coincides with our assumption. Suppose that $d_{i-1} \in A^{\mathcal{I}}$ for every $A \in \mathcal{C}_{i-1}$. Since P_i is necessary on some $A \in \mathcal{C}_{i-1}$, there exists an element d_i such that $(d_{i-1}, d_i) \in P_i^{\mathcal{I}}$. Moreover, for every $B \in \mathcal{C}_i$ we have $d_i \in B^{\mathcal{I}}$, since there is a transition $A \xrightarrow{P_i}_{\mathcal{S}} B$ for some $A \in \mathcal{C}_{i-1}$. It follows that $d_k \in \bigcap_{B \in \mathcal{C}_k} B^{\mathcal{I}}$, which is impossible because $\mathcal{C}_k$ is a conflict node.

"$\Leftarrow$" Suppose that no conflict node is reachable by a path issuing from $\mathcal{C}$. We construct a model $\mathcal{I}$ of $\mathcal{S}$ such that $A_1^{\mathcal{I}} \cap \cdots \cap A_m^{\mathcal{I}} \neq \emptyset$. We define $\Delta^{\mathcal{I}}$ as the set of all nodes in $\mathcal{G}_{\mathcal{S}}$ that are reachable by a (possibly empty) path issuing from $\mathcal{C}$. For a concept name A we define

$$A^{\mathcal{I}} := \{\mathcal{C}' \in \Delta^{\mathcal{I}} \mid A' \in \mathcal{C}' \text{ for some } A' \preceq_{\mathcal{S}} A\}.$$

For a role P we define

$$P^{\mathcal{I}} := \{(\mathcal{C}', \mathit{range}(P, \mathcal{C}')) \mid \mathcal{C}' \in \Delta^{\mathcal{I}} \text{ and } P \text{ is necessary on some } A' \in \mathcal{C}'\}.$$

We have to check that $\mathcal{I}$ satisfies every axiom in $\mathcal{S}$.

Suppose that $P \sqsubseteq A \times B \in \mathcal{S}$. Let $(\mathcal{C}', \mathcal{C}'') \in P^{\mathcal{I}}$. Then there is some $A' \in \mathcal{C}'$ such that P is necessary on A'. Thus, there is some A'' with $A' \preceq_{\mathcal{S}} A''$ such that $A'' \sqsubseteq (\geq 1\, P) \in \mathcal{S}$. Since $\mathcal{S}$ is isa-complete, we have $A'' \preceq_{\mathcal{S}} A$. Hence, $A' \preceq_{\mathcal{S}} A$, which implies $\mathcal{C}' \in A^{\mathcal{I}}$. Also, there is a transition $A' \xrightarrow{P}_{\mathcal{S}} B$, which implies that $B \in \mathcal{C}''$. Hence, $\mathcal{C}'' \in B^{\mathcal{I}}$.

We now show that $\mathcal{I}$ satisfies all axioms of the form $A \sqsubseteq C$ in $\mathcal{S}$. Consider a concept name A and some $\mathcal{C}' \in A^{\mathcal{I}}$. Then there exists some $A' \preceq_{\mathcal{S}} A$ with $A' \in \mathcal{C}'$.

Suppose that $A \sqsubseteq B \in \mathcal{S}$. Then $\mathcal{C}' \in B^{\mathcal{I}}$, since $A' \preceq_{\mathcal{S}} B$.

Suppose that $A \sqsubseteq (\geq 1\, P) \in \mathcal{S}$. Then P is necessary on A'. With $\mathcal{C}'' := \mathit{range}(P, \mathcal{C}')$ we have (i) $(\mathcal{C}', \mathcal{C}'')$ is an edge in $\mathcal{G}_{\mathcal{S}}$, (ii) $\mathcal{C}'' \in \Delta^{\mathcal{I}}$, and (iii) $(\mathcal{C}', \mathcal{C}'') \in P^{\mathcal{I}}$.

Suppose that $A \sqsubseteq \forall P . B \in \mathcal{S}$. Let $(\mathcal{C}', \mathcal{C}'') \in P^{\mathcal{I}}$. Then $B \in \mathcal{C}''$, since $\mathcal{C}'' = \mathit{range}(P, \mathcal{C}')$, which implies that $\mathcal{C}'' \in B^{\mathcal{I}}$.

Suppose that $A \sqsubseteq (\leq 1\, P) \in \mathcal{S}$. This axiom is satisfied because, by construction of $\mathcal{I}$, every role is interpreted as a partial function.

Suppose that $A \sqsubseteq \neg B \in \mathcal{S}$. Assume that $\mathcal{C}' \in B^{\mathcal{I}}$. Then there is some $B' \preceq_{\mathcal{S}} B$ with $B' \in \mathcal{C}'$. This implies that $\mathcal{C}'$ is a conflict node, which is impossible, since $\Delta^{\mathcal{I}}$ contains only nodes reachable from $\mathcal{C}$, and no conflict node can be reached from $\mathcal{C}$. □

