ELSEVIER

Artificial Intelligence 99 (1998) 209–260

Artificial Intelligence

A refined architecture for terminological systems: Terminology = Schema + Views

M. Buchheit [a,1], F.M. Donini [b,2], W. Nutt [c,3], A. Schaerf [b,*]

[a] *Interactive Objects Software GmbH (iO), Freiburg, Germany*

[b] *Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Roma, Italy*

[c] *German Research Center for Artificial Intelligence GmbH (DFKI), Saarbrücken, Germany*

Received June 1995; revised April 1997

Abstract

Traditionally, the core of a Terminological Knowledge Representation System (TKRS) consists of a TBox or terminology, where concepts are introduced, and an ABox or world description, 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. Two case studies show the computational benefits to be gained from the refined architecture. © 1998 Elsevier Science B.V.

Keywords: Domain modelling; Knowledge representation systems; Description logics; Knowledge-base architectures; Subumption of concepts

* Corresponding author. Email: aschaerf@dis.uniroma1.it.

[1] Email: buchheit@io.freinet.de.

[2] Email: donini@dis.uniroma1.it.

[3] Email: werner.nutt@dfki.de.

PII S0004-3702(97)00079-9

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 [5], BACK [31], LOOM [26], and KRIS [3], the development of algorithms (see e.g., [28]), and the investigation of the computational properties of inferences (see e.g., [17,29]).

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.[4]

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., [27]).

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, having a salary in HighSalary. Such introductions are similar to class declarations in object-oriented systems. For this purpose, a language with limited expressivity 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.

[4] In [24] a combination of a weak language for ABoxes and a strong language for queries has been investigated.

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 one 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 set of possible interpretations, first order semantics—also called descriptive semantics in this context (see [30])—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 degrees of freedom for improving systems, since language, form of statements, and semantics can be specified differently for schema and views. In fact, 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, which yields a more fine-grained picture of the computational complexity of reasoning.

Beside the proposal for a new architecture, the paper presents various technical results on the semantics and the complexity of terminological reasoning. First, we analyse the effect of fixpoint semantics for inclusion axioms and we provide some equivalence results for various semantics proposed in the literature [28,35]. Second, we provide complexity results on reasoning in presence of terminological cycles for three different schema languages under descriptive semantics. In particular, we prove that reasoning is polynomial in the basic language, whereas it is intractable for two of its extensions. For one of the two extension we identify a syntactic restriction that still allows for polynomial reasoning.

Moreover, we prove that in two systems, namely KRIS, and CONCEPTBASE [22], it is possible to add a cyclic schema without increasing the complexity of reasoning. As

Table 1
Syntax and semantics of concept forming constructs

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	$(\geqslant n\, R)$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \geqslant n\}$
	$(\leqslant n\, R)$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \leqslant n\}$

Table 2
Syntax and semantics of role 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	$\{(d_1, d_1) \mid d_1 \in \Delta^{\mathcal{I}}\}$

byproducts of this result, we also prove that instance checking in KRIS is in PSPACE (which was proven by Hollunder [18] only for a restricted language).

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 our architecture and in Section 4, schemas are considered in detail. In Section 5, we show by two 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* (or *individuals*) (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. Concepts in the schema language contain only schema names whereas concepts in the view language may contain 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.

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. Fig. 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 (remember that only schema names can appear in 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*, 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*

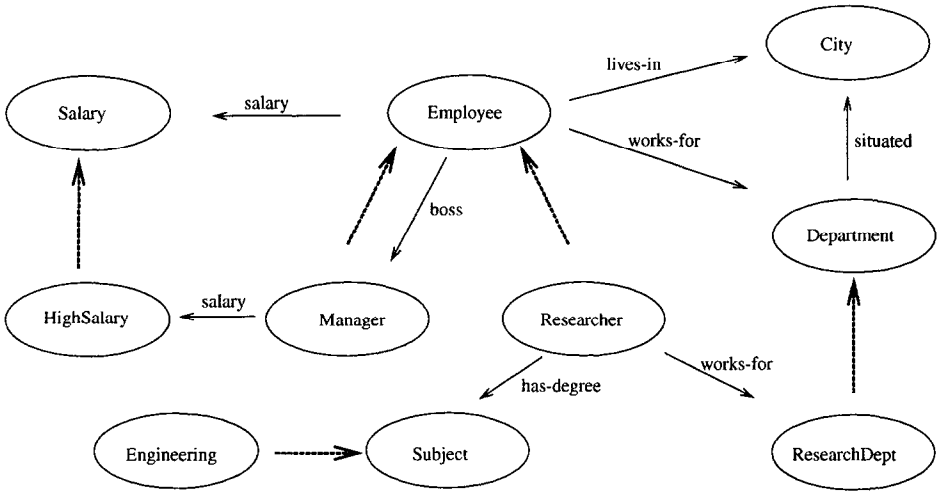

Fig. 1. Concepts and roles in the company environment.

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

Fig. 2. Schema axioms for the company environment.

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 Fig. 2 we give the schema axioms for the company example of Fig. 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 $\sqsubseteq$ ($\geqslant$ 1 salary) and Employee $\sqsubseteq$ ($\leqslant$ 1 salary).[5] The fact that every Manager is an Employee leads to the axiom Manager $\sqsubseteq$ Employee, and that a Manager must have a HighSalary to Manager $\sqsubseteq$ $\forall$salary.HighSalary.[6]

[5] Two axioms of the form $A \sqsubseteq (\leqslant 1\ P)$ and $A \sqsubseteq (\geqslant 1\ P)$ are abbreviated by $A \sqsubseteq (= 1\ P)$.

[6] 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.

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

A schema may contain *cycles* through inclusion axioms. So one may state that the bosses of an employee are themselves employees, writing "Employee $\sqsubseteq \forall$boss. 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 [30] 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 taxonomy* contains *view definitions* of the form

$$V \doteq C,$$

where V is a view name and C is a concept in the view language (remember that both schema and view names can appear in view concepts). 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 "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 $\mathcal{V}$.

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

$$a{:}\,C, \qquad aRb,$$

where C and R are concept and role descriptions in the view language. Intuitively, an assertion $a{:}\,C$ 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 $a{:}\,A$ 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* $a{:}\,C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ 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.

2.3. Reasoning services

There are several reasoning services that a terminological system must provide. We concentrate on the following as the basic ones.

- *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.
- *Schema subsumption*:
  Given a schema $\mathcal{S}$, and schema names A_1 and A_2, check whether $A_1^{\mathcal{I}} \subseteq A_2^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{S}$. This is written as $\mathcal{S} \models A_1 \sqsubseteq A_2$ or as $A_1 \sqsubseteq_{\mathcal{S}} A_2$.
- *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 $\Sigma \models a{:}\,V$.

Schema validation and schema subsumption support 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., [7]). Note that the schema $\mathcal{S}$ has to be taken into account in all 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 [29]). This is not possible when $\mathcal{S}$ and/or $\mathcal{V}$ are cyclic.

Notice that we do not consider the world description for subsumption problems. The explanation for this is twofold: First, in most of the languages we consider, the assertions play no role in determining the subsumption relation (see e.g., [29]), and

therefore the world description can be neglected in such cases. Second, even in the languages in which assertions do affect the subsumption relation (those languages containing references to individuals e.g., the singleton set constructor) we are interested in the static relationship between views, independently of the current state of the world description.

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 [37] 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 [1,29] and [17,32], respectively.

For a general description of the complexity classes we use, see [23].

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 are dealt with in Sections 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 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 extend this to role inclusions. There we 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 $D(\mathcal{V})$ 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 either $D(\mathcal{S})$ or $D(\mathcal{V})$ contains one.

To come up with a semantics for a terminology means to define which of its models should be considered for reasoning. Let us concentrate on definitions and 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 $i \neq j$. 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 there are no role definitions.

The problem when cycles are present is that an interpretation of the atomic concepts might be extendible to a model of the terminology in more than one way (see e.g., [28]). 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 [28] 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.

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 $\mathcal{T}_\mathcal{J}: (2^\Delta)^n \to (2^\Delta)^n$ by

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

where $\boldsymbol{O} := (O_1, \ldots, O_n)$ 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 $\boldsymbol{O} \in (2^\Delta)^n$ such that $\mathcal{T}_\mathcal{J}(\boldsymbol{O}) = \boldsymbol{O}$. Obviously, the interpretation defined by $\mathcal{J}$ and $\boldsymbol{O}$ is a model of $\mathcal{T}$ if and only if $\boldsymbol{O}$ is a fixpoint of $\mathcal{T}_\mathcal{J}$.

A mapping $T: D \to D$ on a complete lattice $(D, \leqslant)$ is called *monotonic* if $a \leqslant b$ implies $T(a) \leqslant T(b)$ 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., [25, Chapter 1, Section 5]). Let "$\leqslant$" be the componentwise subset ordering on $(2^\Delta)^n$. Since $((2^\Delta)^n, \leqslant)$ 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., [35]). 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 [1] 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 [1, p. 626]). In [30] 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 [28, p. 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 [12, 13] base their approach to the semantics of cycles on non-well-founded 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 [2] for the language $\mathcal{ALC}$ and in [6] 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.[7] $\mathcal{ALCNR}$ is the language of the system KRIS. For both $\mathcal{ALC}$ and $\mathcal{ALCNR}$

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

subsumption checking with cyclic definitions is EXPTIME-hard (for $\mathcal{ALC}$ it has been proven EXPTIME-complete), whereas the problem is PSPACE-complete for cycle-free terminologies.

An approach based on the μ-calculus was proposed independently by Schild [35] and De Giacomo and Lenzerini [11]. 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 [28] 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 [35] 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 inclusions $A \sqsubseteq D_1, \ldots, A \sqsubseteq D_m$ can be replaced by the single inclusion $A \sqsubseteq D_1 \sqcap \cdots \sqcap D_n$.

