Finite Model Reasoning in Description Logics

Diego Calvanese
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
calvanese@dis.uniroma1.it

Abstract

For the basic Description Logics reasoning with respect to finite models amounts to reasoning with respect to arbitrary ones, but finiteness of the domain needs to be considered if expressivity is increased and the finite model property fails. Procedures for reasoning with respect to arbitrary models in very expressive Description Logics have been developed, but these are not directly applicable in the finite case. We first show that we can nevertheless capture a restricted form of finiteness and represent finite modeling structures such as lists and trees, while still reasoning with respect to arbitrary models. The main result of this paper is a procedure to reason with respect to finite models in an expressive Description Logic equipped with inverse roles, cardinality constraints, and in which arbitrary inclusions between concepts can be specified without any restriction. This provides the necessary expressivity to go beyond most semantic and object-oriented Database models, and capture several useful extensions.

1 INTRODUCTION

From the start of the discipline, researchers working in Knowledge Representation (KR), have aimed at augmenting the expressivity of the formalisms used for representing structured knowledge, without compromising the computational properties of the procedures adopted for reasoning about it. This research has paralleled the one in Databases (DBs), where expressive data models have been developed, aimed at representing the data manipulated by commercial applications. There is a strong similarity between all these approaches, in which the domain knowledge is represented in form of a schema by defining classes, which denote subsets of the domain, and by specifying various types of relations between classes, which establish their structural properties (Calvanese, Lenzerini, & Nardi, 1994; Borgida, 1995). In this context it is fundamental to provide methods for reasoning about such specifications in order to detect inconsistencies and hidden dependencies which arise from the specified constraints. This is precisely the objective of all KR systems in the style of KL-ONE, and of Description Logics (DL) introduced for their formalization. It is gaining an increased importance also in DBs, both to support the phase of analysis and design of DB systems, and in the process of query answering to perform for example semantic query optimization.

However, while in DBs it is usually assumed that the underlying domain is finite, reasoning in DLs is performed without making such hypothesis. This seems to be in contrast with the purpose of a KR system to represent real world structures, which are inherently finite. It can however be justified by observing that most DLs studied so far have the finite model property, stating that a consistent schema (or KB) always admits a model with finite domain. This implies that one does not need to take special care about finiteness of the domain, since it can always be assumed. Unfortunately, the finite model property does not hold for more expressive logics, as those studied for example in (De Giacomo & Lenzerini, 1994a; Buongarzoni, Menghini, Salis, Sebastiani, & Straccia, 1995). In particular, this happens also for the logics which include inverse roles and functionality of roles, and which therefore have sufficient expressivity to represent semantic and object-oriented DB models (Calvanese et al., 1994). For those logics it becomes important to provide specific reasoning methods for both the finite and the unrestricted case.

While for unrestricted reasoning, techniques have been developed to deal with very expressive DLs (De Giacomo & Lenzerini, 1995), decidability of reasoning with respect to finite models is still open for important cases. In particular, there is no known method capable of handling schemata in which classes can be defined by necessary and sufficient conditions, in a logic featuring the constructs typical of DB models. Such methods are however necessary if one wants to reason for example on views in object-oriented models. Moreover, excessively limiting expressivity goes against the spirit of capturing in the schema as much as possible of the semantics of the modeled reality. Only an expressive formalism equipped with sound and complete reasoning procedures provides the user with the necessary tools to reason about relevant aspects of the domain he needs to represent. In this paper we present such formalisms in which different forms of finiteness can be captured.

On one hand, we show that while keeping the maximum expressivity, one can add to the language a *well-founded* constructor which allows one to impose well-foundedness and therefore finiteness of certain structures, without necessarily requiring the whole domain to be finite. By means of the well-founded constructor it becomes possible to define classes that represent modeling structures such as lists and trees, and reason about them like about any other class in the schema. This represents a significant improvement with respect to traditional data models, where such modeling structures, if present at all, are ad hoc additions that require a special treatment by the reasoning procedures (Cattell, 1994). The known methods for reasoning with respect to arbitrary models in expressive DLs are based on a correspondence between DLs and Propositional Dynamic Logics (PDLs) and exploit the powerful reasoning methods developed for PDLs (Schild, 1991; De Giacomo & Lenzerini, 1994a). We show that the correspondence can in fact be extended to handle also well-foundedness, and present a reasoning technique for the resulting PDL.

On the other hand, we develop a method to reason with respect to finite models on schemata built using a very expressive DL, and show its decidability in deterministic double exponential time. The DL we consider includes besides number restrictions and inverse roles also general negation of concept expressions. The schemata are of the most general form, in which one can express arbitrary inclusions between complex expressions without any restriction at all. This makes our language not only capable of capturing the structural properties of class-based representation formalisms used in DBs, but provides also the necessary expressivity to subsume the terminological component of most existing concept based systems.

The rest of the paper is organized as follows: In Section 2 we present the representation formalism we adopt, in Section 3 we discuss the issues related to reasoning about finite modeling structures and in finite models and in Sections 4 and 5 we analyze the reasoning methods used in both contexts. In Section 6 we outline possible directions for future work.

2 THE REPRESENTATION FORMALISM

In this section, we present the family of class-based formalisms we deal with. These formalisms exploit Description Logics for the definition of schemata, and the logics we define extend the well known logics of the $\mathcal{AL}$-family (Donini, Lenzerini, Nardi, & Nutt, 1991a) by several expressive constructs.

2.1 SYNTAX AND SEMANTICS OF DESCRIPTION LOGICS

The basic elements of *Description Logics* (*DLs*) are *concepts* and *roles*, which denote classes and binary relations, respectively. Complex concept and role expressions are built, starting from a set of *concept* and *role names*, by applying certain constructors. Each DL is characterized by the set of allowed constructors. Table 1 shows the concept and role forming constructors we consider in this paper. The first column specifies the name of each constructor and the second column specifies a letter that is used to denote each constructor when naming a logic. The basic logic we consider is $\mathcal{AL}$, which contains only the constructors to which no letter is associated. All other logics we deal with include the constructors of $\mathcal{AL}$. Each logic is named with a string that starts with $\mathcal{AL}$ and includes the letters that are associated to the additional constructors in the logic.

The syntax of each constructor in concept and role expressions is shown in the third column of the table. Concept names are denoted by C and complex concept expressions by E. Role names and complex role expressions are denoted by P and R, respectively. m denotes a positive and n a nonnegative integer.