Proof of Lemma 4.17.
"⇒" Suppose there is a path $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_k$ in $\mathcal{G}_{\mathcal{S}}$ from $\mathcal{C}_0 = \{A_1, \ldots, A_m\}$ to some conflict node $\mathcal{C}_k$. Then $\mathcal{C}_k$ contains names B_k, $\tilde{B}_k$ such that there are B'_k, $\tilde{B}'_k$ with $B_k \preceq_{\mathcal{S}} B'_k$, $\tilde{B}_k \preceq_{\mathcal{S}} \tilde{B}'_k$, and $B'_k \sqsubseteq \neg\tilde{B}_k \in \mathcal{S}$. Thus, $\{B_k, \tilde{B}_k\}$ is a conflict node in $\mathcal{D}_{\mathcal{S}}$. For some B_{k-1}, $\tilde{B}_{k-1} \in \mathcal{C}_{k-1}$ and a role P_k there are P_k-transitions $B_{k-1} \xrightarrow{P_k}_{\mathcal{S}} B_k$, $\tilde{B}_{k-1} \xrightarrow{P_k}_{\mathcal{S}} \tilde{B}_k$. Also, the role P_k is necessary on some $\bar{B}_{k-1} \in \mathcal{C}_k$. Since $\mathcal{S}$ is dichotomic, P_k is necessary on $\mathit{dom}(P_k)$. Thus, there is a P_k-edge in $\mathcal{D}_{\mathcal{S}}$ from $\{B_{k-1}, \tilde{B}_{k-1}\}$ to $\{B_k, \tilde{B}_k\}$.

Going on this way, we find for any $l \in 1..k$ names B_l, $\tilde{B}_l \in \mathcal{C}_l$ and edges from $\{B_{l-1}, \tilde{B}_{l-1}\}$ to $\{B_l, \tilde{B}_l\}$ in $\mathcal{D}_{\mathcal{S}}$. Thus, for some A_i, $A_j \in \mathcal{C}_0$ there is a path in $\mathcal{D}_{\mathcal{S}}$ from $\{A_i, A_j\}$ to the conflict node $\{B_k, \tilde{B}_k\}$.

"⇐" Suppose there is a path $\{B_0, \tilde{B}_0\} \xrightarrow{P_1} \{B_1, \tilde{B}_1\}, \ldots, \{B_{k-1}\tilde{B}_{k-1}\} \xrightarrow{P_k} \{B_k, \tilde{B}_k\}$ in $\mathcal{D}_{\mathcal{S}}$ from $\{B_0, \tilde{B}_0\} = \{A_i, A_j\}$ to some conflict node $\{B_k, \tilde{B}_k\}$. We inductively define $\mathcal{C}_0 := \{A_1, \ldots, A_m\}$ and $\mathcal{C}_l := \mathit{range}(P_l, \mathcal{C}_{l-1})$ for $i \in 1..l$. Obviously, $\{B_l, \tilde{B}_l\} \subseteq \mathcal{C}_l$

for any $l \in 0..k$. Moreover, since each P_l is necessary on its domain, $\mathcal{C}_{l-1}$ and $\mathcal{C}_l$ are linked in $\mathcal{G}_{\mathcal{S}}$ by an edge with label P_l. Since B_k, $\tilde{B}_k \in \mathcal{C}_k$, we have that $\mathcal{C}_k$ is a conflict node in $\mathcal{G}_{\mathcal{S}}$.

Summarizing, we have exhibited a path in $\mathcal{G}_{\mathcal{S}}$ that connects $\{A_1, \ldots, A_m\}$ to the conflict node $\mathcal{C}_k$. □

Proof of Lemma 4.20.
"⇒" If C is satisfiable, then by Lemma 4.5 there is a model $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ of $\mathcal{S}_C$ such that $A_C^{\mathcal{I}} \neq \emptyset$. We modify $\mathcal{J}$ to a model $\mathcal{I}$ of $\bar{\mathcal{S}}_C$ with $A_C^{\mathcal{I}} \neq \emptyset$ and $A_0^{\mathcal{I}} = \emptyset$.

Let $\mathcal{I}$ have the same domain as $\mathcal{J}$. We define $A_i^{\mathcal{I}} := \emptyset$ for $i \in 0..k$. On the other concept and role names, $\cdot^{\mathcal{I}}$ coincides with $\cdot^{\mathcal{J}}$.

Obviously, $\mathcal{I}$ satisfies every axiom in $\bar{\mathcal{S}}_C$ that occurs in $\mathcal{S}_C$. Also, every axiom $A_i \sqsubseteq \forall P^{-1}.A_{i-1}$ for $i \in 1..k$ is satisfied by $\mathcal{I}$ because $A_i^{\mathcal{I}} = \emptyset$. Finally, we consider the case of the subconcept $D = \forall Q.\bot$. Since $\mathcal{J}$ is a model of $\mathcal{S}_C$, no element of $A_D^{\mathcal{J}}$ has a filler for any role $P \in \mathcal{P}_C$. This shows that every axiom $A_D \sqsubseteq \forall P.A_k$ with $P \in \mathcal{P}_C$ is satisfied.

Summing up, we have shown that there is a model $\mathcal{I}$ of $\bar{\mathcal{S}}_C$ such that $A_C^{\mathcal{I}} \not\subseteq A_0^{\mathcal{I}}$. We conclude that $\bar{\mathcal{S}}_C \not\models A_C \sqsubseteq A_0$.
"⇐" Suppose that $\bar{\mathcal{S}}_C \not\models A_C \sqsubseteq A_0$. Then there is a model $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ of $\bar{\mathcal{S}}_C$ and an element $d_0 \in \Delta^{\mathcal{J}}$ such that $d_0 \in A_C^{\mathcal{J}}$, but $d_0 \notin A_0^{\mathcal{J}}$. We construct a model $\mathcal{I}$ of $\mathcal{S}_C$ such that $d_0 \in A_C^{\mathcal{I}}$. With an argument as in the proof of Lemma 4.5 this implies that S_C is valid and hence that C is satisfiable.

Given an interpretation $\Delta^{\mathcal{I}}$, we say that an element $d' \in \Delta^{\mathcal{I}}$ is *reachable* from $d \in \Delta^{\mathcal{I}}$ if the pair (d, d') is in the transitive-reflexive closure of the union of all role interpretations.