With $\bar{\mathcal{S}}$ we denote the transformation proposed by Nebel, with $\mathcal{S}^{\sqcap}$ the one proposed by Schild, and with $\mathcal{S}^{=}$ the one that replaces the inclusions by definitions, that is,

- $\bar{\mathcal{S}} := \{A_i \doteq \bar{A}_i \sqcap C_i \mid i \in 1..n\}$,
- $\mathcal{S}^{\sqcap} := \{A_i \doteq A_i \sqcap C_i \mid i \in 1..n\}$,
- $\mathcal{S}^{=} := \{A_i \doteq C_i \mid i \in 1..n\}$.

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 one 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 [1]). 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 is-deputy-of MARY and MARY is-deputy-of JOE, with the lfp-semantics, JOE and MARY cannot be Employees. Notice however that this is a problem of lfp-semantics in general, and not one of the specific transformation only and it shows that the approach of taking lfp-semantics is not acceptable in this situation.

Before we consider the combinations of gfp-semantics with the two transformations, we have to introduce some notations. Let $T: D \to D$ be a mapping on a complete lattice $(D, \leqslant)$. With $gfp(T)$ we denote the greatest fixpoint of T. Let X be a subset of D. With $lub\, X$ we denote the least upper bound of X. The next result is a weak form of the Proposition 5.1 in [25].

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

The following proposition, due to Schild [34], 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 $\mathcal{S}^{=}$ is 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. First, notice that if $\mathcal{S}^{=}$ is monotonic then $\mathcal{S}^{\sqcap}$ is monotonic. In fact, let $\mathcal{S} = \{A_i \sqsubseteq C_i \mid i \in 1..n\}$ and $\mathcal{I}_1$ and $\mathcal{I}_2$ two interpretations such that $A_i^{\mathcal{I}_1} \subseteq A_i^{\mathcal{I}_2}$ for $i \in 1..n$. If $\mathcal{S}^{=}$ is monotonic then $C_i^{\mathcal{I}_1} \subseteq C_i^{\mathcal{I}_2}$ for $i \in 1..n$. From the two set inclusions $A_i^{\mathcal{I}_1} \subseteq A_i^{\mathcal{I}_2}$ and $C_i^{\mathcal{I}_1} \subseteq C_i^{\mathcal{I}_2}$, it follows that $A_i^{\mathcal{I}_1} \cap C_i^{\mathcal{I}_1} \subseteq A_i^{\mathcal{I}_2} \cap C_i^{\mathcal{I}_2}$, proving that $\mathcal{S}^{\sqcap}$ is monotonic.

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 $((2^{\Delta})^n, \leqslant)$ is a complete lattice. With $\boldsymbol{C}^{\mathcal{I}}$ 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:

$$
\begin{aligned}
gfp(\mathcal{S}_{\mathcal{J}}^{\sqcap}) &= lub\{\boldsymbol{O} \mid \boldsymbol{O} \leqslant \mathcal{S}_{\mathcal{J}}^{\sqcap}(\boldsymbol{O})\} && (1)\\
&= lub\{\boldsymbol{O} \mid \boldsymbol{O} \leqslant \boldsymbol{O} \wedge \boldsymbol{C}^{\mathcal{I}}\} && (2)\\
&= lub\{\boldsymbol{O} \mid \boldsymbol{O} \leqslant \boldsymbol{C}^{\mathcal{I}}\} && (3)\\
&= lub\{\boldsymbol{O} \mid \boldsymbol{O} \leqslant \mathcal{S}_{\mathcal{J}}^{=}(\boldsymbol{O})\} && (4)\\
&= gfp(\mathcal{S}_{\mathcal{J}}^{=}). && (5)
\end{aligned}
$$

Eqs. (1) and (5) follow from the monotonicity of $\mathcal{S}^{\sqcap}$ and $\mathcal{S}^{=}$ and from Proposition 3.1, Eq. (2) and (4) follow by definition of the mappings $\mathcal{S}_{\mathcal{J}}^{\sqcap}$ and $\mathcal{S}_{\mathcal{J}}^{=}$, respectively, and Eq. (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

$$\mathcal{S} = \{\mathsf{Horse} \sqsubseteq \forall \mathtt{child}.\mathsf{Horse}, \mathsf{Human} \sqsubseteq \forall \mathtt{child}.\mathsf{Human}\},$$

horses and humans 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.

The following theorem summarizes our results on gfp-semantics for the transformations proposed by Nebel and Schild.

Theorem 3.4. *Let $\mathcal{S}$ be a set of inclusion axioms and A_1, A_2 two schema names.*
(i) *If $\mathcal{S}^=$ is monotonic, then $\mathcal{S}^\sqcap$ and $\mathcal{S}^=$ are equivalent under gfp-semantics.*
(ii) *$\bar{\mathcal{S}} \models A_1 \sqsubseteq A_2$ under gfp-semantics if and only if $\mathcal{S} \models A_1 \sqsubseteq A_2$ under descriptive semantics.*

In conclusion, 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 "BillsBosses $\doteq \exists \mathsf{boss}^{-1}.\{\mathsf{BILL}\}$" and "BillsSuperBosses $\doteq$ BillsBosses$\sqcup \exists \mathsf{boss}^{-1}.$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 boss there may be several ways to interpret 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 frequently 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 [7], which is defined by the syntax rule

$$D \longrightarrow A \mid \forall P.A \mid (\geqslant 1\, P) \mid (\leqslant 1\, P).$$

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 individuals 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. 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 $\mathcal{S}$.

An $\mathcal{SL}$-schema may entail non-obvious subsumptions. For example, from the schema

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

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 already follow from the statements about inclusions, i.e., if $A_1 \preceq_{\mathcal{S}} A_2$, whenever $\mathcal{S}$ contains axioms $P \sqsubseteq A_2 \times B$ and $A_1 \sqsubseteq (\geqslant 1\ P)$. Obviously, verifying whether a schema is isa-complete takes polynomial time. Moreover, an arbitrary schema can be transformed in polynomial time into an equivalent schema that is isa-complete.[8]

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

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

Proof. This is a consequence of Proposition 4.15. □

We conclude that subsumption of schema names with respect to 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 Fig. 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

[8] Two sets of axioms are equivalent if they have the same models.

$A \sqsubseteq \neg B$.

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

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

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

Proof. See the Appendix. □

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

We show that deciding the validity of $\mathcal{SL}_{\text{dis}}$-schemas is co-NP-hard. The proof consists in a reduction of the satisfiability problem for concepts of the language $\mathcal{ALE}$ (see [36]), 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 [14] 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 [14] reveals, more specifically, that satisfiability is still co-NP-complete for *restricted* $\mathcal{ALE}$-concepts C, which satisfy the following properties:

(i) only one role symbol occurs in C;
(ii) no concept name other than $\top$ and $\bot$ occurs in C;
(iii) there is exactly one occurrence of $\bot$ in C;
(iv) 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 $\mathcal{SL}_{\text{dis}}$-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. Then we enter into the schema $\mathcal{S}_C$ the axiom $A^+ \sqsubseteq \neg A^-$, and furthermore, for every subconcept D of C the following axioms:

(1) $A_D \sqsubseteq A_{D'}$, $A_D \sqsubseteq A_{D''}$, if $D = D' \sqcap D''$;
(2) $A_D \sqsubseteq (\geqslant 1\, P_D)$, $A_D \sqsubseteq \forall P_D.A_{D'}$, 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 $P \in \mathcal{P}_C$, 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 $\mathcal{SL}_{\text{dis}}$-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 the Appendix. □

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

4.2.2. An algorithm for reasoning about $\mathcal{SL}_{\text{dis}}$-schemas

Next we describe an algorithm for deciding the local validity of an $\mathcal{SL}_{\text{dis}}$-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 $A' \sqsubseteq (\geqslant 1\, P) \in \mathcal{S}$. 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 $\mathcal{SL}_{\text{dis}}$-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 C to C' if
  - P is necessary on A for some $A \in C$, and
  - $C' = \text{range}(P, C)$.

A node C is a *conflict node* if there are $A, B \in C$ 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 $C = \{A_1, \ldots, A_m\}$ 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 C to $C' = \{B_1, \ldots, B_n\}$, 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 C' is the range of P on C). The set C' might be the empty set. But then there is no edge going out of 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 isa-complete $\mathcal{SL}_{\text{dis}}$-schema and $C = \{A_1, \ldots, A_m\}$. Then $A_1 \sqcap \cdots \sqcap A_m$ is $\mathcal{S}$-unsatisfiable if and only if there is a path in $\mathcal{G}_{\mathcal{S}}$ from C to a conflict node.*

Proof. See the Appendix. □

By Lemma 4.8, $A_1 \sqcap \cdots \sqcap A_m$ is not $\mathcal{S}$-satisfiable if and only if there is a path in $\mathcal{G}_{\mathcal{S}}$ from $C = \{A_1, \ldots, A_m\}$ to some conflict node 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 [21]), 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 $\mathcal{SL}_{\text{dis}}$-schema $\mathcal{S}$ and schema names $A_1, \ldots, A_m$ whether the conjunction $A_1 \sqcap \cdots \sqcap A_m$ 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 $\mathcal{SL}_{\text{dis}}$-schemas is in* PSPACE *and* co-NP-*hard.*

4.2.3. Cycles in $\mathcal{SL}_{\text{dis}}$-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 $\mathcal{SL}_{\text{dis}}$-schemas and adopt it also to role inclusions. Then we identify a class of cycles that increases the complexity of reasoning about $\mathcal{SL}_{\text{dis}}$-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 $(\geqslant 1\, P) \sqsubseteq A_1, \top \sqsubseteq \forall P.A_2$.[9] 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 $A \sqsubseteq (\geqslant 1\, P)$, since $(\geqslant 1\, P) \sqsubseteq A_1$ holds.