Concepts are interpreted as subsets of a domain and roles as binary relations over that domain. More precisely, an *interpretation* $\mathcal{I} := (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ (the *domain* of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ (the *interpretation function* of $\mathcal{I}$) that maps every concept name C to a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, every role name P to a sub-

Constructor Name		Syntax	Semantics
concept name		C	$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
top		$\top$	$\Delta^{\mathcal{I}}$
atomic negation		$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction		$E_1 \sqcap E_2$	$E_1^{\mathcal{I}} \cap E_2^{\mathcal{I}}$
universal quantification		$\forall R.E$	$\{o \mid \forall o' \colon (o,o') \in R^{\mathcal{I}} \rightarrow o' \in E^{\mathcal{I}}\}$
unqualified existential quantification		$\exists R$	$\{o \mid \exists o' \colon (o,o') \in R^{\mathcal{I}}\}$
existential quantification	$\mathcal{E}$	$\exists R.E$	$\{o \mid \exists o' \colon (o,o') \in R^{\mathcal{I}} \wedge o' \in E^{\mathcal{I}}\}$
disjunction	$\mathcal{U}$	$E_1 \sqcup E_2$	$E_1^{\mathcal{I}} \cup E_2^{\mathcal{I}}$
general negation	$\mathcal{C}$	$\neg E$	$\Delta^{\mathcal{I}} \setminus E^{\mathcal{I}}$
number restrictions	$\mathcal{N}$	$\exists^{\geq m} R$	$\{o \mid \sharp\{o' \mid (o,o') \in R^{\mathcal{I}}\} \geq m\}$
		$\exists^{\leq n} R$	$\{o \mid \sharp\{o' \mid (o,o') \in R^{\mathcal{I}}\} \leq n\}$
qualified number restrictions	$\mathcal{Q}$	$\exists^{\geq m} R.E$	$\{o \mid \sharp\{o' \mid (o,o') \in R^{\mathcal{I}} \wedge o' \in E^{\mathcal{I}}\} \geq m\}$
		$\exists^{\leq n} R.E$	$\{o \mid \sharp\{o' \mid (o,o') \in R^{\mathcal{I}} \wedge o' \in E^{\mathcal{I}}\} \leq n\}$
well-founded	$\mathcal{W}$	$wf(R)$	$\{o_0 \mid \forall o_1, o_2, \ldots$ (ad infinitum) $\exists i \geq 0 \colon (o_i, o_{i+1}) \notin R^{\mathcal{I}}\}$
role value map	$\mathcal{V}$	$(R_1 \subseteq R_2)$	$\{o \mid \{o' \mid (o,o') \in R_1^{\mathcal{I}}\} \subseteq \{o' \mid (o,o') \in R_2^{\mathcal{I}}\}\}$
role name		P	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
inverse	$\mathcal{I}$	R^-	$\{(o,o') \mid (o',o) \in R^{\mathcal{I}}\}$
union	$\mathcal{R}$	$R_1 \cup R_2$	$R_1^{\mathcal{I}} \cup R_2^{\mathcal{I}}$
concatenation	$\mathcal{R}$	$R_1 \circ R_2$	$R_1^{\mathcal{I}} \circ R_2^{\mathcal{I}}$
reflexive transitive closure	$\mathcal{R}$	R^*	$(R^{\mathcal{I}})^*$
identity	$\mathcal{R}$	$id(E)$	$\{(o,o) \mid o \in E^{\mathcal{I}}\}$
difference	$\mathcal{D}$	$R_1 \setminus R_2$	$R_1^{\mathcal{I}} \setminus R_2^{\mathcal{I}}$

Table 1: Syntax and semantics of the concept and role forming constructors.

set $P^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and concept and role expressions according to the last column of Table 1[1].

Most of the constructors we have presented are well known in DLs, and have a correspondence in modeling constructs used both in Frame Based Systems and in DB models. This is not the case for the constructor wf, called *well-founded*, which has rarely been considered in KR, but which proves particularly useful for expressing modeling structures that are inherently finite.

2.2 $\mathcal{AL}$-SCHEMATA

Using concept expressions of a DL, intensional knowledge can be specified through schemata. Given an $\mathcal{AL}$-DL $\mathcal{L}$, an *$\mathcal{L}$-schema* is a triple $\mathcal{S} := (\mathcal{C}, \mathcal{P}, \mathcal{T})$, where $\mathcal{C}$ is a finite set of concept names, $\mathcal{P}$ is a finite set of role names, and $\mathcal{T}$ is a finite set of *assertions*. Each such assertion has one of the forms

$C \doteq E$ *(primitive concept specification)*
$C \doteq E$ *(concept definition)*

where $C \in \mathcal{C}$ and E is a concept expression of $\mathcal{L}$ involving only names of $\mathcal{C} \cup \mathcal{P}$. In the following, when not specified otherwise, we assume that $\mathcal{S} := (\mathcal{C}, \mathcal{P}, \mathcal{T})$, and that $\mathcal{C}$ and $\mathcal{P}$ are the sets of concept and role names that appear in $\mathcal{T}$.

Primitive concept specifications are used to specify necessary conditions for an object to be an instance of a concept, while concept definitions specify both necessary and sufficient conditions. More formally, an interpretation $\mathcal{I}$ *satisfies* an assertion $C \doteq E$ if $C^{\mathcal{I}} \subseteq E^{\mathcal{I}}$, and it satisfies an assertion $C \doteq E$ if $C^{\mathcal{I}} = E^{\mathcal{I}}$. An interpretation that satisfies all assertions in a schema $\mathcal{S}$ is called a *model* of $\mathcal{S}$. A *finite model* is a model with finite domain. A concept expression E is *(finitely) consistent* in $\mathcal{S}$ if $\mathcal{S}$ admits a (finite) model $\mathcal{I}$ such that $E^{\mathcal{I}} \neq \emptyset$, and E_1 is *(finitely) subsumed* by E_2 in $\mathcal{S}$ if $E_1^{\mathcal{I}} \subseteq E_2^{\mathcal{I}}$ for every (finite) model $\mathcal{I}$ of $\mathcal{S}$.

Both in KR and in DB models several different assumptions on the form of a schema are made, either implicitly or explicitly:

1. For each concept name there is at most one assertion in which that name appears in the left-hand side.
2. The schema contains no cycles[2].

[1] $\sharp S$ denotes the cardinality of a set S.

[2] See (Nebel, 1991) for a formal definition of cycle in a schema.

3. The schema contains only primitive concept specifications (*primitive schema*).

Assumption 1 corresponds to the natural requirement that all the information concerning a class is contained in one place, which contributes to a better structuring of the represented knowledge (Nebel, 1990; Baader, 1990; Lecluse & Richard, 1989; Bergamaschi & Sartori, 1992). It may however be too restrictive, especially in those cases where one wants to state additional constraints that are not specific to a certain class.

Assumption 2 of acyclicity (which is meaningful only in the presence of assumption 1) is made in most existing concept-based KR systems, although imposing this condition strongly limits the expressive power of the system (Nebel, 1990). Many real world concepts can in fact be expressed naturally only in a recursive way and thus through cycles. The reason for not admitting cycles is twofold. On one hand cycles greatly increase the computational complexity of reasoning on a schema (Baader, 1990; Calvanese, 1996a), and on the other hand, there is still no agreement in the field on which semantics to adopt in their presence. Besides *descriptive semantics*, which is the semantic specification described above, so called *fixpoint semantics* have been considered (Nebel, 1991). Descriptive semantics, which is the one we use, has been advocated to be the one which gives the most intuitive results (Nebel, 1991; Buchheit, Donini, Nutt, & Schaerf, 1995), and it is also the only one that can be adopted for the most general type of schemata, called *free*, in which none of the three conditions above is required to hold[3]. (Baader, 1990) argues that from a constructive point of view greatest fixpoint semantics should be preferred. Least fixpoint semantics, on the other hand, seems the most appropriate for the definition of inductive structures. We will see in Section 3 that we can obtain a similar result by using the well-founded constructor together with descriptive semantics. We remark that recently there have been also proposals for an integration of the different types of semantics by including fixpoint constructors in the logic (Schild, 1994; De Giacomo & Lenzerini, 1994b) and interpreting cycles with descriptive semantics.

Assumption 3 is sometimes made in concept-based KR systems (Buchheit et al., 1995) and it is usually implicit in DB models, in which the classes are specified only through necessary conditions (Calvanese et al., 1994). Although it strongly limits expressivity, reasoning on primitive schemata is nevertheless intractable even in the simplest setting, where we use only the constructors of $\mathcal{AL}$ (which are common to almost all representation formalisms) and the schema is acyclic (Buchheit et al., 1995). In fact assumption 3 reduces worst case complexity of reasoning on a schema only if relatively weak languages are adopted (Calvanese, 1996a).