Let $\Delta^{\mathcal{I}}$ be the set of elements of $\Delta^{\mathcal{J}}$ that are reachable from d_0. We define $(A^+)^{\mathcal{I}} := (A^-)^{\mathcal{I}} := \emptyset$. For the other concept and role names A and P we put $A^{\mathcal{I}} := A^{\mathcal{J}} \cap \Delta^{\mathcal{I}}$ and $P^{\mathcal{I}} := P^{\mathcal{J}} \cap (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$.

It is an easy task to check that every axiom of the form $A_D \sqsubseteq A_{D'}$ or $A_D \sqsubseteq \forall P.A_{D'}$ is satisfied by $\mathcal{I}$ if it is satisfied by $\mathcal{J}$. Suppose that $\bar{\mathcal{S}}_C$ contains some axiom $A_D \sqsubseteq (\geq 1\, P)$ and that $d \in A_D^{\mathcal{I}}$. Then $d \in A_D^{\mathcal{J}}$ and there is some $d' \in \Delta^{\mathcal{J}}$ such that $(d, d') \in P^{\mathcal{J}}$. Since d is reachable from d_0, so is d', and it follows that $(d, d') \in P^{\mathcal{I}}$. This shows that every axiom $A_D \sqsubseteq (\geq 1\, P)$ is satisfied by $\mathcal{I}$.

Finally, we consider the case of the subconcept $D = \forall Q.\bot$. We show that no element of $A_D^{\mathcal{J}}$ that is reachable from d_0 has a filler for any role $P \in \mathcal{P}_C$. ¿From this it follow that every axiom $A_D \sqsubseteq \forall P.A^+$ and $A_D \sqsubseteq \forall P.A^-$ with $P \in \mathcal{P}_C$ is satisfied by $\mathcal{I}$.

Assume, on the contrary, that there is an element $d \in A_D^{\mathcal{J}}$ reachable from d_0, and a role $P \in \mathcal{P}_C$ such that $(d, d') \in P^{\mathcal{J}}$ for some $d' \in \Delta^{\mathcal{J}}$. Since $\mathcal{J}$ is a model of $\bar{\mathcal{S}}_C$, it follows from the axiom $A_D \sqsubseteq A_k$ that $d' \in A_k^{\mathcal{I}}$.

We have $\lambda(\bot) = k$, which implies $\lambda(D) = k - 1$. Since d is reachable from d_0, and $\lambda(D) = k-1$, we conclude that there is a chain $d_0, d_1, \ldots, d_{k-2}d$ of length $k-1$ from d_0 to d. This chain can be extended to a chain of length k from d_0 to d'. Now, the axioms $A_i \sqsubseteq \forall P^{-1}.A_{i-1}$ imply that $d \in A_{k-1}$, $d_{k-2} \in A_{k-2}^{\mathcal{J}}, \ldots, d_0 \in A_0^{\mathcal{J}}$, which contradicts our initial assumption that $d_0 \notin A_0^{\mathcal{J}}$. Thus, no element of $A_D^{\mathcal{I}}$ has a filler for any role $P \in \mathcal{P}_C$ in $\mathcal{J}$. □

Proof of Proposition 5.4.
Suppose $\Sigma = \langle \mathcal{S}, \mathcal{W} \rangle$ is a complete clash-free $\mathcal{ALCNR}$-knowledge base. We show that the canonical interpretation $\mathcal{I}_\Sigma$ is a model of Σ. The assertions of the form sPt, and $s \neq t$ in $\mathcal{W}$ are obviously satisfied by $\mathcal{I}_\Sigma$. The assertions of the form $s\colon C$ can be proved to be satisfied based on known results for (analogous) constraint systems (see *e.g.*, Buchheit et al., (1993)); the proof is by induction on the structure of C.

For the axioms of the form $A \sqsubseteq C$, we have to prove that for every $d \in \Delta^{\mathcal{I}_\Sigma}$, if d is in $A^{\mathcal{I}_\Sigma}$ then d is in $C^{\mathcal{I}_\Sigma}$. Based on the definition of $\mathcal{I}_\Sigma$, the domain element d can be in $A^{\mathcal{I}_\Sigma}$ in two cases: either $d = u$ or $d = s$ and $s\colon A$ is in $\mathcal{W}$.

In the first case, from the definition of $\mathcal{I}_\Sigma$, we see that u is in the extension of every $\mathcal{SL}$-concept, thus u is in $C^{\mathcal{I}_\Sigma}$, too.

In the second case, if C is of one of the forms B or $(\leq 1\, P)$ then the axiom is satisfied based on the following line of reasoning: Since $s\colon A$ is in $\mathcal{W}$ and Σ is complete, based on the schema rules S4 and S5, $s\colon C$ is in $\mathcal{W}$ too, and therefore $s \in C^{\mathcal{I}_\Sigma}$.

Suppose now that $A \sqsubseteq \forall P.B$ is in $\mathcal{S}$ and $s\colon A$ is in $\mathcal{W}$. We have to show that for all d such that $(s, d) \in P^{\mathcal{I}_\Sigma}$, we have that $d \in B^{\mathcal{I}_\Sigma}$. From the definition of $\mathcal{I}_\Sigma$, for any such d either $d = u$ or there exists t such that $d = t$ and sPt is in $\mathcal{W}$. In the first case, u is in $B^{\mathcal{I}_\Sigma}$ because of the definition of $\mathcal{I}_\Sigma$. In the second case, since Σ is complete, for the rule S3, $t\colon B$ is in $\mathcal{W}$ and thus $t \in B^{\mathcal{I}_\Sigma}$ by definition of $\mathcal{I}_\Sigma$.