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 A_1, say, while keeping essentially the same meaning. We extend the definition of the dependency graph 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 $A_1 \sqsubseteq (\geqslant 1\, P)$ 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 a sequence of isa-edges 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 $C_0, C_1, \ldots, C_n$ is a path in the schema graph $\mathcal{G}_{\mathcal{S}}$ of an all-cycle-free schema $\mathcal{S}$, then any two distinct sets C_i, 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 non-

[9] An interpretation $\mathcal{I}$ satisfies an arbitrary inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

deterministic 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 $\mathcal{SL}_{\text{dis}}$-schema. Then deciding whether a conjunction $A_1 \sqcap \cdots \sqcap A_m$ of schema names is $\mathcal{S}$-unsatisfiable 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 $\mathcal{SL}_{\text{dis}}$-schemas is* co-NP-*complete.*

4.2.4. Subsumption in $\mathcal{SL}_{\text{dis}}$

Deciding subsumption of schema names with respect to an $\mathcal{SL}_{\text{dis}}$-schema $\mathcal{S}$ cannot be easier than checking unsatisfiability: $A_1 \sqcap \cdots \sqcap A_m$ is $\mathcal{S}$-unsatisfiable iff for any name B not occurring in $\mathcal{S}$ we have $A_1 \sqcap \cdots \sqcap A_m \sqsubseteq_{\mathcal{S}} B$. 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 isa-complete $\mathcal{SL}_{\text{dis}}$-schema and $A, A_1, \ldots, A_m$, be schema names. Suppose that $A_1 \sqcap \cdots \sqcap A_m$ is $\mathcal{S}$-satisfiable. Then $A_1 \sqcap \cdots \sqcap A_m \sqsubseteq_{\mathcal{S}} A$ 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 $A_1 \sqcap \cdots \sqcap A_m \sqsubseteq_{\mathcal{S}} A$.

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

Corollary 4.16. *Subsumption of conjunctions of schema names with respect to $\mathcal{SL}_{\text{dis}}$-schemas is* NP-*hard. For arbitrary $\mathcal{SL}_{\text{dis}}$-schemas this problem can be decided using polynomial space. For all-cycle free $\mathcal{SL}_{\text{dis}}$-schemas, there is a nondeterministic polynomial time algorithm.*

Proof. NP-hardness holds because unsatisfiability of schema names with respect to $\mathcal{SL}_{\text{dis}}$-schemas is co-NP-hard (see Construction 4.4 and Lemma 4.5).

An algorithm can decide whether $A_1 \sqcap \cdots \sqcap A_m$ is subsumed by $B_1 \sqcap \cdots \sqcap B_n$ by first checking whether $A_1 \sqcap \cdots \sqcap A_m$ is $\mathcal{S}$-unsatisfiable. If so, it returns "yes". Otherwise, it checks whether for every B_j there is an A_i such that $A_i \preceq_{\mathcal{S}} B_j$. Clearly, the second check can be in polynomial time. Now, the upper bounds follow from the upper bounds for checking $\mathcal{S}$-satisfiability. □

4.2.5. Dichotomic schemas

We now investigate a restricted class of $\mathcal{SL}_{\text{dis}}$-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 either $A' \sqsubseteq (\geqslant 1\, P) \in \mathcal{S}$, or $A' \sqsubseteq (\leqslant 1\, P) \in \mathcal{S}$, 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 $A \sqsubseteq (\geqslant 1\, P)$, 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 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 4.17. For every $\mathcal{SL}_{\text{dis}}$-schema $\mathcal{S}$ the *dichotomic schema graph* $\mathcal{D}_{\mathcal{S}}$ is defined as follows:

- the *nodes* are sets $\{A, B\}$ consisting of one or two concept names occurring in $\mathcal{S}$ (note that A, B need not be distinct);
- there is an *edge* with label P from $\{A, B\}$ to $\{A', B'\}$ if
  - P is necessary on $dom(P)$,
  - $A \xrightarrow{P}_{\mathcal{S}} A'$ and $B \xrightarrow{P}_{\mathcal{S}} B'$, and
  - A, $B \preceq_{\mathcal{S}} dom(P)$

  (note that this definition also captures the case that $A = B$ or $A' = B'$).

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 4.18. *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 the Appendix. □

Corollary 4.19. *Let $\mathcal{S}$ be a dichotomic schema and $A_1, \ldots, A_m$ be concept names. Then the following are equivalent:*

(i) *$A_1 \sqcap \cdots \sqcap A_m$ is not $\mathcal{S}$-satisfiable;*
(ii) *there are A_i, A_j such that $A_i \sqcap A_j$ is not $\mathcal{S}$-satisfiable;*
(iii) *there are A_i, A_j such that a conflict node in $\mathcal{D}_{\mathcal{S}}$ is reachable from $\{A_i, A_j\}$.*

Corollary 4.20. *For dichotomic schemas, satisfiability and subsumption of conjunctions of concept names can be decided in polynomial time.*

4.2.6. Related work

Theorem 3.4 tells us that subsumption with respect to to a schema $\mathcal{S}$, that is a set of inclusion axioms, under descriptive semantics can be reduced to subsumption with respect to to the set of definitions $\bar{\mathcal{S}}$ under greatest fixpoint semantics. Baader [1] and Nebel [29,30] have investigated the subsumption problem with respect to sets of—possibly cyclic—concept definitions under various semantics, among them gfp-semantics. They determined the complexity of subsumption and developed algorithms. This could suggest to make use of their techniques to reason about schemas.

In addition, their technical approach looks similar to ours. They reduce the subsumption problem under gfp-semantics to inclusion problems for regular languages. To do so, they construct for a given set of definitions a nondeterministic automaton where the states are the names of concepts and where the transitions between states are marked with role symbols. The regular languages in question are defined in terms of the automaton.

The automaton contains a cycle if and only if the terminology contains a cycle. By a translation of well-known results from automata theory, this yields PSPACE-completeness of subsumption in general terminologies and co-NP-completeness in the case of acyclic terminologies.

At first glance, the method strongly resembles ours. Evidently, automata can be viewed as directed graphs with labeled arcs. In particular, constructing a deterministic out of a nondeterministic automaton by the powerset technique is very similar to constructing a schema graph out of a schema.

However, it turns out that the languages for which Baader and Nebel give algorithms are virtually subsets of $\mathcal{SL}$. Also, they do not consider role inclusions as we do. While reasoning problems for schemas become difficult when disjointness is added to $\mathcal{SL}$, reasoning about $\mathcal{SL}$-schemas is almost trivial: schema names and conjunctions of schema names are always satisfiable, and all subsumption relations can be derived by transitivity from those given explicitly. This is different if inclusions are replaced by definitions: reasoning is costly already for very limited languages.

A further difference is that in the present work the connection between schemas and automata is not so straightforward as in the work by Baader and Nebel. In their case, concepts of the form $(\geqslant 1\, P)$ do not have a particular impact on subsumption:

(i) the subsumption problem is not any more difficult for schemas where such concepts occur than it is for schemas without, and the algorithms are essentially the same;

(ii) only concepts of the form $\forall P.B$ are relevant for the existence of transitions in their automata.

In $\mathcal{SL}_{\text{dis}}$-schemas, instead, the interplay between existential quantification in expressions $(\geqslant 1\, P)$ and universal quantification in expressions $\forall P.B$ is the primary source of complexity. It is not too difficult to see that for schemas that either contain no inclusion of the type $A \sqsubseteq (\geqslant 1\, P)$ or no inclusion of the form $A \sqsubseteq \forall P.B$ satisfiability and subsumption are polynomial.

Finally, the complexity results look similar but, in fact, are different. For acyclic schemas, we have shown that unsatisfiability—and therefore subsumption—is NP-complete. For acyclic terminologies, Baader and Nebel have shown that subsumption is co-NP-complete.

Summarizing, the two lines of research concentrated on different aspects of reasoning about terminologies. While Baader and Nebel studied the impact of definitions on reasoning, our work deals with reasoning in the presence of constraints on classes.

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 employs be a shorthand for works-for^{-1}. Then with the axiom ResearchDept $\sqsubseteq \forall$employs.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 with respect to $\mathcal{SL}_{\text{dis}}$-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 with an example. Consider the following fragment of the company schema:

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

Although the schema is isa-complete and Researcher $\preceq_{\mathcal{S}}$ Employee does *not* hold, it entails that Researcher is subsumed by Employee. Suppose that JOE is an arbitrary researcher. Then JOE works for some research department, say D007. Since research departments only employ employees, every individual employed by 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 $\forall P^{-1}.A$, $(\geqslant 1\, P^{-1})$ and $(\leqslant 1\, P^{-1})$. In this subsection, we prove that subsumption of concept names with respect to $\mathcal{SL}_{\text{inv}}$-schemas is NP-hard. Moreover, we show that for $\mathcal{SL}_{\text{inv}}$-schemas there is a difference between reasoning with respect to all models and reasoning with respect to all *finite* models.

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

We construct for every restricted $\mathcal{ALE}$-concept C an $\mathcal{SL}_{\text{inv}}$-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 level 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 Section 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 4.21. *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 the Appendix. □

Theorem 4.22. *Subsumption of concept names with respect to $\mathcal{SL}_{\text{inv}}$-schemas is* NP-*hard.*

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

As an upper bound, we have that reasoning in $\mathcal{SL}_{\text{inv}}$ is in EXPTIME. This follows from the same bound given in [10] for reasoning in the language $\mathcal{ALUNI}$, of which $\mathcal{SL}_{\text{inv}}$ is a sublanguage.

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

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

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 with respect to finite models has been investigated in [9,10]. 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 two systems by the language $\mathcal{SL}$ for cyclic schemas. The view languages are derived from two implemented systems described in the literature, namely KRIS [3] and CONCEPTBASE [22].

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 both 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 [29] 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 two systems stand for two different design paradigms (see [4]). Thus each case study emphasizes a different aspect of the benefits that can be gained from our proposal.

V_1 = Researcher $\sqcap$ $\forall$has-degree.Engineering

V_2 = Employee $\sqcap$ $\exists$has-degree.Engineering

Fig. 3. $\mathcal{ALCNR}$-views.