[3]The expressivity of free schemata defined in this way is equivalent to the one of schemata constituted by *free inclusion assertions*, which have the form $E_1 \doteq E_2$, where both E_1 and E_2 are arbitrary concept expressions with no restriction at all.

3 FROM FINITE MODELING STRUCTURES TO FINITE MODELS

Both in KR and in DBs representation formalisms have been developed which offer powerful structuring capabilities and procedures to reason about a specification. However, while in DBs the common assumption is that the modeled domain is finite, this has seldom been an issue in KR in general and in DLs in particular, where the developed reasoning methods do not take care of finiteness. For most of the DLs that have been considered till now this does not represent a limitation, neither for reasoning on concept expressions (Donini et al., 1991a; Donini, Lenzerini, Nardi, & Nutt, 1991b) nor on schemata (Nebel, 1991; Buchheit et al., 1995), since for these formalisms the *finite model property* holds. It states that if a schema (or concept expression) admits a model, then it also admits one with a finite domain, and therefore reasoning with respect to unrestricted models amounts to reasoning with respect to finite ones. This does not hold anymore for the more expressive logics studied recently, which include inverse roles and functionality of roles, since these constructors in combination with cycles in the schema cause the finite model property to fail to hold.

Example 1 Consider the following schema:

$$\begin{aligned} \texttt{Guard} &\doteq \exists \texttt{shields.Guard} \sqcap \exists^{\leq 1} \texttt{shields}^- \\ \texttt{FirstGuard} &\doteq \texttt{Guard} \sqcap \exists^{\leq 0} \texttt{shields}^- \end{aligned}$$

Each `Guard` shields at least some guard, and is shielded by at most one individual. A `FirstGuard` is a guard whom nobody shields. It is easy to see that the existence of a `FirstGuard`forces the existence of an infinite sequence of guards, each one shielding the next. Therefore `FirstGuard` is consistent but not finitely consistent. ∎

For other examples, see the logics in (De Giacomo & Lenzerini, 1994a; Calvanese et al., 1994), or the one in (Buongarzoni et al., 1995), where not cycles but

singleton concepts are used to specify concepts that are consistent only in an infinite domain.

For the most expressive DLs (and in particular for those that lack the finite model property), methods for reasoning with respect to arbitrary models have been developed, which are based on a correspondence between DLs and Propositional Dynamic Logics (PDLs). PDLs are formalisms specifically designed for reasoning about program schemes (Kozen & Tiuryn, 1990), and their correspondence with DLs was described first in (Schild, 1991) and extended to more expressive logics in (De Giacomo & Lenzerini, 1994a). The correspondence allows one to reduce concept consistency to satisfiability of a formula of some PDL, and to use the advanced methods developed for PDLs to solve this latter task. These methods exploit the fundamental *tree model property*, shared by all PDLs, which states that every satisfiable formula admits particular models that have the form of an (infinite) tree of bounded degree. If the domain has to be finite, however, the existence of tree-like models is not guaranteed, and the known reasoning methods are not applicable.

It is nevertheless possible to express restricted forms of finiteness, even in those logics that lack the finite model property, by imposing conditions that force certain structures in the domain to be well-founded, without assuming the whole interpretation domain to be finite. This allows us to deal with finite modeling structures, while exploiting the powerful techniques for reasoning with respect to unrestricted models. Well-foundedness can be imposed by using $wf(R)$, which expresses that there is no infinite sequence of objects, connected one to the next by means of role R. In terms of PDLs, $wf(R)$ corresponds to a negated repeat formula $\neg\Delta(R)$ (Streett, 1982), which is in turn equivalent to the least fixpoint expression[4] $\mu X.\forall R.X$ (Kozen, 1983). Thus we can use such constructor to express finite structures that admit a least-fixpoint definition. We illustrate this on the example of binary trees.

Example 2 We can define a binary tree inductively as the least set *bin-tree* such that:

- a node having no successors is a *bin-tree*,
- a node whose successors are all *bin-trees*, is a *bin-tree*,

and where additionally a *node* is constrained to have at most one predecessor and at most two successors (plus possibly additional properties such as a label).

The inductive part of the definition can be captured by the least fixpoint expression

$$\mu X.((\texttt{Node} \sqcap \exists^{\leq 0}\texttt{edge}) \sqcup (\texttt{Node} \sqcap \forall \texttt{edge}.X))$$

which is equivalent to

$$\mu X.(\texttt{Node} \sqcap \forall \texttt{edge}.X).$$

Thus, the following (non-recursive) schema reflects the full definition of binary tree, where the first assertion in the schema captures the local properties of nodes:

$$\begin{aligned} \texttt{Node} &\doteq \exists^{\leq 2}\texttt{edge} \sqcap \exists^{\leq 1}\texttt{edge}^{-} \sqcap \exists \texttt{label}.\ldots \\ \texttt{BinTree} &\doteq \mu X.(\texttt{Node} \sqcap \forall \texttt{edge}.X) \end{aligned} \quad (1)$$

It is possible to show that, for every model $\mathcal{I}$ of a schema $\mathcal{S}$ containing the assertions above, the set of objects that are instances of `BinTree` remains the same if we replace assertion 1 by the following:

$$\texttt{BinTree} \doteq \texttt{Node} \sqcap \forall \texttt{edge}.\texttt{BinTree} \sqcap wf(\texttt{edge}) \quad (2)$$

Notice that this assertion is recursive and makes use of the well-founded constructor to achieve the same effect as the least-fixpoint expression. In particular, given an interpretation $\mathcal{I}$ of $\mathcal{S}$ that interprets all concept and role names in $\mathcal{S}$ except `BinTree`, there is a unique way to extend $\mathcal{I}$ to `BinTree` such that it satisfies assertion 2. The extension of `BinTree` is exactly the set of objects that are instances of the least-fixpoint expression on the right hand side of assertion 1. Therefore assertion 2, despite being recursive, represents a proper definition. ∎

By proceeding in a similar way, most data structures encountered in Computer Science which admit an inductive definition, such as lists, trees, and directed acyclic graphs, can be modeled in a DL including the well-founded constructor. Notice that in general to this end both inverse roles and number restrictions are indispensable. For this reason we can not directly make use of fixpoint constructors as in the DLs presented in (Schild, 1994; De Giacomo & Lenzerini, 1994b), since decidability is still open for those logics including fixpoints in combination with inverse roles.

Another important use of the well-founded constructor is for the definition of well-founded relations, of which the *part-of* relation is a notable example with great practical relevance (Artale, Franconi, Guarino, & Pazzi, 1996; Sattler, 1995). Transitivity of the part-of

[4] A least fixpoint expression $\mu X.f(X)$, where X is a concept variable and f is a monotone operator, is interpreted as the least set $X^{\mathcal{I}}$ of objects in $\Delta^{\mathcal{I}}$ satisfying $X^{\mathcal{I}} = (f(X))^{\mathcal{I}}$. See (Schild, 1994; De Giacomo & Lenzerini, 1994b) for more details.

relation can be captured by taking the transitive closure of a `basic_part_of` role, while asymmetricity and finiteness (i.e. well-foundedness) are imposed by an assertion of the form $\top \dot{\preceq} wf(\texttt{basic\_part\_of})$. Notice also that by means of role value maps on role names we can capture different specializations of the part-of relation.