Consider now the case that $A \sqsubseteq (\geq 1\, P)$ is in $\mathcal{S}$ and $s\colon A$ is in $\mathcal{W}$. If there exists an individual t such that sPt is in $\mathcal{W}$, then (s, t) is in $P^{\mathcal{I}_\Sigma}$, and therefore s is in $(\geq 1\, P)^{\mathcal{I}_\Sigma}$. In case there is no t such that sPt is in $\mathcal{W}$, then based on the definition of $\mathcal{I}_\Sigma$, (s, u) is in $P^{\mathcal{I}_\Sigma}$, and thus $s \in (\geq 1\, P)^{\mathcal{I}_\Sigma}$ again.

One can prove that the axioms of the form $P \sqsubseteq A_1 \times A_2$ are satisfied by $\mathcal{I}_\Sigma$, using similar arguments. □

Proof of Proposition 5.5.
Point 1: By induction on the application of rules. The induction is obvious for schema rules, which never add assertions involving subconcepts of C. For view rules, the induction is straightforward, *e.g.*, if rule V4′ is applied because $y_i\colon \exists R.D$ is in $\mathcal{W}$ (condition 1), it adds the new assertion $y_{i+1}\colon D$. By the induction hypothesis, $depth(\exists R.D) \leq depth(C) - i$. For the new assertion, the claim holds since $depth(D) = depth(\exists R.D) - 1 \leq depth(C) - (i+1)$.

Point 2: Suppose no: Then, there is a direct successor y_{k+1} of y_k, with $k = |C|$. But such a successor has been introduced by the application of a generating rule, requiring the presence in $\mathcal{W}$ of an assertion of the form $y_k\colon D$, where either $D = \forall P.E$ (rule S2′), or $D = \exists R.E$ (rule V4′), or $D = (\geq n\, R)$ (rule V5′). Observe that all the concepts involved are subconcepts of C, hence Point 1 above applies: $depth(D) \leq depth(C) - k = depth(C) - |C|$. However, $depth(C)$ is obviously less or equal than $|C|$ and therefore $depth(D) \leq 0$. Since $depth(D)$ is at least 1, a contradiction follows.

Point 3: The number N is bounded by the sum of all numbers n in concepts of the form $(\geq n\, R)$, plus all concepts of the form $\exists R.D$, both appearing in C, plus all concepts of the form $\forall P.D$ appearing in C (condition 1 of the generating rule S2). Hence, $N \leq |C|$, if numbers are coded in unary notation.

Point 4: The individuals in a trace are a chain $x, y_1, \ldots, y_h$ plus all their direct successors. Therefore the total number of individuals in a trace is bounded by $(h+1) \cdot (N+1) \leq (|C|+1)^2$ which is in $O(|C|^2)$. The number of assertions of the form $s\colon D$ is then $O(|C|^2 \cdot (|C| + |\mathcal{S}|))$ (each subconcept of either C or $\mathcal{S}$, times the number of individuals). Given that in assertions $s \not\doteq t$ the individuals s, t must be both direct successors of the same individual, generated by the application of a rule V5, the number of assertions $s \not\doteq t$ is $O(N^2 \cdot |C|) = O(|C|^3)$. Finally, in the assertions of the form sPt the individual t must be a direct successor of s, hence their total number is $O(|C| \cdot N) = O(|C|^2)$. We conclude that the number of assertions in a trace (hence its size) is polynomial in $|C|$ and linear in $|\mathcal{S}|$.

Point 5: A proof for a similar problem is given in (Hollunder & Nutt, 1990) by showing that each rule application in Σ can be transformed into a trace rule application in a set of traces. By Proposition 5.2 the "successor" relation restricted to new individuals forms a tree. Hence, every completion can be decomposed into as many parts as there are branches in the successor tree. No assertion is lost, since the conditions of application of each rule are local, *i.e.*, they depend only on an individual and (possibly) its direct successors.

Point 6: The claim follows from the locality of clashes: All two types of clash depend on an individual s, and on constraints involving either s alone (first type of clash) or both s and direct successors of s (second type of clash). If Σ' contains a clash, consider the trace in which the successors of s—if any—are generated (there always must be such a trace, from the previous point). That trace contains the same clash as Σ'. □

Proof of Proposition 5.7.

Let N be the maximal number of direct successors of an old individual in a precompletion: similarly to the Point 3 of Proposition 5.5, N is bounded by the sum of all numbers n in concepts of the form $(\geq n\, R)$, plus all concepts of the form $\exists R.C$, plus all concepts of the form $\forall P.C$, all appearing in $\mathcal{W}$. Hence, $N \leq |\mathcal{W}|$, if numbers are coded in unary notation. Call ω the number of old individuals. The total number of individuals in the precompletion is then ω old individuals, plus as many new individuals as N times ω, in total $O(\omega \cdot (N+1))$ which is in $O(|\mathcal{W}|^2)$.

This proves that the number of individuals does not depend on $|\mathcal{S}|$.

The number of possible (sub)concepts is $O(|\mathcal{S}| + |\mathcal{W}|)$; hence the number of assertions of the form $s\colon C$ is bounded by the number of individuals times the number of possible concepts, that is $O(|\mathcal{W}|^2 \cdot (|\mathcal{S}|+|\mathcal{W}|))$. Similarly, the number of assertions $s \neq t$ is bounded by ω^2 (UNA on old individuals) plus $\omega \cdot N^2$, that is $O(|\mathcal{W}|^3)$. The number of assertions of the form sPt is bounded by $\omega^2 \cdot |\mathcal{W}|$ relations between old individuals plus $\omega \cdot N$, that is $O(|\mathcal{W}|^3)$. Summing up all assertions, the size of a precompletion is $O(|\mathcal{W}|^2 \cdot (|\mathcal{S}| + |\mathcal{W}|))$. □

Proof of Proposition 5.8.