The system KRIS is built at DFKI and used in several applications as the knowledge representation component (see e.g., [38]). 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 [3]. 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. Instance checking was proved PSPACE-complete in [18] for the sublanguage of KRIS excluding role conjunction. As a byproduct, our proofs extend that result for the first time to the full language of KRIS.

CONCEPTBASE is a deductive object-oriented database system, which was developed 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 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 [7] 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 in one language 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:

$$C, D \longrightarrow A \mid \top \mid \bot \mid C \sqcap D \mid C \sqcup D \mid \neg C \mid \forall R.C \mid \exists R.C \mid (\geqslant n\, R) \mid (\leqslant n\, R)$$
$$R \longrightarrow P_1 \sqcap \cdots \sqcap P_k$$

The language $\mathcal{ALCNR}$, first introduced in [20], allows one to express intersection, union, and complement of concepts, universal and existential quantification on roles,

number restrictions and role conjunction. Fig. 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 Fig. 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 $\langle \mathcal{S}, \mathcal{W} \rangle \models a: D$, 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 [36] first used such a calculus for the language $\mathcal{ALC}$ which is a sublanguage of $\mathcal{ALCNR}$ that allows to express neither 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

The $\mathcal{ALCNR}$-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 distinct old individuals appearing in $\mathcal{W}$.

The following proposition is an immediate consequence of the above definitions. It shows that for developing algorithms one can concentrate on knowledge base satisfiability.

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

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

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

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

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

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

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

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

Fig. 4. The schema rules for $\mathcal{ALCNR}$.

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 and a an old individual. Then:*

(i) $C \sqsubseteq_{\mathcal{S}} D$ *if and only if the knowledge base* $\langle \mathcal{S}, \{x\!:\!C \sqcap \neg D\} \rangle$ *is unsatisfiable.*

(ii) $\langle \mathcal{S}, \mathcal{W} \rangle \models a\!:\!D$ *if and only if the knowledge base* $\langle \mathcal{S}, \mathcal{W} \cup \{a\!:\!\neg D\} \rangle$ *is unsatisfiable.*

We assume that concepts are 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 [15].

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 $\mathcal{ALCNR}$. The input of the calculus is an $\mathcal{ALCNR}$-knowledge base $\Sigma = \langle \mathcal{S}, \mathcal{W} \rangle$. The completion rules add assertions to $\mathcal{W}$ until either a contradiction is generated or the knowledge base is recognized to be satisfiable.

The schema rules are presented in Fig. 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 $s\!:\!A$ and $t\!:\!B$ are not both in $\mathcal{W}$.

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

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

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

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

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

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

V4: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{sP_1 y, \ldots, sP_k y,\ y{:}\, C\} \cup \mathcal{W} \rangle$

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

V5: $\langle \mathcal{S}, \mathcal{W} \rangle \rightarrow \langle \mathcal{S}, \{sP_1 y_i, \ldots, sP_k y_i \mid i \in 1..n\} \cup \{y_i \neq y_j \mid i, j \in 1..n, i \neq j\} \cup \mathcal{W} \rangle$

if 1. $s{:}\, (\geqslant n\, R)$ is in $\mathcal{W}$,
2. $R = P_1 \sqcap \cdots \sqcap P_k$,
3. $y_1, \ldots, y_n$ are fresh new individuals, and
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. $s{:}\, (\leqslant n\, R)$ 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

Fig. 5. The view rules for $\mathcal{ALCNR}$.

Note that the schema rules treat axioms of the form $A \sqsubseteq (\geqslant 1\, P)$ 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 $s{:}\, \forall P.C$ or $s{:}\, (\geqslant 1\, P)$; this is done for termination and complexity considerations (see Section 5.1.2). Moreover, rule S2 is applied only if a corresponding assertion exists in $\mathcal{W}$ (condition 1). Hence, the number of new individuals generated because of schema axioms is bounded by the number of assertions in the world description.

Before providing the view rules, we introduce some more notation. Let $\mathcal{W}$ be a world description and $R = P_1 \sqcap \cdots \sqcap P_k$ $(k \geqslant 1)$ be a role. We then say that t is an *R-successor of s in $\mathcal{W}$* if $sP_1 t, \ldots, sP_k t$ are in $\mathcal{W}$.

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 Fig. 5. The rules V1 to V5 break up assertions into more elementary assertions. The rule V6 identifies individuals according to at-most restrictions.

The rules V2 and V6 can be applied in different ways to the same assertion, so that the result of their application depends on a nondeterministic choice. For this reason, we call them *nondeterministic* rules. Moreover, we call the rules S2, V6 and V5 *generating* rules, since they introduce new individuals into the world description. All other rules are called *nongenerating*.

If Σ and 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.

The next theorem shows 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 5.2 (Invariance). *Let Σ be an $\mathcal{ALCNR}$-knowledge base.*

(i) *Let Σ' be directly derived from Σ. If Σ' is satisfiable then Σ is satisfiable.*

(ii) *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 of the Invariance Theorem is mainly a rephrasing of the soundness of tableaux rules in first-order logic. A similar theorem was proved in [6] 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.

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.

In Section 5.1.2 we show that the completion process always terminates, i.e., it always reaches a completion, and that the satisfiability of a complete knowledge base can be decided very easily by looking for obvious contradictions, which we call clashes.

An $\mathcal{ALCNR}$-knowledge base $\langle \mathcal{S}, \mathcal{W} \rangle$ contains a *clash* if one of the following situations occurs:

(i) $s{:}\,\bot \in \mathcal{W}$, for some individual s,

(ii) $\{s{:}\,A,\ s{:}\,\neg A\} \subseteq \mathcal{W}$, for some individual s and some concept name A,

(iii) $\{s{:}\,(\leqslant n\,R)\} \cup \{sP_1t_i, \ldots, sP_kt_i \mid i \in 1..n+1\}$
$\cup \{t_i \neq t_j \mid i, j \in 1..n+1, i \neq j\} \subseteq \mathcal{W}$,
where $R = P_1 \sqcap \cdots \sqcap P_k$.

A clash is given by an evidently unsatisfiable set of assertions, hence any world description containing a clash is obviously unsatisfiable. The third 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.

Obviously, a knowledge base that contains a clash cannot be satisfiable. Consequently, if all completions contain a clash, then by the Invariance Theorem 5.2(ii) we know that the original knowledge base was unsatisfiable. If, however, one of the completions is clash-free then the corresponding canonical interpretation $\mathcal{I}_\Sigma$ (as defined below) is a model of the completion (see Proposition 5.4). Again by the Invariance Theorem 5.2(i) we conclude that the original knowledge base was satisfiable.

Given a complete knowledge base $\Sigma = \langle \mathcal{S}, \mathcal{W} \rangle$, we choose a new individual u not appearing in $\mathcal{W}$ and define the *canonical interpretation* $\mathcal{I}_\Sigma$ as follows:

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

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. As pointed out before, the schema rules are designed such that not every necessary role will get a new individual as filler. The purpose of u is to satisfy all axioms $A \sqsubseteq (\geqslant 1\, P)$ for those individuals s such that $s{:}\, A$ is in $\mathcal{W}$, but that have no P-successor in $\mathcal{W}$.

These considerations show that satisfiability of $\mathcal{ALCNR}$-knowledge bases is decidable, which we prove in the following subsection. In fact, we do more: Instead of just proving termination of the calculus, we prove the stronger result that there is a way of applying completion rules such that they always terminate *and* use just polynomial space with respect to combined complexity and polynomial time with respect to schema complexity. Thus, we have the following result:

Theorem 5.3. *With $\mathcal{SL}$ as schema language and $\mathcal{ALCNR}$ as view language, view subsumption and instance checking are* PSPACE*-complete with respect to combined complexity, and in* PTIME *with respect to schema complexity.*

5.1.2. Correctness and complexity of the $\mathcal{ALCNR}$-calculus

To prove Theorem 5.3, we first show that the canonical interpretation of a complete clash-free knowledge base is a model. Together with the Invariance Theorem this gives us the soundness of the $\mathcal{ALCNR}$-calculus. Then we turn to termination and complexity and thus show completeness.

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

Proof. See the Appendix. □

Before we consider termination and complexity, we need some more definitions, which are based on viewing the individuals in a world description as nodes in a graph, and assertions about roles as labeled arcs in this graph. 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" and *predecessor* its inverse.

For $\mathcal{ALCNR}$-knowledge bases, the "successor" relation restricted to new individuals forms a tree:

Proposition 5.5. *Let* $\langle \mathcal{S}, \mathcal{W}' \rangle$ *be derived from the initial knowledge base* $\langle \mathcal{S}, \mathcal{W} \rangle$. *Then no new individual* $\mathcal{W}'$ *is a direct successor of two different individuals.*

Proof. Obviously, in $\mathcal{W}$ there is no new individual. By an analysis of all rules of the $\mathcal{ALCNR}$-calculus it can be shown that the property of the proposition is invariant under rule applications. □

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 with respect to the size of Σ, hence the 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 of it 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.

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 an assertion $s:C$, or sPt (condition 1 of all rules). We exploit this property by saying that a rule is *applied to* the assertion $s:C$, or *applied to* the individual s (instead of saying that it is applied to the knowledge base $\langle \mathcal{S}, \mathcal{W} \rangle$).

We require that trace rules are applied using the following strategy:

(i) apply a rule to a new individual only if no rule is applicable to an old one;
(ii) apply a rule to a new individual x only if no rule is applicable to a new individual y such that y is a predecessor of x;
(iii) apply generating rules only if no nongenerating rule is applicable.

Using this strategy, 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 possibly further nongenerating rules are applied to s, one new individual y_i is (nondeterministically) 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, which plays the role of y_i before.

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 assertions about them are combined, which may lead to clashes that otherwise would not have occurred.

Trace rules for checking satisfiability of concepts without the presence of a schema for the language $\mathcal{ALC}$ were defined in [36], and were extended to more expressive languages in [15, 19, 20]. A polynomial-space algorithm that checks the satisfiability of an $\mathcal{ALCNR}$-concept C by generating all completions while keeping only one trace in memory at a time, is given in [15].