The constructors that cause the loss of the finite model property are all necessary in order to capture the characteristics of the most important representation formalisms used both in KR and in DBs (Calvanese et al., 1994). In fact, neither the Entity-Relationship model nor expressive object-oriented models have the finite model property, and since the assumption of a finite domain is essential in this context, specialized methods capable of reasoning in these formalisms with respect to finite models are needed. This seems to be more difficult than reasoning with respect to arbitrary ones. As already noticed, if the domain has to be finite the tree model property does not hold, and the correspondence with PDLs cannot be exploited. In fact, decidability of finite concept consistency and finite subsumption for more expressive formalisms is still an open problem, and there are no known methods to handle concept definitions or even (qualified) existential quantification in conjunction with the constructors that cause the finite model property not to hold. The main result of this paper, presented in Section 5, is a method to reason with respect to finite models on free $\mathcal{ALCQI}$-schemata. The possibility in such schemata to make use of full negation and of concept definitions provides indeed the necessary expressivity to go beyond the basic features of DB models and capture and reason upon several useful constructs that have been proposed as extensions to existing models. Relevant examples are:

- Classes defined through necessary and sufficient conditions, which correspond to so called *views* in object-oriented DBs (Abiteboul & Bonner, 1991). Using free assertions[5] it is also possible to specify only sufficient conditions for an object to be an instance of a class, and more general forms of integrity constraints.
- Classes defined as the union of other classes.
- ISA relations between relationships in the Entity-Relationship model (Di Battista & Lenzerini, 1993).
- Negative assertions about classes, such as disjointness, or non-participation in a relationship (Di Battista & Lenzerini, 1993).

[5] See footnote 3.

`Athlete`	$\dot{\preceq}$	$\forall$`partic.Comp` $\sqcap$ $\exists^{\leq 4}$`partic` $\sqcap$ $\forall$`wins.RunningComp` $\sqcap$ $\exists$`wins.Final`
`Comp`	$\dot{\preceq}$	$\forall$`partic`$^-$`.Athlete`
`Final`	$\dot{\preceq}$	`Comp` $\sqcap$ $\exists^{\leq 1}$`wins`$^-$
`RunningComp`	$\dot{\preceq}$	`Comp`
`Final` $\sqcap$ `RunningComp`	$\dot{\preceq}$	$\exists^{\geq 6}$`partic`$^-$

Figure 1: A schema with finitely inconsistent concepts

Example 3 The schema $\mathcal{S}$ shown in Figure 1 models the participation of athletes in sports competitions. The last (free) inclusion assertion represents a constraint that is not associated to a specific concept name. It turns out, however, that $\mathcal{S}$ contains a modeling error that causes `Athlete` to be finitely inconsistent. In fact, the conditions imposed on `Athlete` should only be required for (fast) runners that win a final. Therefore it is not correct to assume that only such athletes can participate in a competition. Notice that the inconsistency above would not be detected by a reasoning procedure that ignores finiteness, since all concepts in $\mathcal{S}$ are consistent. ∎

Before discussing in Section 5 the method to reason with respect to finite models, we briefly illustrate in Section 4 how to reason with respect to arbitrary models in a very expressive DL which includes the well-foundedness constructor, and therefore allows us to represent finite modeling structures.

4 REASONING ABOUT FINITE MODELING STRUCTURES

We now sketch a method to reason with respect to arbitrary models on free schemata expressed in a DL that includes all constructors shown in Table 1, and in particular the well-founded constructor that allows us to define finite modeling structures. Some of the other constructors, however, and in particular role-value maps, intersections of complex roles, and number restrictions on complex roles, make reasoning highly undecidable if they are used without restrictions (Schmidt-Schauß, 1989; Harel, 1983). Therefore, we syntactically restrict the use of these problematic constructors in a way similar to what done in (De Giacomo & Lenzerini, 1995) for the logic $\mathcal{CAT}$. The logic $\mathcal{ALT}$, in which we distinguish between *basic roles*, denoted by the letter B, and arbitrary complex roles, is defined as follows:

$$E \longrightarrow C \mid E_1 \sqcap E_2 \mid \neg E \mid \forall R.E \mid \exists^{\leq n} B.E \mid \exists^{\leq n} B^-.E \mid (B_1 \subseteq B_2) \mid (B_1^- \subseteq B_2^-) \mid wf(R)$$

$$B \longrightarrow P \mid B_1 \cup B_2 \mid B_1 \setminus B_2$$
$$R \longrightarrow B \mid R_1 \cup R_2 \mid R_1 \circ R_2 \mid R^- \mid R^* \mid id(E).$$

The semantics for $\mathcal{ALT}$ class and role expressions is defined as in Table 1, considering that basic roles are just particular types of complex roles.

Decidability in deterministic exponential time of reasoning in an object-oriented model with the same expressivity as free $\mathcal{ALT}$-schemata was already shown in (Calvanese, De Giacomo, & Lenzerini, 1995). We briefly sketch the idea underlying the proof, which exploits the correspondence with PDLs: Concepts of DLs correspond to formulae of PDLs, roles correspond to programs, and (almost) every constructor of DLs has its counterpart in PDLs.

We can reduce the problem of reasoning on a free $\mathcal{ALT}$-schema to the problem of verifying the satisfiability of a formula of RFCPDL, which is basic PDL augmented with the converse operator on programs (corresponding to inverse roles), local functionality on both direct and inverse atomic programs (which corresponds to functional restrictions, to which existential quantifications are reduced by "reifying" basic roles) and the repeat constructor (which corresponds to well-foundedness). The details of the reduction can be found in (Calvanese, 1996b).

Proposition 4 *Given a free $\mathcal{ALT}$-schema $\mathcal{S}$ and a concept expression E, one can construct in polynomial time in $|\mathcal{S}| + |E|$ an* RFCPDL*-formula $f_{\mathcal{S},E}$ such that $f_{\mathcal{S},E}$ is satisfiable if and only if E is consistent in $\mathcal{S}$.*

In order to prove the desired decidability result, however, we cannot exploit known decision procedures for PDLs, since the decidability of RFCPDL is open. To show decidability of RFCPDL we reduce satisfiability of a formula to nonemptiness of the language accepted by a finite automaton on infinite objects derived from the formula (Thomas, 1990). This is a standard technique that provides tight upper bounds for various modal and temporal logics (Emerson & Jutla, 1989; Vardi & Wolper, 1994) and logics of programs (Vardi, 1985; Vardi & Wolper, 1986; Vardi & Stockmeyer, 1985), and we extend it to RFCPDL. It is based on the tree model property which holds also for RFCPDL. In particular, the automaton we obtain from a RFCPDL formula f is a hybrid automaton $H_f := (T_f, W_f)$, that combines an automaton on infinite trees T_f of exponential size in $|f|$, with an automaton on infinite words W_f of polynomial size in $|f|$ and which handles the repeat sub-formulae. Exploiting the results described in (Emerson & Jutla, 1989), the nonemptiness problem for such a hybrid tree automaton can be solved in deterministic time that is polynomial in the number of states of T_f and exponential in the number of states of W_f.

This allows us to establish the desired upper bound for satisfiability in RFCPDL, and therefore for unrestricted model reasoning in $\mathcal{ALT}$. By the well known **EXPTIME** lower bound for reasoning on free $\mathcal{FL}^-$-schemata[6] the complexity bound is tight.