"$\Rightarrow$" Each precompletion is derived from Σ using completion rules. If Σ itself is not a precompletion, then a rule is applicable to an old individual. If Σ is satisfiable, Point 2 of Theorem 5.3 says that there exists a satisfiable knowledge base directly derived from Σ by applying that rule. If the new knowledge base is not a precompletion, one can repeat the same argument, and so on until a satisfiable precompletion is reached. This calculus for obtaining a satisfiable precompletion eventually terminates, because it is just a restricted version of the general calculus—*i.e.*, the condition of application of the rules are more restrictive.

"$\Leftarrow$" By induction on the number of rule applications needed to obtain Σ' from Σ. The base case is trivial, while in the inductive case Point 1 of Theorem 5.3 proves the claim. □

Proof of Proposition 5.9.

"$\Rightarrow$" Obviously, a precompletion must be clash-free to be satisfiable. For each new individual x, let C_x be the conjunction of the concepts D such that $x\colon D$ is in $\mathcal{W}_x$. Obviously, the knowledge base $\langle \mathcal{S}, x\colon C_x\rangle$ is satisfiable if and only if $\langle \mathcal{S}, \mathcal{W}'_x\rangle$ is satisfiable (it is sufficient to apply Rule V1 as many times to decompose again C_x). Combining Propositions 5.5 and 5.4, we know that $\langle \mathcal{S}, x\colon C_x\rangle$ is satisfiable if and only if there exists a finite, clash-free completion of it. Such a clash-free completion contains a clash-free completion of $\langle \mathcal{S}, \mathcal{W}'_x\rangle$.

"$\Leftarrow$" Suppose there exists a clash-free precompletion $\Sigma' = \langle \mathcal{S}, \mathcal{W}'\rangle$ such that for each new individual x in $\mathcal{W}'$, the knowledge base $\langle \mathcal{S}, \mathcal{W}'_x\rangle$ has a clash-free completion; then one can compute a clash-free completion of Σ' as the union of Σ' and, for each x, the clash-free completion of $\langle \mathcal{S}, \mathcal{W}'_x\rangle$ (up to renaming of new individuals). Recall that all application conditions of each completion rule are local, *i.e.*, whether or not a rule is applied depends on assertions about one individual s, and possibly its direct successors. Hence, a completion of Σ can actually be constructed from Σ' and from separate completions of $\langle \mathcal{S}, \mathcal{W}'_x\rangle$, since each rule application in one completion does not need to check for assertions from other completions. Since also clash conditions are local, such a completion is clash-free, and by Proposition 5.4, Σ' is satisfiable. □

Proof of Theorem 5.12.

Suppose $\Sigma = \langle \mathcal{S}, \mathcal{W}\rangle$ is a complete clash-free $\mathcal{CL}$-knowledge base. We show that the canonical interpretation $\mathcal{I}_\Sigma$ can be extended to a model of Σ. The assertions of the form sPt, and $s \neq t$ in $\mathcal{W}$ are obviously satisfied by $\mathcal{I}_\Sigma$. The assertions of the form $s\colon C$ can be proved to be satisfied by induction on the structure of C.

Base cases. Assertions of the form $s\colon A$ are satisfied by definition of $\mathcal{I}_\Sigma$, and assertions of the form $s\colon \neg A$ are satisfied because Σ is clash-free, hence it does not contain the complementary assertion $s\colon A$. Given an individual s, all assertions of the form $s\colon S_1,\ldots,s\colon S_h$, with $S_i = \{I_{i_1},\ldots,I_{i_k}\}$, can be satisfied because Σ is complete, hence there is an i such that $S_i = \bigcap_{j=1}^{h} S_j$, and $S_i \neq \emptyset$ because Σ is clash-free. Assertions of the form $s\colon (\geq n\, P)$ are satisfied because Σ is complete, hence n P-successors of s were generated by an application of Rule V6, and they were not identified by an application of Rule V4, since they are pairwise separated. On the other hand, assertions of the form $s\colon (\leq n\, P)$ are satisfied because Σ is complete, hence there cannot be more than n P-successors of s in Σ, unless they are pairwise separated; but since sets of pairwise separated individuals are introduced only because of the presence of an assertion of the form $s\colon (\geq m\, P)$ (condition 1 of Rule V6), with $m > n$, this would mean that Σ contains a clash, contradicting the hypothesis of the theorem.

Induction cases. Assertions of the form $s\colon C$ and $s\colon \neg C$, where C is a concept in $\mathcal{CL}$, can be shown to be satisfied based on a straightforward induction, since Σ is complete.

It can be proved that the inclusion axioms contained in $\mathcal{S}$ are satisfied by $\mathcal{I}_\Sigma$, analogously to the proof of Proposition 5.4. □

Proof of Proposition 5.13.

The assertion $x\colon C$ must have been introduced by the application of some rule: By inspection of all rules, Rules V1, V2, V3, and the schema rules are the only rules which can add an assertion of the form $x\colon C$.

Regarding the schema rules, they add an assertion of the form $x\colon C$ only if there exists in $\mathcal{W}$ an assertion α of the form $x\colon A$ or sPx. The assertion α, on its turn, can have been added either by a schema rule or by a view rule. If α has been added by a schema rule there exists another assertion on x, say α', that allows the application of the rule that created α. Again, α' has been added either by a schema rule or by a view rule. Continuing this argument, we see that there must have been an application of a view rule on x that has generated the first assertion on x.

Rule V3 is applied only to thread individuals, therefore the assertion $x\colon C$ cannot have been added by the application of this rule.