We now adapt those previous results to the presence of a schema, first for view subsumption and later on also for instance checking. 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:

$$depth(D) := \begin{cases} 0 & \text{if } D \text{ is of the form } A, \bot, \text{ or } \top, \\ 1 & \text{if } D \text{ is of the form } (\geqslant n\, R) \text{ or } (\leqslant n\, R), \\ depth(C) & \text{if } D \text{ is of the form } \neg C, \\ \max(depth(D_i)) & \text{if } D \text{ is of the form } D_1 \sqcap D_2 \text{ or } D_1 \sqcup D_2, \\ depth(C) + 1 & \text{if } D \text{ is of the form } \forall R.C \text{ or } \exists R.C. \end{cases}$$

The following proposition collects a number of properties concerning the depth of concepts and traces.

Proposition 5.6. *Let C be an $\mathcal{ALCNR}$-concept, x a new individual, and $\mathcal{W} = \{x{:}\, C\}$ 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:*

(i) *For every chain of direct successors $x, y_1, \ldots, y_h$ in a knowledge base derived from Σ, if $y_i{:}\, D$ is in $\mathcal{W}$, and $y_i{:}\, D$ has been added by the application of a view rule then $depth(D) \leqslant depth(C) - i$.*

(ii) *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).*

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

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

(v) *Every completion of Σ can be obtained as the union of finitely many traces.*

(vi) *Suppose $\Sigma' = \langle \mathcal{S}, \mathcal{W}' \rangle$ is a completion of 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 the Appendix. □

Now we can prove the main result about the complexity of view subsumption in $\mathcal{ALCNR}$-knowledge bases:

Proposition 5.7. *Let $\mathcal{S}$ be an $\mathcal{SL}$-schema, and C, D be $\mathcal{ALCNR}$-concepts. Then:*

(ii) *Checking $C \sqsubseteq_{\mathcal{S}} D$ can be done in polynomial space with respect to $|\mathcal{S}|$, $|C|$, and $|D|$;*

(ii) *Checking $C \sqsubseteq_{\mathcal{S}} D$ can be done in polynomial time with respect to $|\mathcal{S}|$.*

Proof. Combining Proposition 5.6 with Proposition 5.1(i), one directly proves that view subsumption can be checked in nondeterministic polynomial space with respect to combined complexity. In fact, in order to check that $\langle \mathcal{S}, \{x: C \sqcap \neg D\}\rangle$ is satisfiable, one generates a clash-free completion keeping in memory only one of its traces at a time.

A deterministic check must also explore all possible choices given by the assertions of the form $s: C_1 \sqcup C_2$ and $s: (\leqslant n\, R)$. Any time the most recent choice (e.g., $s: C_1$) having an alternative gives rise to at least one trace containing a clash, the alternative is taken (e.g., $s: C_2$), and all traces are recomputed for this new choice. The method is basically a rephrasing of the exploration of an AND-OR tree (AND nodes correspond to different traces to be generated, while OR nodes correspond to alternative choices). Since the nested choices to be kept in memory at one time are polynomially many (they cannot exceed the size of a trace), also this deterministic version of the method is in PSPACE. This proves Proposition 5.7(i).

The result on schema complexity of view subsumption (Proposition 5.7(ii)) is proved in two steps:

Step 1: We prove that both the number of traces in a completion of $\langle \mathcal{S}, \{x: C \sqcap D'\}\rangle$, where D' is the negation normal form of $\neg D$, and the number of completions depends 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 $s: (\leqslant 1\, P)$, which can be introduced by rule S5. 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 5.5), the number of individuals can be estimated by N^h, where N is the tree branching factor—the number of direct successors of an individual—and h is the tree depth. From Proposition 5.6(ii) and (iii), both h and N are bounded by $|C| + |D|$, which proves the claim.

Step 2: 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 Proposition 5.6(iv). Therefore, schema complexity is in PTIME, and more precisely in $O(|\mathcal{S}|)$. □

We now turn to to the problem of *instance checking*. For view subsumption we can suppose that the initial knowledge base contains only one assertion, namely an assertion of the form $x: C$ for an $\mathcal{ALCNR}$-concept C. Then, x is the root of the "tree of traces". This is no longer true for instance checking. Since the problem of checking whether a is an instance of C with respect to a knowledge base $\langle \mathcal{S}, \mathcal{W} \rangle$ is reduced to the satisfiability of the knowledge base $\langle \mathcal{S}, \mathcal{W} \cup \{a: \neg C\} \rangle$, the world description may contain arbitrary assertions.

The trace algorithm for subsumption of $\mathcal{ALC}$-concepts given by Schmidt-Schauß and Smolka [36] was extended by Baader and Hollunder [3] to solve instance checking in $\mathcal{ALC}$-world descriptions (see also [17]). The basic idea there is to reduce satisfiability of an arbitrary knowledge base to the satisfiability of a number of knowledge bases with only one individual. This is achieved by introducing projections.

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

In order to get a correct method, the basic idea has to be refined. Before considering projections, one has to make all properties of old individuals explicit. This is captured by the notion of a precompletion.

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. [10]

Now, for checking the satisfiability of a knowledge base Σ, one can examine each clash-free precompletion $\Sigma' = \langle \mathcal{S}, \mathcal{W}' \rangle$ of Σ, extract the various knowledge bases $\langle \mathcal{S}, \mathcal{W}'_x \rangle$ for all new individuals x appearing in $\mathcal{W}'$, and independently check them for satisfiability. The correctness of this method follows from the next 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 the Appendix. □

Intuitively, the above proposition states 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 for each individual independently.

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.*

[10] Notice that this notion of precompletion is different from the one given in [17].

Proof. See the Appendix. □

The size of precompletions is polynomially bounded by the size of the schema and the world description of the initial knowledge base:

Proposition 5.10. *Every precompletion of a knowledge base $\Sigma = \langle \mathcal{S}, \mathcal{W} \rangle$ has polynomial size with respect to Σ, and the number of individuals in it does not depend on $|\mathcal{S}|$.*

Proof. See the Appendix. □

Now we can prove the main result concerning instance checking in $\mathcal{ALCNR}$-knowledge bases:

Proposition 5.11. *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:*

(i) *Checking $\langle \mathcal{S}, \mathcal{W} \rangle \models a: D$ can be done in polynomial space with respect to $|\mathcal{S}|$, $|\mathcal{W}|$, and $|D|$;*

(ii) *Checking $\langle \mathcal{S}, \mathcal{W} \rangle \models a: D$ can be done in polynomial time with respect to $|\mathcal{S}|$.*

Proof. To check whether $\langle \mathcal{S}, \mathcal{W} \rangle \models a: D$, one has to check whether the knowledge base $\langle \mathcal{S}, \mathcal{W} \cup \{a: D'\} \rangle$, where D' is the negation normal form of $\neg D$, is unsatisfiable (Proposition 5.1 To this end, compute (nondeterministically) a clash-free precompletion Σ' of $\langle \mathcal{S}, \mathcal{W} \cup \{a: D'\} \rangle$ (this needs polynomial space by Proposition 5.10); 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 before. Again, this needs polynomial space. The deterministic version just keeps track of all backtracking points in applications of nondeterministic rules.

We now turn to the second point of the proposition. Let $\Sigma = \langle \mathcal{S}, \mathcal{W} \rangle$ be an $\mathcal{ALCNR}$-knowledge base. We prove the claim in four steps.

Step 1: The number of individuals in a precompletion does not depend on $|\mathcal{S}|$, by Proposition 5.10. 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 $s: (\leqslant n\, R)$ is bounded by $I(I-1)\cdots(n+1)$ (the number of sequences of elements of $1..I$ of length $I-n$), 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, this number is $O(1)$ with respect to $|\mathcal{S}|$.

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

- the maximal number of precompletions (a constant with respect to $|\mathcal{S}|$),
- the time to compute a precompletion (linear in $|\mathcal{S}|$ from Proposition 5.10),

- the number of new individuals in a precompletion (I, a constant with respect to $\mathcal{S}$),
- the 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 $O(|\mathcal{S}|^2)$. □

Summing up, by Propositions 5.7 and 5.11 we proved Theorem 5.3.

While the combined complexity of inferences in our case, with $\mathcal{ALCNR}$ as view language and $\mathcal{SL}$ as schema language, is PSPACE, using $\mathcal{ALCNR}$ also as the schema language raises the combined complexity to EXPTIME-hardness [34]. However, 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* CONCEPTBASE *as view language*

In [7] 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. 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.

Fig. 6 contains some examples of $\mathcal{QL}$ queries. Suppose we are given the schema of Fig. 2. Query Q_1 denotes all the managers and Q_2 all the employees 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

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

Fig. 6. $\mathcal{QL}$ queries.

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 5.12. *With $\mathcal{SL}$ as schema language and $\mathcal{QL}$ as view language, view subsumption and instance checking are in* PTIME *with respect to 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 [33]). Intuitively, the assertion $a{:}C$ 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 old individuals, then

$$\Phi(\mathcal{W}) := \sqcap_{(\alpha \in \mathcal{W})} \Phi(\alpha),$$

$$\Phi(a{:}C) := \exists Q.(\{a\} \sqcap C),$$

$$\Phi(aRb) := \exists Q.(\{a\} \sqcap \exists R.\{b\}),$$

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 5.13. *Given a schema $\mathcal{S}$, a world description $\mathcal{W}$, an old 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 $\Phi(\mathcal{W}) \sqcap \{a\} \sqsubseteq_{\mathcal{S}} C$.*

Proposition 5.13 can be proved analogously to Lemma 6.6 of [33].