Theorem 5 *Unrestricted concept consistency and concept subsumption in free $\mathcal{ALT}$-schemata are **EXPTIME**-complete.*

5 REASONING WITH RESPECT TO FINITE MODELS

In this section we present a technique for reasoning with respect to finite models in $\mathcal{L}$-schemata. The method extends the one proposed in (Calvanese et al., 1994) to reason on primitive $\mathcal{ALUNI}$-schemata, and is based on the same idea of constructing from a schema a system of linear inequations, and relating the existence of finite models of the schema to the existence of particular solutions of the system. The method we present decides concept consistency (and therefore also subsumption) in free $\mathcal{ALCQI}$-schemata. Due to the presence of arbitrary free inclusion assertions, and the higher expressivity of the underlying language (in particular the presence of existential quantification) the construction of the system of inequations is much more involved than for primitive $\mathcal{ALUNI}$-schemata.

We describe the method for reasoning on free $\mathcal{ALCNI}$-schemata and sketch then how it must be extended to deal with qualified number restrictions. We use the term *literal*, ranged over by L, to denote either a concept name or a concept name preceded by the symbol "$\neg$". It is easy to see that each free schema can be transformed in linear time into an equivalent *normalized schema*, where each assertion has the form $L \doteq E$, and E has one of the forms

$$L_1 \mid L_1 \sqcup L_2 \mid \forall R.L_1 \mid \exists R.L_1 \mid \exists^{\geq m} R \mid \exists^{\leq n} R.$$

Therefore, in the following we assume to deal with free normalized $\mathcal{ALCNI}$-schemata.

5.1 EXPANSION OF A SCHEMA

We generalize now the definition of expansion of a schema introduced in (Calvanese et al., 1994) to free $\mathcal{ALCNI}$-schemata. Let $\mathcal{S} := (\mathcal{C}, \mathcal{P}, \mathcal{T})$ be such a schema. We call a set $\widehat{D} \in 2^{\mathcal{C}}$ a *compound concept*,

[6] $\mathcal{FL}^-$ is the language obtained from $\mathcal{AL}$ by dropping atomic negation.

and an element $\widehat{Q} \in \mathcal{P} \times 2^{\mathcal{C}} \times 2^{\mathcal{C}}$ a *compound role*. $\widehat{D}$ ranges over compound concepts and $\widehat{Q}$ over compound roles. A *compound assertion* has one of the forms

$$\widehat{D} \doteq \exists^{\geq m} R, \qquad \widehat{D} \doteq \exists^{\leq n} R, \qquad \widehat{D} \doteq \exists R.L.$$

For a negative literal $L = \neg C$, when we write $L \in \widehat{D}$ we mean $C \notin \widehat{D}$.

Intuitively, the instances of a compound concept $\widehat{D}$ are all those objects of the domain that are instances of all concepts in $\widehat{D}$ and are not instances of any concept not in $\widehat{D}$. A compound role $(P, \widehat{D}_1, \widehat{D}_2)$ is interpreted as the restriction of role P to the pairs whose first component is an instance of $\widehat{D}_1$ and whose second component is an instance of $\widehat{D}_2$. More formally, the semantics of compound concepts and roles is given by extending the interpretation function as follows:

$$\begin{aligned} \widehat{D}^{\mathcal{I}} &:= \bigcap_{C \in \widehat{D}} C^{\mathcal{I}} \setminus \bigcup_{C \in (\mathcal{C} \setminus \widehat{D})} C^{\mathcal{I}} \\ (P, \widehat{D}_1, \widehat{D}_2)^{\mathcal{I}} &:= P^{\mathcal{I}} \cap (\widehat{D}_1^{\mathcal{I}} \times \widehat{D}_2^{\mathcal{I}}). \end{aligned}$$

Note that according to this definition two different compound concepts have necessarily disjoint interpretations.

We say that a compound concept $\widehat{D} \in 2^{\mathcal{C}}$ is *$\mathcal{S}$-consistent*, if for every $L \in \widehat{D}$,

- if $(L \doteq L') \in \mathcal{T}$, then $L' \in \widehat{D}$, and
- if $(L \doteq L_1 \sqcup L_2) \in \mathcal{T}$, then $L_1 \in \widehat{D}$ or $L_2 \in \widehat{D}$.

We say that a compound role $(P, \widehat{D}_1, \widehat{D}_2) \in \mathcal{P} \times 2^{\mathcal{C}} \times 2^{\mathcal{C}}$ is *$\mathcal{S}$-consistent*, if

- $\widehat{D}_1$ and $\widehat{D}_2$ are $\mathcal{S}$-consistent,
- for every $L \in \widehat{D}_1$, if $(L \doteq \forall P.L') \in \mathcal{T}$, then $L' \in \widehat{D}_2$, and
- for every $L \in \widehat{D}_2$, if $(L \doteq \forall P^{-1}.L') \in \mathcal{T}$, then $L' \in \widehat{D}_1$.

Intuitively, compound concepts and roles that are not consistent necessarily have an empty extension in all models of the schema.

The *expansion* $\widehat{\mathcal{S}} := (\mathcal{C}, \mathcal{P}, \widehat{\mathcal{D}}, \widehat{\mathcal{Q}}, \widehat{\mathcal{T}})$ of $\mathcal{S}$ is defined as follows:

- $\widehat{\mathcal{D}}$ is the set of all $\mathcal{S}$-consistent compound concepts.
- $\widehat{\mathcal{Q}}$ is the set of all $\mathcal{S}$-consistent compound roles.
- $\widehat{\mathcal{T}}$ is the smallest set of compound assertions such that for every $\widehat{D} \in \widehat{\mathcal{D}}$:
  - if for some $L \in \widehat{D}$ there is an assertion $(L \doteq \exists^{\geq m} R) \in \mathcal{T}$, then $\widehat{\mathcal{T}}$ contains the compound assertion $\widehat{D} \doteq \exists^{\geq m_{max}} R$, where $m_{max} := \max\{m \mid \exists L \in \widehat{D} \colon (L \doteq \exists^{\geq m} R) \in \mathcal{T}\}$;
  - if for some $L \in \widehat{D}$ there is an assertion $(L \doteq \exists^{\leq n} R) \in \mathcal{T}$, then $\widehat{\mathcal{T}}$ contains the compound assertion $\widehat{D} \doteq \exists^{\leq n_{min}} R$, where $n_{min} := \min\{n \mid \exists L \in \widehat{D} \colon (L \doteq \exists^{\leq n} R) \in \mathcal{T}\}$.
  - if for some $L \in \widehat{D}$ there is an assertion $(L \doteq \exists R.L') \in \mathcal{T}$, then $\widehat{\mathcal{T}}$ contains the compound assertion $\widehat{D} \doteq \exists R.L'$.

It is easy to see that the size of the expansion is exponential in the size of the original schema and that it can also be effectively constructed in exponential time.

In (Calvanese et al., 1994) it is shown that for primitive $\mathcal{ALUNI}$-schemata a system of linear inequations can be directly constructed from the compound assertions in the expansion, such that particular solutions of this system correspond to finite models of the schema. Unfortunately, for free $\mathcal{ALCNI}$-schemata this approach does not work directly, and this is due to the presence of assertions of the form $L \doteq \exists R.L'$. One could think to handle such assertions by constructing the expansion as specified above, simply coding existential quantifications inside compound assertions and leaving their treatment to the system of inequations (as done for number restrictions).