Regarding Rule V1, it adds an assertion of the form $x\colon C$ only if there exists in $\mathcal{W}$ an assertion α of the form $x\colon C \sqcap E$. Following the same line of reasoning for schema rules, we reach the conclusion that there must have been an application of Rule V2 on x that has generated the first assertion on x.

Conditions 1–2 of Rule V2 require the existence of an individual v and a role Q such that the assertions $v\colon \forall Q.C$, vQx are in $\mathcal{W}$. We can therefore conclude that such assertions are in $\mathcal{W}$.

Furthermore, we know that y is a sibling of x, hence there is an individual u and a role R such that uRx, uRy are in $\mathcal{W}$. Hence, x is both a direct Q-successor of v and a direct R-successor of u. If v and u are different, this can happen only because u, v, x were generated by an application of Rule V8—by inspection of all rules, Rule V8 is the only one which can force a new individual to be a direct successor of two different individuals. But this rule requires the functionality of R, hence x would have no sibling. Therefore, u and v are the same, and (using the same reasoning) also R and Q are the same.

In conclusion, we have that the assertions $v{:}\forall Q.C$, vQx, vQy are in $\mathcal{W}$. It follows that, due to the strategy, the assertion $y{:}C$ has been added by Rule V2. This proves the claim. □

Proof of Proposition 5.14.
The theorem is trivially true if the rule applied is any rule but Rule V4 due to Theorem 5.3.

We therefore assume that the rule applied is Rule V4, according to the strategy.

We first show that in the "otherwise" case of Point 3 of the strategy, $\mathcal{W}$ is unsatisfiable. If there is more than one P-successor of s, then they have been generated by the applications of Rules V7′, V3 and V8, since from the strategy they are applied before Rules S2′ and V6′, and once they are applied, Condition 3 of Rule V6′ and Condition 5 of Rule S2′ are not fulfilled any more. We do not consider the case in which Rule V8 has been applied, since this rule deals with functional roles; hence all P-successors will be eventually identified, and all alternative substitutions between them lead to the same knowledge bases (up to renaming of new individuals). Hence we can concentrate on the case where all P-successors have been introduced by Rules V7′ and V3. If there is no thread individual t, then two individuals generated by the application of Rule V7 must be identified, which will lead to a clash of the form $\{y{:}\emptyset\}$. Also, if there is such a t, C is of the form $\neg S$, but any other individual z is such that $z{:}\{I\}$ is in $\mathcal{W}$, and $I \in S$, then the substitution leads to a clash of the form $\{t{:}S_1, t{:}\neg S_2\}$, with $S_1 \subseteq S_2$.

Therefore, alternative substitutions in the "otherwise" case all lead to a clash. However, there are alternative substitutions also in the case the conditions in the strategy are fulfilled; in what follows we show that these alternatives all lead to a satisfiable knowledge base, if the original knowledge base was satisfiable.

We now show that in the other cases all the alternative applications of the Rule V4 lead to the same result. Suppose there are two different individuals x, y that can be substituted with the same thread individual t. By inspection of the rules, we know that x and y have been generated because of the presence of two assertions of the form $s{:}(P{:}\, I_1)$ and $s{:}(P{:}\, I_2)$. Therefore, the variables x and y are included in the assertions sPx, $x{:}\{I_1\}$, sPy, and $y{:}\{I_2\}$. By Proposition 5.13 any other assertion on x and y is common to the two individuals.

Hence, the only assertions that distinguish x from y are the assertions regarding the concepts I_1 and I_2. If we are not in the "otherwise" case, then both these assertions do not lead to a clash with the assertions on t, therefore the fact that we substitute x or y with t does not affect the satisfiability of the knowledge base. $\square$

Proof of Proposition 5.15.
Let $\Sigma_0 = \langle \mathcal{S}, \mathcal{W}_0 \rangle$ be a quasi-completion; since we know that Σ_0 is contained in a completion $\langle \mathcal{S}, \mathcal{W} \rangle$, we add missing assertions to Σ_0 to obtain it, then we prove that it does not contain a clash.

Note that the only assertions which were not fully analyzed are assertions of the form $s\colon (P\colon I)$, and $s\colon (\geq n\, P)$. We divide the addition process in three steps.

Step 1: We consider assertions of the form $x\colon (P\colon I)$, such that Σ_0 does not contain the assertions xPz, $z\colon \{I\}$, for any z. Consider Rule V7$'$: Conditions 1–2 are fulfilled, hence, if the rule is not applicable, Condition 3 is not fulfilled; that is, x has a sibling y such that yPz is in $\mathcal{W}$, for some z. Hence, from Proposition 5.13, we know that there is an individual v and a role Q such that the assertions $v\colon \forall Q \mathbf{.} (P : I)$, vQx, and vQy, are in $\mathcal{W}$. Since Σ_0 is quasi-complete, also the assertion $y\colon (P : I)$ is in $\mathcal{W}$. In this case Condition 3 of Rule V7$'$ is fulfilled (in fact, due to Rule V7$'$, no sibling of y can have P-successors), hence the assertions yPu, $u\colon \{I\}$ for some new individual u are in $\mathcal{W}$. Now add to $\mathcal{W}$ the assertion xPu. Observe that if $\mathcal{W}$ contains an assertion of the form $x\colon \forall P \mathbf{.} C$, then by Proposition 5.13 also the assertion $y\colon \forall P \mathbf{.} C$ is in $\mathcal{W}$, hence $u\colon C$ is in $\mathcal{W}$. Hence the added assertion does not cause Rule V2 to be applicable. However, schema Rule S1 could now be applicable, and its application could fire other schema rules. Apply these rules, with the exclusion of generating rules, and observe that for each added assertion involving x, there is a corresponding assertion involving y (since yPu was already in $\mathcal{W}$, schema Rule S1 was already applied to y).