A detailed proof of the view subsumption part of Theorem 5.12 can be found in [7]. But, since the proof requires techniques quite different from the ones used in the preceding case study, 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

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

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

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

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

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

if 1. $s{:}\, A_1$ 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 $s{:}\, A$ is in $\mathcal{F}$,
2. $A \sqsubseteq (\geqslant 1\, P)$ is in $\mathcal{S}$,
3. there is no t such that sPt is in $\mathcal{F}$, and $s{:}\, \exists P.C$ is in $\mathcal{G}$, and
4. y is a fresh new individual

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

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

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

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

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

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

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

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

Fig. 7. The decomposition, schema, goal, and composition rules.

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

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 s and transform C into a prototypical knowledge base where s 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 s 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 5.14. *Let $\mathcal{S}$ be an $\mathcal{SL}^-$-schema, C, D be $\mathcal{QL}^-$-concepts, and s be an individual. Then*

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

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 s and start with the knowledge base $\langle \mathcal{S}, \{s{:}\, C\}\rangle$ and the goal $\{s{:}\, D\}$. 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 $s{:}\, D$. This is a difference to the refutation style calculus of the first case study, where we start with the knowledge base $\langle \mathcal{S}, \{s{:}\, C, s{:}\, \neg D\}\rangle$ 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 Fig. 7. A rule is applicable to a pair if it satisfies the conditions associated with the rule and if the pair 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 $s{:}\, C \sqcap D$ *and* if either $s{:}\, C$ or $s{:}\, D$ is not in $\mathcal{F}$.

The *decomposition rules* (D1, D2) work on facts. They break up the initial fact $s{:}\, C$ 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 study. There the application is restricted to universally constrained individuals (see rule S2 in Fig. 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 $s{:}\, D$. The interesting rule is G2, since it relates goals to facts: if the goal is to find $s{:}\, \exists P.C$, then the only individuals tested are the ones 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 [7] 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 $s{:}\,D$ 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 $\langle \mathcal{S}, \{s{:}\,C\} \rangle \langle \{s{:}\,D\} \rangle$ 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 $\langle \mathcal{S}, \{s{:}\,C\} \rangle \langle \{s{:}\,D\} \rangle$ can be computed in time polynomial in the size of C, D and $\mathcal{S}$. This yields our claim.

Theorem 5.12 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. Section 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 two 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,[11] or we

[11] Recently, Calvanese [8] has determined the complexity of reasoning about schemas in various extensions of $\mathcal{SL}$, building on the results and techniques described in the present paper.

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 [16]).

Acknowledgements

This work has been partly funded by the Esprit Long Term Research Project No. 22469 "Foundations of Data Warehouse Quality (DWQ)", and by ASI (Agenzia Spaziale Italiana), CNR (Consiglio Nazionale delle Ricerche) and MURST (Ministero dell'Università e della Ricerca Scientifica e Tecnologica).

Appendix A. Proofs

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. □

Proof of Lemma 4.5. "⇒" 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 $\mathcal{I} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{I}})$ 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 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 $\mathcal{S}_C$ 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 $A_D \sqsubseteq (\geqslant 1\, P_D)$ 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 $A_D \sqsubseteq (\geqslant 1\, P_D)$ 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 $P^{\mathcal{I}} \subseteq Q^{\mathcal{J}}$. 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. □

Proof of Lemma 4.8. "$\Rightarrow$" Suppose there is a path $C_0, C_1, \ldots, C_k$ in $\mathcal{G}_{\mathcal{S}}$ from $C = C_0$ to some conflict node C_k. Then there are roles $P_1, \ldots, P_k$ such that P_i is necessary on some concept in C_{i-1}, and $C_i = \mathit{range}(P_i, C_{i-1})$. Obviously, $C_i \neq \emptyset$ 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 $d \in A_1^{\mathcal{I}} \cap \cdots \cap A_m^{\mathcal{I}}$. We show by induction that for every $i \in 0..k$ we have $\bigcap_{A \in C_i} A^{\mathcal{I}} \neq \emptyset$. The claim for $i = 0$ coincides with our assumption. Suppose that $d_{i-1} \in A^{\mathcal{I}}$ for every $A \in C_{i-1}$. Since P_i is necessary on some $A \in C_{i-1}$, there exists an element d_i such that $(d_{i-1}, d_i) \in P_i^{\mathcal{I}}$. Moreover, for every $B \in C_i$ we have $d_i \in B^{\mathcal{I}}$, since there is a transition $A \xrightarrow{P_i}_{\mathcal{S}} B$ for some $A \in C_{i-1}$. It follows that $d_k \in \bigcap_{B \in C_k} B^{\mathcal{I}}$, which is impossible because C_k is a conflict node.

"$\Leftarrow$" Suppose that no conflict node is reachable by a path issuing from 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 C. For a concept name A we define

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

For a role P we define

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

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

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

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 $C' \in A^\mathcal{I}$. Then there exists some $A' \preceq_\mathcal{S} A$ with $A' \in C'$.

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

Suppose that $A \sqsubseteq (\geqslant 1\, P) \in \mathcal{S}$. Then P is necessary on A'. With $C'' := range(P, C')$ we have

(i) (C', C'') is an edge in $\mathcal{G}_\mathcal{S}$,

(ii) $C'' \in \Delta^\mathcal{I}$, and

(iii) $(C', C'') \in P^\mathcal{I}$.

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

Suppose that $A \sqsubseteq (\leqslant 1\, P) \in \mathcal{S}$. 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 $C' \in B^\mathcal{I}$. Then there is some $B' \preceq_\mathcal{S} B$ with $B' \in C'$. This implies that C' is a conflict node, which is impossible, since $\Delta^\mathcal{I}$ contains only nodes reachable from C, and no conflict node can be reached from C. □

Proof of Lemma 4.18. "⇒" Suppose there is a path $C_0, C_1, \ldots, C_k$ in $\mathcal{G}_\mathcal{S}$ from $C_0 = \{A_1, \ldots, A_m\}$ to some conflict node C_k. We show that there is a path $\tilde{C}_0, \tilde{C}_1, \ldots, \tilde{C}_k$ in $\mathcal{D}_\mathcal{S}$ such that $\tilde{C}_l \subseteq C_l$ for $l \in 0..k$ and $\tilde{C}_k$ is a conflict node. This yields the claim because $\tilde{C}_0 \subseteq C_0 = \{A_1, \ldots, A_m\}$ and, since $\tilde{C}_0$ is a node in $\mathcal{D}_\mathcal{S}$, it is nonempty and has at most two elements. We proceed by induction on $k - l$.

Base case: Since C_k is a conflict node in $\mathcal{G}_\mathcal{S}$, there are names $B_k, \tilde{B}_k \in C_k$ 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 $B'_k \sqsubseteq \neg \tilde{B}_k \in \mathcal{S}$. Hence, $\tilde{C}_k := \{B_k, \tilde{B}_k\}$ is a conflict node in $\mathcal{D}_\mathcal{S}$.

Inductive case: Suppose that $\tilde{C}_l = \{B_l, \tilde{B}_l\} \subseteq C_l$ has already been defined and that the edge from C_{l-1} to C_l is labeled with P. By definition of $\mathcal{G}_\mathcal{S}$, the role P is necessary on some $\bar{C} \in C_{l-1}$. Hence, P is necessary on $dom(P)$ because $\mathcal{S}$ is dichotomic. Also there are transitions $C \xrightarrow{P}_\mathcal{S} B_l$, $\tilde{C} \xrightarrow{P}_\mathcal{S} \tilde{B}_l$ in $\mathcal{S}$ for some $C, \tilde{C} \in C_{l-1}$.

Let us say that a transition $A \xrightarrow{P}_\mathcal{S} B$ is *proper* if there is an A' with $A \preceq_\mathcal{S} A'$ and $A' \sqsubseteq \forall P.B \in \mathcal{S}$, and *improper* otherwise. Note that in a dichotomic schema $\mathcal{S}$ we have $A \preceq_\mathcal{S} dom(P)$ if $A \xrightarrow{P}_\mathcal{S} B$ is proper and that $B = cod(P)$ if $A \xrightarrow{P}_\mathcal{S} B$ is improper.

We distinguish three cases:

(i) the two transitions $C \xrightarrow{P}_{\mathcal{S}} B_l$, $\tilde{C} \xrightarrow{P}_{\mathcal{S}} \tilde{B}_l$ are both proper;
(ii) one transition is proper and the other is not;
(iii) none of the two transitions is proper.

In case (i), we define $\tilde{\mathcal{C}}_{l-1} := \{C, \tilde{C}\}$. The properness of the two transitions implies that there is a P-edge in $\mathcal{D}_{\mathcal{S}}$ from $\tilde{\mathcal{C}}_{l-1}$ to $\tilde{\mathcal{C}}_l$.

In case (ii), since $C \xrightarrow{P}_{\mathcal{S}} B_l$ is proper, $C \preceq_{\mathcal{S}} dom(P)$, and $\tilde{B}_l = cod(P)$, because $\tilde{C} \xrightarrow{P}_{\mathcal{S}} \tilde{B}_l$ is improper. Hence, $\mathcal{S}$ contains also the transition $C \xrightarrow{P}_{\mathcal{S}} \tilde{B}_l$. Thus, there is an edge with label P from $\tilde{\mathcal{C}}_{l-1} := \{C\}$ to $\tilde{\mathcal{C}}_l$.

In case (iii), since both transitions are improper, $B_l = \tilde{B}_l = cod(P)$. From the fact that P is necessary on $\bar{C} \in \mathcal{C}_{l-1}$ it follows, because of the dichotomy of $\mathcal{S}$, that $\bar{C} \preceq_{\mathcal{S}} dom(P)$. Hence, $\mathcal{S}$ contains the transitions $\bar{C} \xrightarrow{P}_{\mathcal{S}} B_l$ and $\bar{C} \xrightarrow{P}_{\mathcal{S}} \tilde{B}_l$, which yields a P-edge in $\mathcal{D}_{\mathcal{S}}$ from $\tilde{\mathcal{C}}_{l-1} := \{\bar{C}\}$ to $\tilde{\mathcal{C}}_l$.