The intuitive reason why this simple approach does not lead to the desired results, is that it relies on the uniformity of all objects that are instances of the same compound concept. When setting up the system of inequations we are in fact transforming local constraints on the number of connections that a single object may have into global constraints on the total number of connections of a certain type. The necessary differentiation is introduced by constructing the expansion. Once this is done all instances of the same compound concept can be regarded as equivalent. The approach works for the case of $\mathcal{ALUNI}$-schemata, where the concept expressions cannot distinguish between different instances of the same compound concept. If we use existential quantification, however, due to the increased expressivity it is not sufficient anymore to split the schema into compound concepts to obtain a uniform behaviour of the instances. This leads us to introduce the notion of biexpansion of a schema, where we make a more fine-grained separation based also on the existence of connections of certain types.

5.2 BIEXPANSION OF A SCHEMA

In order to define the biexpansion of a schema we need some additional terminology. Let $\mathcal{P}^- := \{P^- \mid P \in \mathcal{P}\}$ and $\mathcal{R} := \mathcal{P} \cup \mathcal{P}^-$. We use $\mathcal{B}_{\mathcal{C},\mathcal{P}}$ to denote $2^{\mathcal{C}} \times (2^{2^{\mathcal{P}}})^{\mathcal{R}}$. An element of $\mathcal{B}_{\mathcal{C},\mathcal{P}}$ is called a *bicompound concept*, and it is constituted by a pair $(\widehat{D}, \varphi)$ in

which the first element is a compound concept, and the second element is a function $\varphi : \mathcal{R} \to 2^{2^{\mathcal{C}}}$ that associates to each role and inverse role a set of compound concepts. We introduce two functions that allow us to refer to the components of bicompound concepts:

- The function $\mathrm{cc} : \mathcal{B}_{\mathcal{C},\mathcal{P}} \to 2^{\mathcal{C}}$ returns the first component $\widehat{D}$ of a bicompound concept $(\widehat{D}, \varphi)$.
- The function $\mathrm{ccs} : \mathcal{B}_{\mathcal{C},\mathcal{P}} \times \mathcal{R} \to 2^{2^{\mathcal{C}}}$ returns for a bicompound concept $\widetilde{B} := (\widehat{D}, \varphi)$ and a role R the set of compound concepts assigned to R by the second component φ of $\widetilde{B}$.

An element of $\mathcal{P} \times \mathcal{B}_{\mathcal{C},\mathcal{P}} \times \mathcal{B}_{\mathcal{C},\mathcal{P}}$ is called a *bicompound role*. $\widetilde{B}$ and $\widetilde{U}$ range over bicompound concepts and bicompound roles respectively.

The intuition behind the definition of bicompound concepts and roles is the following: If $\mathrm{cc}(\widetilde{B}) = \widehat{D}$, then all instances of $\widetilde{B}$ are instances of $\widehat{D}$. Let o be such an instance. Then for each compound concept $\widehat{D}'$ in $\mathrm{ccs}(\widetilde{B}, R)$, there is an instance o' of $\widehat{D}'$ connected to o via role R, while for the compound concepts not in $\mathrm{ccs}(\widetilde{B}, R)$ there is no such instance. In analogy to compound roles, a bicompound role $(P, \widetilde{B}_1, \widetilde{B}_2)$ is interpreted as the restriction of role P to the pairs whose first component is an instance of $\widetilde{B}_1$ and whose second component is an instance of $\widetilde{B}_2$.

A *bicompound assertion* has one of the forms

$$\widetilde{B} \doteq \exists^{\geq m} R, \qquad \widetilde{B} \doteq \exists^{\leq n} R.$$

The semantics of bicompound concepts and roles is again defined by extending the interpretation function, where for simplicity of notation we make use of concept expressions built by applying the $\mathcal{ALCNI}$-constructors to both concepts and compound concepts:

$$\widetilde{B}^{\mathcal{I}} := (\mathrm{cc}(\widetilde{B}))^{\mathcal{I}} \cap \bigcap_{R \in \mathcal{R}} \left(\bigcap_{\widehat{D} \in \mathrm{ccs}(\widetilde{B},R)} (\exists R.\widehat{D})^{\mathcal{I}} \cap \bigcap_{\widehat{D} \in 2^{\mathcal{C}} \setminus \mathrm{ccs}(\widetilde{B},R)} (\neg \exists R.\widehat{D})^{\mathcal{I}} \right)$$

$$(P, \widetilde{B}_1, \widetilde{B}_2)^{\mathcal{I}} := P^{\mathcal{I}} \cap (\widetilde{B}_1^{\mathcal{I}} \times \widetilde{B}_2^{\mathcal{I}})$$

This definition, together with the fact that different compound concepts have disjoint extensions, implies that two different bicompound concepts also have disjoint extensions. The same observation holds for two different bicompound roles $(P, \widetilde{B}_1, \widetilde{B}_2)$ and $(P, \widetilde{B}_1', \widetilde{B}_2')$ that refer to the same role P. Also, given the extensions of all bicompound concepts and bicompound roles, it is immediate to obtain the extensions of concepts and roles.

We say that a bicompound concept $\widetilde{B} \in \mathcal{B}_{\mathcal{C},\mathcal{P}}$ is $\mathcal{S}$-consistent, if

- $\mathrm{cc}(\widetilde{B})$ is $\mathcal{S}$-consistent,
- for every $R \in \mathcal{R}$, for every $\widehat{D} \in \mathrm{ccs}(\widetilde{B}, R)$, $\widehat{D}$ is $\mathcal{S}$-consistent,
- for every $R \in \mathcal{R}$, for every $L \in \mathrm{cc}(\widetilde{B})$, if $(L \doteq \exists R.L') \in \mathcal{T}$, then there is a $\widehat{D} \in \mathrm{ccs}(\widetilde{B}, R)$ such that $L' \in \widehat{D}$, and
- for every $R \in \mathcal{R}$, for every $L \in \mathrm{cc}(\widetilde{B})$, if $(L \doteq \forall R.L') \in \mathcal{T}$, then for all $\widehat{D} \in \mathrm{ccs}(\widetilde{B}, R)$ it holds that $L' \in \widehat{D}$.

We say that a bicompound role $(P, \widetilde{B}_1, \widetilde{B}_2) \in \mathcal{P} \times \mathcal{B}_{\mathcal{C},\mathcal{P}} \times \mathcal{B}_{\mathcal{C},\mathcal{P}}$ is $\mathcal{S}$-consistent, if

- $\widetilde{B}_1$ and $\widetilde{B}_2$ are $\mathcal{S}$-consistent, $\mathrm{cc}(\widetilde{B}_2) \in \mathrm{ccs}(\widetilde{B}_1, P)$, $\mathrm{cc}(\widetilde{B}_1) \in \mathrm{ccs}(\widetilde{B}_2, P^-)$,
- for every $L \in \mathrm{cc}(\widetilde{B}_1)$, if $(L \doteq \forall P.L') \in \mathcal{T}$, then $L' \in \mathrm{cc}(\widetilde{B}_2)$, and
- for every $L \in \mathrm{cc}(\widetilde{B}_2)$, if $(L \doteq \forall P^-.L') \in \mathcal{T}$, then $L' \in \mathrm{cc}(\widetilde{B}_1)$.

Again, $\mathcal{S}$-consistency reflects the assertions that constitute the schema, and only $\mathcal{S}$-consistent bicompound concepts and roles can be populated in some model of $\mathcal{S}$.