After doing that for all x, the resulting knowledge base $\Sigma_1 = \langle \mathcal{S}, \mathcal{W}_1 \rangle$ is complete w.r.t. Rule V7 (the rule of Figure 7).

Step 2: Consider assertions of the form $s\colon (\geq n\, P)$, such that s has k direct P-successors in $\mathcal{W}_1$ with $k < n$.

First consider the case $k > 0$: Take $n - k$ new individuals $y_1, \ldots, y_{n-k}$ (not appearing in $\mathcal{W}_1$), and for $i \in 1..n-k$ add the assertion sPy_i. After doing that, Rule V6 is not applicable; however, Rule V2 could be, because of the presence in $\mathcal{W}_1$ of an assertion $s\colon \forall P \mathbf{.} C$. If so, apply the rule and add the assertions $y_i\colon C$. Apply also schema Rule S1, and possibly other rules, but no generating rule.

Suppose now $k = 0$: By Condition 3 in Rule V6$'$, the individual s has a sibling x such that $x\colon (\geq n\, P)$ is in $\mathcal{W}_1$ and x has $h > 0$ P-successors. Applying the above procedure to x, we have that x has n P-successors y_i for $i \in 1..n$. Then for each P-successor y_i of x we add the assertion sPy_i.

After this step, for all s, the resulting knowledge base $\Sigma_2 = \langle \mathcal{S}, \mathcal{W}_2 \rangle$ is complete w.r.t. all rules but the generating ones (because the added individuals may themselves require the existence of some direct successors).

Step 3: Consider assertions of the form $x\colon(\geq n\,Q)$, $x\colon A$, together with a schema axiom of the form $A \sqsubseteq (\geq 1\,Q)$, where x is a new individual with no successor (*i.e.*, either it has been added in Step 2, or it was an individual already without successors in Step 2). Observe that from Condition 3 of Rule V6′ and Condition 5 of Rule S2′—which were not applicable to s in Σ_0, since Σ_0 is quasi-complete—there is a sibling z of x having a Q-successor. Due to the strategy, x cannot be a thread individual (the Rule V6′ would have been applied to it in that case). Moreover, since x and z are siblings, from Proposition 5.13, for any assertions of the form $x\colon(\geq n\,Q)$, $x\colon A$ there is in Σ_0 a corresponding assertion $z\colon(\geq n\,Q)$, $z\colon A$, and from Steps 1–2 of this construction, all successors of z required by such assertions have been added in $\mathcal{W}_2$. Then, for each role Q and individual u, if zQu is in $\mathcal{W}_2$ then add to $\mathcal{W}_2$ the assertion xQu.

After adding all assertions of the form xQu, one obtains a knowledge base $\Sigma_3 = \langle\mathcal{S}, \mathcal{W}_3\rangle$, which is complete and clash-free. Completeness follows from the fact that the new individuals added have the same successors of previously present individuals, to which rules were already applied. The fact that Σ_3 is clash-free can be shown by enumeration of all possible clashes. One proves that each clash requires the presence of assertions that could have been added (by Proposition 5.13) only if similar ones—for a different individual—were already present in Σ_0. However, this is impossible since, by hypothesis, Σ_0 is clash-free. For example, suppose that Σ_3 contains a clash of the form $\{y_i\colon A, y_i\colon \neg A\}$, where y_i is an new individual introduced in Step 2 because of the presence of the assertion $s\colon(\geq n\,P)$ such that s has k direct P-successors $x_1, \ldots, x_k$ in $\mathcal{W}_1$ with $k < n$. Obviously, the presence of the assertions on y_i must be caused by the presence of the assertions $s\colon\forall P.A$ and $s\colon\forall P.\neg A$ (or the single assertion $s\colon\forall P.(A \sqcap \neg A)$). This implies that $x_i\colon A$ and $x_i\colon\neg A$ must be in Σ_0. This lead to a contradiction, since Σ_0 is assumed to be clash-free. $\square$

References

Baader, F. (1990). Terminological cycles in KL-ONE-based knowledge representation languages. In *Proc. of the 8th Nat. Conf. on Artificial Intelligence (AAAI-90)*, pp. 621–626 Boston, Ma.

Baader, F., Bürckert, H.-J., Hollunder, B., Nutt, W., & Siekmann, J. H. (1990). Concept logics. In Lloyd, J. W. (Ed.), *Computational Logics, Symposium Proceedings*, pp. 177–201. Springer-Verlag.

Baader, F., & Hollunder, B. (1991). A terminological knowledge representation system with complete inference algorithm. In *Proc. of the Workshop on Processing Declarative Knowledge, PDK-91*, No. 567 in Lecture Notes In Artificial Intelligence, pp. 67–86. Springer-Verlag.

Baader, F., & Nutt, W. (1992). Are complete and expressive terminological systems feasible? Position paper. In *Working Notes of the AAAI Fall Symposium on Issues on Description Logics: Users meet Developers*, pp. 1–5.

Borgida, A., Brachman, R. J., McGuinness, D. L., & Alperin Resnick, L. (1989). CLASSIC: A structural data model for objects. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pp. 59–67.

Borgida, A., & Patel-Schneider, P. F. (1994). A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, *1*, 277–308.

Buchheit, M., Donini, F. M., & Schaerf, A. (1993). Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, *1*, 109–138.

Buchheit, M., Jeusfeld, A. M., Nutt, W., & Staudt, M. (1994). Subsumption between queries to object-oriented databases. *Information Systems*, *19*(1), 33–54. Special issue on Extending Database Technology, EDBT'94.