Summarizing, we have shown that in each of the three cases there is a nonempty set $\tilde{\mathcal{C}}_{l-1} \subseteq \mathcal{C}_{l-1}$ with $|\tilde{\mathcal{C}}_{l-1}| \leqslant 2$ from which there is an edge to $\tilde{\mathcal{C}}_l$ in $\mathcal{D}_{\mathcal{S}}$.

"$\Leftarrow$" Suppose there is a path

$$\{B_0, \tilde{B}_0\} \xrightarrow{P_1}_{\mathcal{S}} \{B_1, \tilde{B}_1\}, \ \ldots, \ \{B_{k-1}, \tilde{B}_{k-1}\} \xrightarrow{P_k}_{\mathcal{S}} \{B_k, \tilde{B}_k\}$$

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 $\mathcal{C}_l := range(P_l, \mathcal{C}_{l-1})$ for $l \in 1..k$. 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, and $B_l, \tilde{B}_l \preceq_{\mathcal{S}} dom(P)$, $\mathcal{C}_{l-1}$ 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 4.21. "$\Rightarrow$" 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 $A_C^{\mathcal{J}} \neq \emptyset$. 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 $\bar{\mathcal{S}}_C \not\models A_C \sqsubseteq A_0$.

"$\Leftarrow$" Suppose that $\bar{\mathcal{S}}_C \not\models A_C \sqsubseteq A_0$. Then there is a model $\mathcal{J} = (\Delta, \cdot^{\mathcal{J}})$ of $\bar{\mathcal{S}}_C$ and an element $d_0 \in \Delta$ such that $d_0 \in A_C^{\mathcal{J}}$, but $d_0 \notin A_0^{\mathcal{J}}$. We inductively define a sequence of interpretations $\mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_n$ on the same domain as $\mathcal{J}$ such that $\mathcal{I} := \mathcal{I}_n$ is a model of $\mathcal{S}_C$. The construction is such that for each concept name A and role name P we have $A^{\mathcal{I}_{j-1}} \subseteq A^{\mathcal{I}_j} \subseteq A^{\mathcal{J}}$ and $P^{\mathcal{I}_{j-1}} \subseteq P^{\mathcal{I}_j} \subseteq P^{\mathcal{J}}$ for all $j \in 1..n$.

Base case: We define $\mathcal{I}_0$ by $A_C^{\mathcal{I}_0} := \{d_0\}$ and by interpreting all other concept and role names as the empty set.

Inductive case: Suppose $\mathcal{I}_{j-1}$ has been defined. We distinguish three cases:

(i) *$\mathcal{S}_C$ contains an axiom $A \sqsubseteq B$ and and there is some $d \in A^{\mathcal{I}_{j-1}} \setminus B^{\mathcal{I}_{j-1}}$.* If this condition holds, define $B^{\mathcal{I}_j} := B^{\mathcal{I}_{j-1}} \cup \{d\}$ and let $\cdot^{\mathcal{I}_j}$ and $\cdot^{\mathcal{I}_{j-1}}$ coincide on the other names in $\mathcal{S}_C$.

(ii) *$\mathcal{S}_C$ contains an axiom $A \sqsubseteq (\geqslant 1\, P)$ and there is some $d \in A^{\mathcal{I}_{j-1}}$ such that there does not exist a d' with $(d, d') \in P^{\mathcal{I}_{j-1}}$.* If this condition holds, since $\bar{\mathcal{S}}_C$ contains the same axiom, $\mathcal{J}$ is a model of $\bar{\mathcal{S}}_C$, and $A^{\mathcal{I}_{j-1}} \subseteq A^{\mathcal{J}}$, there is some d' with $(d, d') \in P^{\mathcal{J}}$. Define $P^{\mathcal{I}_j} := P^{\mathcal{I}_{j-1}} \cup \{(d, d')\}$ and let $\cdot^{\mathcal{I}_j}$ and $\cdot^{\mathcal{I}_{j-1}}$ coincide on the other names in $\mathcal{S}_C$.

(iii) *$\mathcal{S}_C$ contains an axiom $A \sqsubseteq \forall P.B$, $A^+ \neq B \neq A^-$, and there are c, c' such that $c \in A^{\mathcal{I}_j}$, $(c, c') \in P^{\mathcal{I}_j}$, but $c' \notin B^{\mathcal{I}_j}$.* If this condition holds, define $B^{\mathcal{I}_j} := B^{\mathcal{I}_{j-1}} \cup \{c'\}$ and let $\cdot^{\mathcal{I}_j}$ and $\cdot^{\mathcal{I}_{j-1}}$ coincide on the other names in $\mathcal{S}_C$.

Note that similarly to case (ii), in the cases (i) and (iii) $\bar{\mathcal{S}}_C$ contains the axioms in question, too. Thus, from the fact that $\mathcal{J}$ is a model of $\bar{\mathcal{S}}_C$, we conclude that $B^{\mathcal{I}_j} \subseteq B^{\mathcal{J}}$ and hence the interpretation of a symbol under $\mathcal{I}_j$ is a subset of the interpretation under $\mathcal{J}$.

Since $\mathcal{S}_C$ does not contain a cycle, the construction process terminates with an interpretation $\mathcal{I}_n$. Let $\mathcal{I} := \mathcal{I}_n$. We show that $\mathcal{I}$ is a model of $\mathcal{S}_C$.

By construction, $\mathcal{I}$ satisfies all axioms of the form $A \sqsubseteq B$, $A \sqsubseteq (\geqslant 1\, P)$, and $A \sqsubseteq \forall P.B$ where $A^+ \neq B \neq A^-$. Also by construction, we have $(A^+)^{\mathcal{I}} = (A^-)^{\mathcal{I}} = \emptyset$. Hence, $\mathcal{I}$ satisfies $A^+ \sqsubseteq A^-$. It remains to show that for all $P \in \mathcal{P}_C$, the interpretation $\mathcal{I}$ satisfies the axioms $A_D \sqsubseteq \forall P.A^+$, $A_D \sqsubseteq \forall P.A^-$, where $D = \forall Q.\bot$. Since $(A^+)^{\mathcal{I}} = (A^-)^{\mathcal{I}} = \emptyset$, we have to show that no element of $A_D^{\mathcal{I}}$ has a filler for any $P \in \mathcal{P}_C$.

Assume, on the contrary, that there is a $d \in A_D^{\mathcal{I}}$ and a role $P \in \mathcal{P}_C$ such that $(d, d') \in P^{\mathcal{I}}$. Since $A_D^{\mathcal{I}} \subseteq A_D^{\mathcal{J}}$, we have $d \in A_D^{\mathcal{J}}$. The schema $\bar{\mathcal{S}}_C$ contains the axioms $A_D \sqsubseteq \forall P.A_k$. Hence, $d' \in A_k{}^{\mathcal{J}}$.

By induction, it can be shown that for each subconcept E of C, if $c \in E^{\mathcal{I}}$, then c is reachable in $\mathcal{I}$ from d_0 by a chain of length $\lambda(E)$. Thus, d is reachable in $\mathcal{I}$ from d_0 by a chain of length $\lambda(D)$.

We have $\lambda(\bot) = k$, which imples $\lambda(D) = k - 1$. Thus, there is a chain $d_0, d_1, \ldots, d_{k-2}, d$. This chain can be extended to a chain of length k from d_0 to d'. Also, $P^{\mathcal{I}} \subseteq P^{\mathcal{J}}$ for all $P \in \mathcal{P}_C$, so that $d_0, d_1, \ldots, d, d'$ is a chain of length k in $\mathcal{J}$, too. Now, since $\mathcal{J}$ satisfies the axioms $A_i \sqsubseteq \forall P^{-1}.A_{i-1}$ for $i \in 1..k$, it follows that $d_0 \in A_0^{\mathcal{J}}$, which contradicts our initial assumption about d_0. Therefore, no element $A_D^{\mathcal{I}}$ has a filler for any $P \in \mathcal{P}_C$, which completes our proof that $\mathcal{I}$ is a model of $\mathcal{S}_C$. □

Proof of Proposition 5.4. Call $\Sigma = \langle \mathcal{S}, \mathcal{W} \rangle$ the 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 $s\!:\!C$ can be proved to be satisfied based on known results for (analogous) constraint system s (see e.g., [6]); the proof is by induction on the structure of C. We treat only the case $s\!:\!\forall P.D$ to clarify the restrictions on applicability of rule S2. We have to show that for all d with $(s, d) \in P^{\mathcal{I}_\Sigma}$ it holds that $d \in D^{\mathcal{I}_\Sigma}$. First we

prove that it cannot be $d = u$. In fact, the pair (s,u) is added to $P^{\mathcal{I}_\Sigma}$ only if there is no sPt in $\mathcal{W}$, but for some A, $s{:}A$ is in $\mathcal{W}$ and $A \sqsubseteq (\geqslant 1\, P)$ is in $\mathcal{S}$. Now since also $s{:}\forall P.D$ is in $\mathcal{W}$, rule S2 would be applicable, contradicting the hypothesis that Σ is complete. Therefore, $d \neq u$. Hence from the definition of $P^{\mathcal{I}_\Sigma}$ in the canonical interpretation the assertion sPd is in $\mathcal{W}$, and since S is complete, also $d{:}D$ must be in S (application of rule V3). Then by induction hypothesis we have $d \in D^{\mathcal{I}_\Sigma}$.

For 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 $s{:}A$ is in $\mathcal{W}$.

In the first case, from the definition of $\mathcal{I}_\Sigma$, one can verify 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 either of the form B or $(\leqslant 1\, P)$ then the axiom is satisfied based on the following line of reasoning: Since $s{:}A$ is in $\mathcal{W}$ and Σ is complete, based on the schema rules S4 and S5, $s{:}C$ 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 $s{:}A$ 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, $t{:}B$ is in $\mathcal{W}$ and thus $t \in B^{\mathcal{I}_\Sigma}$ by definition of $\mathcal{I}_\Sigma$.