The *biexpansion* $\widetilde{\mathcal{S}} = (\mathcal{C}, \mathcal{P}, \widetilde{\mathcal{B}}, \widetilde{\mathcal{U}}, \widetilde{\mathcal{T}})$ of $\mathcal{S}$ is defined as follows:

- $\widetilde{\mathcal{B}}$ is the set of all *$\mathcal{S}$-consistent* bicompound concepts.
- $\widetilde{\mathcal{U}}$ is the set of all *$\mathcal{S}$-consistent* bicompound roles.
- $\widetilde{\mathcal{T}}$ is the smallest set of bicompound assertions such that for every $\widetilde{B} \in \widetilde{\mathcal{B}}$:
  - if for some $L \in \mathrm{cc}(\widetilde{B})$ there is an assertion $(L \doteq \exists^{\geq m} R) \in \mathcal{T}$, then $\widetilde{\mathcal{T}}$ contains the bicompound assertion $\widetilde{B} \doteq \exists^{\geq m_{max}} R$, where $m_{max} := \max\{m \mid \exists L \in \mathrm{cc}(\widetilde{B}) : (L \doteq \exists^{\geq m} R) \in \mathcal{T}\}$.
  - if for some $L \in \mathrm{cc}(\widetilde{B})$ there is an assertion $(L \doteq \exists^{\leq n} R) \in \mathcal{T}$, then $\widetilde{\mathcal{T}}$ contains the bicompound assertion $\widetilde{B} \doteq \exists^{\leq n_{min}} R$, where $n_{min} := \min\{n \mid \exists L \in \mathrm{cc}(\widetilde{B}) : (L \doteq \exists^{\leq n} R) \in \mathcal{T}\}$.

It is easy to see that the size of the biexpansion is in general double exponential in the size of the original schema, and it can also be effectively constructed in double exponential time.

5.3 CHARACTERIZATION OF FINITE MODEL REASONING

We are now ready to define a system $\Psi_{\mathcal{S}}$ of linear inequations whose solutions of a certain type are related to the finite models of a schema $\mathcal{S}$. Let $\widetilde{\mathcal{S}} =$

$(\mathcal{C}, \mathcal{P}, \widetilde{\mathcal{B}}, \widetilde{\mathcal{U}}, \widetilde{\mathcal{T}})$ be the biexpansion of $\mathcal{S}$. Then $\Psi_{\mathcal{S}}$ contains one unknown $\mathrm{Var}(\widetilde{X})$ for each bicompound concept or role $\widetilde{X}$ in $\widetilde{\mathcal{B}} \cup \widetilde{\mathcal{U}}$, and consists of the following linear homogeneous inequations:

- For each $\widetilde{B} \in \widetilde{\mathcal{B}}$, for each $P \in \mathcal{P}$, for each $\widehat{D} \in \mathrm{ccs}(\widetilde{B}, P)$ the inequation

$$\mathrm{Var}(\widetilde{B}) \leq \sum_{(P,\widetilde{B},\widetilde{B}_2) \in \widetilde{\mathcal{U}} \mid \mathrm{cc}(\widetilde{B}_2) = \widehat{D}} \mathrm{Var}((P, \widetilde{B}, \widetilde{B}_2)).$$

- For each $\widetilde{B} \in \widetilde{\mathcal{B}}$, for each $P \in \mathcal{P}$, for each $\widehat{D} \in \mathrm{ccs}(\widetilde{B}, P^-)$ the inequation

$$\mathrm{Var}(\widetilde{B}) \leq \sum_{(P,\widetilde{B}_1,\widetilde{B}) \in \widetilde{\mathcal{U}} \mid \mathrm{cc}(\widetilde{B}_1) = \widehat{D}} \mathrm{Var}((P, \widetilde{B}_1, \widetilde{B})).$$

- For each bicompound assertion $(\widetilde{B} \doteq \exists^{\geq m} R) \in \widetilde{\mathcal{T}}$ the inequation

$$m \cdot \mathrm{Var}(\widetilde{B}) \leq S(\widetilde{B}, R), \qquad \text{where}$$

$$S(\widetilde{B}, P) := \sum_{(P,\widetilde{B},\widetilde{B}_2) \in \widetilde{\mathcal{U}}} \mathrm{Var}((P, \widetilde{B}, \widetilde{B}_2)),$$

$$S(\widetilde{B}, P^-) := \sum_{(P,\widetilde{B}_1,\widetilde{B}) \in \widetilde{\mathcal{U}}} \mathrm{Var}((P, \widetilde{B}_1, \widetilde{B})).$$

- For each bicompound assertion $(\widetilde{B} \doteq \exists^{\leq n} R) \in \widetilde{\mathcal{T}}$ the inequation

$$n \cdot \mathrm{Var}(\widetilde{B}) \geq S(\widetilde{B}, R).$$

It is now possible to relate a solution of $\Psi_{\mathcal{S}}$ to a finite model of $\mathcal{S}$, in which each bicompound concept and bicompound role has a number of instances that is given by the value that the solution assigns to the corresponding unknown. Since in a model of $\mathcal{S}$ each bicompound role that refers (in its second or third component) to a bicompound concept that has an empty extension, also has an empty extension, this condition reflects in an additional requirement that the solution must satisfy. A solution of $\Psi_{\mathcal{S}}$ is *acceptable*, if, whenever it assigns value 0 to an unknown corresponding to a bicompound concept $\widetilde{B}$, then it assigns also value 0 to all unknowns corresponding to bicompound roles that have $\widetilde{B}$ as their second or third component. We can now state the main theorem concerning finite concept consistency.

Theorem 6 *A concept $C \in \mathcal{C}$ is finitely consistent in $\mathcal{S}$ if and only if*

$$\Psi_{\mathcal{S}}^{C} := \Psi_{\mathcal{S}} \bigcup \left\{ \sum_{\widetilde{B} \in \widetilde{\mathcal{B}} \mid C \in \mathrm{cc}(\widetilde{B})} Var(\widetilde{B}) \geq 1 \right\}$$

admits an acceptable nonnegative integer solution.

As shown in (Calvanese, 1996b), by applying linear programming techniques one can decide the existence of acceptable nonnegative integer solutions in polynomial time in the size of the system. Since $\Psi_{\mathcal{S}}$ can be constructed in time which is at most double exponential in $|\mathcal{S}|$, and since in free $\mathcal{ALCNI}$-schemata we can easily reduce both consistency of an arbitrary concept expression and concept subsumption to consistency of a single concept name, we obtain the following worst case upper bound.

Theorem 7 *Finite concept consistency and concept subsumption in free $\mathcal{ALCNI}$-schemata can be decided in worst case deterministic double exponential time.*

The method as described above cannot deal directly with qualified number restrictions, since the differentiation introduced by bicompound concepts is too coarse. In fact, we need to make a separation in bicompound classes based not only on the existence but also on the number of links of a certain type. It turns out that it is in fact sufficient to consider only intervals of numbers of links, where the ranges of these intervals are given by the numbers that effectively appear in the schema. In this way it is still possible to keep the size of the resulting "extended" biexpansion double exponential in the size of the schema. The resulting "extended" bicompound assertions can then again be coded by means of a system of linear inequations, and we can prove the counterpart of Theorem 6 for the system derived from a free $\mathcal{ALCQI}$-schema. We omit further details and state only the final result.

Theorem 8 *Finite concept consistency and concept subsumption in free $\mathcal{ALCQI}$-schemata can be decided in worst case deterministic double exponential time.*

6 CONCLUSIONS

In this paper we have discussed the issues concerning finite model reasoning at the intensional level in expressive DLs that lack the finite model property. Two distinct ways of dealing with finiteness have been proposed:

1. We have introduced in the language a specific constructor to express well-foundedness, and have shown how known methods for reasoning with respect to arbitrary models can be extended to deal with it. This provides an **EXPTIME** decision procedure for concept consistency and subsumption.