Calvanese, D., & Lenzerini, M. (1994). On the interaction between ISA and cardinality constraints. In *Proc. of the 10th IEEE Int. Conf. on Data Engineering (ICDE-94)*, pp. 204–213 Houston. IEEE Computer Society Press.

Calvanese, D., Lenzerini, M., & Nardi, D. (1994). A unified framework for class based representation formalisms. In Doyle, J., Sandewall, E., & Torasso, P. (Eds.), *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-94)*, pp. 109–120 Bonn. Morgan Kaufmann, Los Altos.

De Giacomo, G., & Lenzerini, M. (1994). Concept language with number restrictions and fixpoints, and its relationship with μ-calculus. In *Proc. of the 11th European Conf. on Artificial Intelligence (ECAI-94)*.

Dionne, R., Mays, E., & Oles, F. J. (1992). A non-well-founded approach to terminological cycles. In *Proc. of the 10th Nat. Conf. on Artificial Intelligence (AAAI-92)*, pp. 761–766. AAAI Press/The MIT Press.

Dionne, R., Mays, E., & Oles, F. J. (1993). The equivalence of model theoretic and structural subsumption in description logics. In *Proc. of the 13th Int. Joint Conf. on Artificial Intelligence (IJCAI-93)*, pp. 710–716 Chambery, France. Morgan Kaufmann, Los Altos.

Donini, F. M., Hollunder, B., Lenzerini, M., Spaccamela, A. M., Nardi, D., & Nutt, W. (1992). The complexity of existential quantification in concept languages. *Artificial Intelligence*, *2–3*, 309–327.

Donini, F. M., Lenzerini, M., Nardi, D., & Nutt, W. (1991). The complexity of concept languages. In Allen, J., Fikes, R., & Sandewall, E. (Eds.), *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-91)*, pp. 151–162. Morgan Kaufmann, Los Altos.

Donini, F. M., Lenzerini, M., Nardi, D., Nutt, W., & Schaerf, A. (1992). Adding epistemic operators to concept languages. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-92)*, pp. 342–353. Morgan Kaufmann, Los Altos.

Donini, F. M., Lenzerini, M., Nardi, D., & Schaerf, A. (1994). Deduction in concept languages: From subsumption to instance checking. *Journal of Logic and Computation*, *4*(4), 423–452.

Hollunder, B., & Nutt, W. (1990). Subsumption algorithms for concept languages. Tech. rep. RR-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany.

Hollunder, B., Nutt, W., & Schmidt-Schauß, M. (1990). Subsumption algorithms for concept description languages. In *Proc. of the 9th European Conf. on Artificial Intelligence (ECAI-90)*, pp. 348–353 London. Pitman.

Hopcroft, J. E., & Ullman, J. D. (1969). *Formal Languages and their Relation to Automata*. Addison Wesley Publ. Co., Reading, Massachussetts.

Jarke, M., Gallersdoerfer, R., Jeusfeld, M., Staudt, M., & Eherer, S. (1995). Conceptbase - A deductive object manager for meta databases. *Journal of Intelligent Information Systems*, *4*(2).

Johnson, D. S. (1990). A catalog of complexity classes. In van Leeuwen, J. (Ed.), *Handbook of Theoretical Computer Science*, Vol. A, chap. 2. Elsevier Science Publishers B. V. (North Holland).

Lenzerini, M., & Schaerf, A. (1991). Concept languages as query languages. In *Proc. of the 9th Nat. Conf. on Artificial Intelligence (AAAI-91)*, pp. 471–476.

Lloyd, J. W. (1987). *Foundations of Logic Programming (Second, Extended Edition)*. Springer, Berlin, Heidelberg.

MacGregor, R. (1991). Inside the LOOM description classifier. *SIGART Bulletin*, *2*(3), 88–92.

MacGregor, R. (1992). What's needed to make a description logic a good KR citizen. In *Working Notes of the AAAI Fall Symposium on Issues on Description Logics: Users meet Developers*, pp. 53–55.

Nebel, B. (1990a). *Reasoning and Revision in Hybrid Representation Systems*. Lecture Notes in Artificial Intelligence. Springer-Verlag.

Nebel, B. (1990b). Terminological reasoning is inherently intractable. *Artificial Intelligence*, *43*, 235–249.

Nebel, B. (1991). Terminological cycles: Semantics and computational properties. In Sowa, J. F. (Ed.), *Principles of Semantic Networks*, pp. 331–361. Morgan Kaufmann, Los Altos.

Peltason, C. (1991). The BACK system – an overview. *SIGART Bulletin*, *2*(3), 114–119.

Schaerf, A. (1993). On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, *2*, 265–278.

Schaerf, A. (1994). Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, *13*(2), 141–176.

Schild, K. (1994a). Private communication.

Schild, K. (1994b). Terminological cycles and the propositional μ-calculus. In Doyle, J., Sandewall, E., & Torasso, P. (Eds.), *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-94)*, pp. 509–520 Bonn. Morgan Kaufmann, Los Altos.

Schmidt-Schauß, M., & Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial Intelligence*, *48*(1), 1–26.

Vardi, M. (1982). The complexity of relational query languages. In *Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC-82)*, pp. 137–146.

Wahlster, W., André, E., Finkler, W., Profitlich, H.-J., & Rist, T. (1993). Plan-based integration of natural language and graphics generation. *Artificial Intelligence*, *63*(1–2), 387–428.

Wright, J. R., Weixelbaum, E. S., Vesonder, G. T., Brown, K. E., Palmer, S. R., Berman, J. I., & Moore, H. H. (1993). A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T network systems. *AI Magazine*, *14*(3), 69–80.