Consider now the case that $A \sqsubseteq (\geqslant 1\, P)$ is in $\mathcal{S}$ and $s{:}A$ 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 $(\geqslant 1\, P)^{\mathcal{I}_\Sigma}$. 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 $s \in (\geqslant 1\, P)^{\mathcal{I}_\Sigma}$ 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 5.6.

(i) By induction on the application of rules. By hypothesis, the assertion $y_i{:}D$ has been added by a view rule, therefore we do not consider schema rules. For view rules, the induction is straightforward, e.g., if rule V4′ is applied because $y_i{:}\exists R.D$ is in $\mathcal{W}$ (condition 1), it adds the new assertion $y_{i+1}{:}D$. By the induction hypothesis, $depth(\exists R.D) \leqslant depth(C) - i$. For the new assertion, the claim holds since $depth(D) = depth(\exists R.D) - 1 \leqslant depth(C) - (i+1)$.

(ii) 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 $y_k{:}D$, where either $D = \forall P.E$ (rule S2′), or $D = \exists R.E$ (rule V4′), or $D = (\geqslant n\, R)$ (rule V5′). Observe that all the concepts involved are subconcepts of C, hence Proposition 5.6(i) above applies: $depth(D) \leqslant depth(C) - k = depth(C) - |C|$. However, $depth(C)$ is obviously less or equal than $|C|$ and therefore $depth(D) \leqslant 0$. Since $depth(D)$ is at least 1, a contradiction follows.

(iii) The number N is bounded by the sum of all numbers n in concepts of the form $(\geqslant n\, R)$, 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, $N \leqslant |C|$, if numbers are coded in unary notation.

(iv) 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 $(h+1)\cdot(N+1) \leqslant (|C|+1)^2$ 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 $s \neq t$ 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 $s \neq t$ 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 $|\mathcal{S}|$.

(v) A proof for a similar problem is given in [19] by showing that each rule application in Σ can be transformed into a trace rule application in a set of traces. By Proposition 5.5 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.

(vi) The claim follows from the locality of clashes: All two types of clash depend on an individual s, and on assertions 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 and from the strategy of application of trace rules). That trace contains the same clash as Σ'. □

Proof of Proposition 5.8. "⇒" 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, Theorem 5.2(ii) 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.

"⇐" By induction on the number of rule applications needed to obtain Σ' from Σ. The base case is trivial, while in the inductive case Theorem 5.2(i) proves the claim. □

Proof of Proposition 5.9. "⇒" 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 5.6 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 $\langle \mathcal{S}, \mathcal{W}'_x\rangle$.

"⇐" 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 Proposition 5.10. Let N be the maximal number of direct successors of an old individual in a precompletion: similarly to Proposition 5.6(iii), N is bounded by the sum of all numbers n in concepts of the form $(\geqslant n\, R)$, plus all concepts of the form $\exists R.C$, plus all concepts of the form $\forall P.C$, all appearing in $\mathcal{W}$. Hence, $N \leqslant |\mathcal{W}|$, 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 at most $N \cdot \omega$ new individuals; 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 $s{:}C$ 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 $O(|\mathcal{W}|^2 \cdot (|\mathcal{S}| + |\mathcal{W}|))$. □

References

[1] F. Baader, Terminological cycles in KL-ONE-based knowledge representation languages, in: Proceedings AAAI-90, Boston, MA, 1990, pp. 621–626.

[2] F. Baader, H.-J. Bürckert, B. Hollunder, W. Nutt, J.H. Siekmann, Concept logics in: J.W. Lloyd (Ed.), Computational Logics, Symposium Proceedings, Springer, Berlin, 1990, pp. 177–201.

[3] F. Baader, B. Hollunder, A terminological knowledge representation system with complete inference algorithm, in: Proceedings Workshop on Processing Declarative Knowledge (PDK-91), Lecture Notes In Artificial Intelligence, Vol. 567, Springer, Berlin, 1991, pp. 67–86.

[4] F. Baader, W. Nutt, Are complete and expressive terminological systems feasible? Position paper, in: Working Notes AAAI Fall Symposium on Issues on Description Logics: Users Meet Developers, 1992, pp. 1–5.

[5] A. Borgida, R.J. Brachman, D.L. McGuinness, L. Alperin Resnick, CLASSIC: a structural data model for objects, in: Proceedings ACM SIGMOD International Conference on Management of Data, 1989, pp. 59–67.

[6] M. Buchheit, F.M. Donini, A. Schaerf, Decidable reasoning in terminological knowledge representation systems, J. Artif. Intell. Res. 1 (1993) 109–138.

[7] M. Buchheit, A.M. Jeusfeld, W. Nutt, M. Staudt, Subsumption between queries to object-oriented databases, in: Extending Database Technology, EDBT'94 (Special Issue), Information Systems 19 (1) 1994) 33–54

[8] D. Calvanese, Reasoning with inclusion axioms in description logics: algorithms and complexity, in: W. Wahlster (Ed.), Proceedings 12th European Conference on Artificial Intelligence (ECAI-96), Budapest, Hungary, John Wiley and Sons, 1996, pp. 303–307.

[9] D. Calvanese, M. Lenzerini, On the interaction between ISA and cardinality constraints, in: Proceedings 10th IEEE International Conference on Data Engineering (ICDE-94), Houston, TX, IEEE Computer Society Press, 1994, pp. 204–213.

[10] D. Calvanese, M. Lenzerini, D. Nardi, A unified framework for class based representation formalisms, in: J. Doyle, E. Sandewall, P. Torasso (Eds.), Proceedings 4th International Conference on the Principles of Knowledge Representation and Reasoning (KR-94), Bonn, Germany, Morgan Kaufmann, Los Altos, CA, 1994, pp. 109–120.

[11] G. De Giacomo, M. Lenzerini, A uniform framework for concept definitions in description logics, J. Artif. Intell. Res. 6 (1997) 87–110.

[12] R. Dionne, E. Mays, F.J. Oles, A non-well-founded approach to terminological cycles, in: Proceedings AAAI-92, San Jose, CA, AAAI Press/MIT Press, 1992, pp. 761–766.

[13] R. Dionne, E. Mays, F.J. Oles, The equivalence of model theoretic and structural subsumption in description logics, in: Proceedings IJCAI-93, Chambery, France, Morgan Kaufmann, Los Altos, CA, 1993, pp. 710–716.

[14] F.M. Donini, B. Hollunder, M. Lenzerini, A. Marchetti, D. Nardi, W. Nutt, The complexity of existential quantification in concept languages, Artificial Intelligence 53 (1992) 309–327.

[15] F.M. Donini, M. Lenzerini, D. Nardi, W. Nutt, The complexity of concept languages, Inform. and Comput. 134 (1997) 1–58.

[16] F.M. Donini, M. Lenzerini, D. Nardi, W. Nutt, A. Schaerf, Adding epistemic operators to concept languages, in: Proceedings 3rd International Conference on the Principles of Knowledge Representation and Reasoning (KR-92),Cambridge, MA, Morgan Kaufmann, Los Altos, 1992, pp. 342–353.

[17] F.M. Donini, M. Lenzerini, D. Nardi, A. Schaerf, Deduction in concept languages: from subsumption to instance checking, J. Logic and Computation 4 (4) (1994) 423–452.

[18] B. Hollunder, Algorithmic foundations of terminological knowledge representation systems, Ph.D. Thesis, University of Saarbrücken, 1994.

[19] B. Hollunder, W. Nutt, Subsumption algorithms for concept languages, Technical Report RR-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany, 1990.

[20] B. Hollunder, W. Nutt, M. Schmidt-Schauß, Subsumption algorithms for concept description languages, in: Proceedings 9th European Conference on Artificial Intelligence (ECAI-90), Stockholm, Sweden, Pitman, London, 1990, pp. 348–353.

[21] J.E. Hopcroft, J.D. Ullman, Formal Languages and their Relation to Automata, Addison-Wesley, Reading, MA, 1969.

[22] M. Jarke, R. Gallersdörfer, M. Jeusfeld, M. Staudt, S. Eherer, Conceptbase—a deductive object manager for meta databases, J. Intelligent Information Systems 4 (2) (1995).

[23] D.S. Johnson, A catalog of complexity classes, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Vol. A, Elsevier, Amsterdam, 1990, Chapter 2.

[24] M. Lenzerini, A. Schaerf, Concept languages as query languages, in: Proceedings AAAI-91, Anaheim, CA, 1991, pp. 471–476.

[25] J.W. Lloyd. Foundations of Logic Programming, 2nd ext. ed., Springer, Berlin, 1987.

[26] R. MacGregor, Inside the LOOM description classifier, SIGART Bull. 2 (3) (1991) 88–92.

[27] R. MacGregor, What's needed to make a description logic a good KR citizen, in: Working Notes AAAI Fall Symposium on Issues on Description Logics: Users Meet Developers, 1992, pp. 53–55.

[28] B. Nebel, Reasoning and Revision in Hybrid Representation Systems, Lecture Notes in Artificial Intelligence, Springer, Berlin, 1990.

[29] B. Nebel, Terminological reasoning is inherently intractable, Artificial Intelligence 43 (1990) 235–249.

[30] B. Nebel, Terminological cycles: semantics and computational properties, in: J.F. Sowa (Ed.), Principles of Semantic Networks, Morgan Kaufmann, Los Altos, 1991, pp. 331–361.

[31] C. Peltason, The BACK system—an overview, SIGART Bull. 2 (3) (1991) 114–119.

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

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

[34] K. Schild, Personal communication, 1994.

[35] K. Schild, Terminological cycles and the propositional μ-calculus, in: J. Doyle, E. Sandewall, P. Torasso (Eds.), Proceedings 4th International Conference on the Principles of Knowledge Representation and Reasoning (KR-94), Bonn, Germany, Morgan Kaufmann, Los Altos, 1994, pp. 509–520.

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

[37] M. Vardi, The complexity of relational query languages, in: Proceedings 14th ACM SIGACT Sympsium on Theory of Computing (STOC-82), 1982, pp. 137–146.

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