2. We have developed a technique to reason with respect to finite models on schemata of the most general form expressed in a DL capable of capturing most known known data models. The proposed algorithm works in worst case deterministic double exponential time.

We are currently extending our work in several directions. A major point is the extension of the reasoning techniques to deal also with extensional knowledge, i.e. with assertions stating properties of specific objects. In the unrestricted case, the reasoning technique for $\mathcal{ALT}$ discussed in Section 4 can be directly integrated with the work in (De Giacomo, 1996), still obtaining an **EXPTIME** upper-bound. In the finite case, the proposed technique can be applied under specific assumptions which are rather restrictive but common in DBs. However, a general method for finite model reasoning in expressive DLs on a schema together with assertions on individuals is still open.

We aim also at increasing the expressivity of the schema definition language while still preserving decidability. Concerning point 1 above, we have mentioned in Section 3 that the well-founded constructor is equivalent to a particular type of least-fixpoint expression. A natural extension would be to allow arbitrary nested fixpoints as in the μ-calculus (Kozen, 1983) together with inverse roles and number restrictions. Concerning point 2, it can be shown that the reasoning method developed for $\mathcal{ALCQI}$ can be extended to deal with some of the constructors of $\mathcal{ALT}$, and in particular basic roles (i.e. union, intersection, and difference of role names and inverse roles) and role value maps on basic roles. Qualified number restrictions turn out to be indispensable for this. If the addition of reflexive transitive closure preserves decidability also in the finite case is still open.

Acknowledgements

I would like to thank Maurizio Lenzerini and Giuseppe De Giacomo for many inspiring discussions and Franz Baader for his hospitality at the RWTH Aachen, where part of this work was carried out.

References

Abiteboul, S. & Bonner, A. (1991). Objects and views. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pp. 238–247 New York (NY, USA).

Artale, A., Franconi, E., Guarino, N., & Pazzi, L. (1996). Part-whole relations in object-centered systems: An overview. *Data and Knowledge Engineering.* To appear.

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

Bergamaschi, S. & Sartori, C. (1992). On taxonomic reasoning in conceptual design. *ACM Transactions on Database Systems, 17*(3), 385–422.

Borgida, A. (1995). Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering, 7*(5), 671–682.

Borgida, A., Lenzerini, M., Nardi, D., & Nebel, B. (Eds.). (1995). *Working Notes of the 1995 Description Logics Workshop*, Technical Report, RAP 07.95, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Rome (Italy).

Buchheit, M., Donini, F. M., Nutt, W., & Schaerf, A. (1995). A refined architecture for terminological systems: Terminology = schema + views. Tech. rep. RR-95-09, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI).

Buongarzoni, P., Menghini, C., Salis, R., Sebastiani, F., & Straccia, U. (1995). Logical and computational properties of the description logic MIRTL.. In Borgida et al. (Borgida, Lenzerini, Nardi, & Nebel, 1995), pp. 80–84.

Calvanese, D. (1996a). Reasoning with inclusion axioms in description logics: Algorithms and complexity. In *Proc. of the 12th European Conf. on Artificial Intelligence (ECAI-96)*, pp. 303–307. John Wiley & Sons.

Calvanese, D. (1996b). *Unrestricted and Finite Model Reasoning in Class-Based Representation Formalisms.* Ph.D. thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza".

Calvanese, D., De Giacomo, G., & Lenzerini, M. (1995). Structured objects: Modeling and reasoning. In *Proc. of the 4th Int. Conf. on Deductive and Object-Oriented Databases (DOOD-95)*, No. 1013 in LNCS, pp. 229–246. Springer-Verlag.

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

Cattell, R. G. G. (Ed.). (1994). *The Object Database Standard: ODMG-93.* Morgan Kaufmann. Release 1.1.

De Giacomo, G. (1996). Tbox and abox reasoning in expressive description logics. In *Proc. of the 5th Int.*

Conf. on the Principles of Knowledge Representation and Reasoning (KR-96). Morgan Kaufmann.

De Giacomo, G. & Lenzerini, M. (1994a). Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI-94)*, pp. 205–212. AAAI Press/The MIT Press.

De Giacomo, G. & Lenzerini, M. (1994b). Concept language with number restrictions and fixpoints, and its relationship with μ-calculus. In *Proc. of the 11th European Conf. on Artificial Intelligence (ECAI-94)*, pp. 411–415.

De Giacomo, G. & Lenzerini, M. (1995). What's in an aggregate: Foundations for description logics with tuples and sets. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI-95)*, pp. 801–807.

Di Battista, G. & Lenzerini, M. (1993). Deductive entity-relationship modeling. *IEEE Transactions on Knowledge and Data Engineering, 5*(3), 439–450.

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

Donini, F. M., Lenzerini, M., Nardi, D., & Nutt, W. (1991b). Tractable concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pp. 458–463 Sydney.

Emerson, E. A. & Jutla, C. S. (1989). On simultaneously determinizing and complementing ω-automata. In *Proc. of the 4th Int. Conf. of Logic in Computer Science (LICS-89)*, pp. 333–342.

Harel, D. (1983). Recurring dominoes: Making the highly undecidable highly understandable. In *Proc. of the Int. Conf. on Foundations of Computational Theory (FCT-83)*, No. 158 in LNCS, pp. 177–194. Springer-Verlag.

Kozen, D. (1983). Results on the propositional μ-calculus. *Theoretical Computer Science, 27*, 333–354.

Kozen, D. & Tiuryn, J. (1990). Logics of programs. In *Handbook of Theoretical Computer Science - Formal Models and Semantics*, pp. 789–840. Elsevier Science Publishers (North-Holland).

Lecluse, C. & Richard, P. (1989). Modeling complex structures in object-oriented databases. In *Proc. of the 8th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS-89)*, pp. 362–369.

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

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

Sattler, U. (1995). A concept language for an engineering application with part-whole relations.. In Borgida et al. (Borgida et al., 1995), pp. 119–123.

Schild, K. (1991). A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pp. 466–471 Sydney, Australia.

Schild, K. (1994). Terminological cycles and the propositional μ-calculus. In *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-94)*, pp. 509–520 Bonn. Morgan Kaufmann.

Schmidt-Schauß, M. (1989). Subsumption in KL-ONE is undecidable. In *Proc. of the 1st Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-89)*, pp. 421–431. Morgan Kaufmann.

Streett, R. E. (1982). Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Computation, 54*, 121–141.

Thomas, W. (1990). Automata on infinite objects. In van Leuven, J. (Ed.), *Handbook of Theoretical Computer Science*, Vol. B, chap. 4, pp. 134–189. Elsevier Science Publishers (North-Holland).

Vardi, M. Y. (1985). The taming of converse: Reasoning about two-way computations. In *Proc. of the 4th Workshop on Logics of Programs*, No. 193 in LNCS, pp. 413–424. Springer-Verlag.

Vardi, M. Y. & Stockmeyer, L. (1985). Improved upper and lower bounds for modal logics of programs: Preliminary report. In *Proc. of the 17th ACM SIGACT Sym. on Theory of Computing (STOC-85)*, pp. 240–251.

Vardi, M. Y. & Wolper, P. (1986). Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences, 32*, 183–221. A preliminary version appeared in *Proc. of the 16th ACM SIGACT Symp. on Theory of Computing (STOC-84)*.

Vardi, M. Y. & Wolper, P. (1994). Reasoning about infinite computations. *Information and Computation, 115*(1), 1–37.