Description Logics with Inverse Roles, Functional Restrictions, and N-ary Relations

Giuseppe De Giacomo and Maurizio Lenzerini

Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italia
e-mail: [degiacom,lenzerini]@assi.dis.uniroma1.it

Abstract. Description Logics (DLs) are used in Artificial Intelligence to represent knowledge in terms of objects grouped into classes, and offer structuring mechanisms for both characterizing the relevant properties of classes in terms of binary relations, and establishing several interdependencies among classes. One of the main themes in the area of DLs has been to identify DLs that are both very expressive and decidable. This issue can be profitably addressed by relying on a correspondence between DLs and propositional dynamic logics (PDLs). In this paper, we exploit the correspondence as a framework to investigate the decidability and the complexity of a powerful DL, in which functional restrictions on both atomic roles and their inverse are expressible. We then show that such DL is suitable to represent n-ary relations, as needed in the applications of class-based formalisms to databases. The PDL that we use in this work is a proper extension of Converse Deterministic PDL, and its decidability and complexity is established contextually.

1 Introduction

The research in Artificial Intelligence and Computer Science has always paid special attention to formalisms for the structured representation of classes and relations. In Artificial Intelligence, the investigation on such formalisms began with semantic networks and frames, which have been influential for many formalisms proposed in the areas of knowledge representation, data bases, and programming languages, and developed towards formal logic-based languages, that will be called here *description logics*[1] (DLs). Generally speaking, DLs are decidable logics specifically designed for allowing the representation of knowledge in terms of objects (individuals) grouped into classes (concepts), and offer structuring mechanisms for characterizing the relevant properties of classes in terms of relations (roles).

Description logics have been the subject of many investigations in the last decade. It is our opinion that the main reason for investigating such logics is that they offer a clean, formal and effective framework for analyzing several important issues related to class-based representation formalisms, such as expressive power, deduction algorithms, and computational complexity of reasoning. This

[1] Terminological logics, and concept languages are other possible names.

is confirmed by the fact that the research in DLs both produced several theoretical results (see [22] for an overview), and has been influential for the design of knowledge representation systems, like LOOM [19], CLASSIC [6], and KRIS [2].

In order to use DLs as abstract formalisms for addressing diverse issues related to class-based representation schemes, they should be sufficiently general, and, at the same time, sufficiently simple so as to not fall into undecidability of reasoning. Currently, however, those DLs that have been studied from a formal point of view suffer from several limitations:

1. Relationships between classes are modeled by binary relations (roles), while n-ary relations are not supported.
2. They do not allow the modeler to refer to the inverse of a binary relation; or, if they do, they impose several restrictions in the usage of inverse relations (for example, although in general one can state that a relation is actually a function, there is no possibility to state that the inverse of a relation is a function).
3. While they offer a rich variety of constructs for building class descriptions (i.e. expressions denoting classes), they do not generally allow one to represent universal properties of classes (such as: all instances of class A must be related to at least another instance of A by means of the relation R).

All the above limitations prevent one to consider DLs general enough to capture a sufficiently broad family of class-based representation formalisms. Indeed:

1. Nonbinary relations are important in general and in particular for capturing conceptual and semantic database models (see [16]).
2. Inverse relations are essential in domain modeling (see [29]), for example, without the possibility of referring to the inverse of a relation, we are forced to use two unrelated relations *child* and *parent*, with no chance of stating their mutual dependency; also, in case inverse relations can be used in the DL, they should be used as any other relation (for example, it should be possible to state that the inverse of a relation is actually a function, analogously to the case of direct relations).
3. The possibility of expressing universal properties of classes is a basic feature for capturing both conceptual and object-oriented data models (see [9]).

Our goal in this paper is to propose a very expressive DL that both supports all the above features, and such that reasoning in the logic is decidable. To this end, we resort to the work by Schild [26], which singled out an interesting correspondence between DLs and several propositional dynamic logics (PDL), which are modal logics specifically designed for reasoning about program schemes. The correspondence is based on the similarity between the interpretation structures of the two logics: at the extensional level, objects in DLs correspond to states in PDLs, whereas connections between two objects correspond to state transitions. At the intensional level, classes correspond to propositions, and roles corresponds to programs. The correspondence is extremely useful mainly for two reasons. On

one hand, it makes clear that reasoning about assertions on classes is equivalent to reasoning about dynamic logic formulae. On the other hand, the large body of research on decision procedures in PDL (see, for example, [17]) can be exploited in the setting of DLs, and, on the converse, the various works on tractability/intractability of DLs (see for example [14]) can be used in the setting of PDL.

We argue that the work on PDLs is a good starting point for our investigation, because it provides us with:

- a general method for reasoning with universal properties of classes;
- a general method for reasoning with inverses of relations (indeed, several PDLs proposed in the literature, include a construct that exactly correspond to the inverse of relations).

However, looking carefully at the expressive power of PDLs, it turns out that the following problems need to be addressed:

1. No existing PDL allows one to impose that the inverse of relation is functional.
2. No existing PDL provides a construct that can be directly used to model nonbinary relations.

In this paper we present a solution to such problems. The solution is based on a particular methodology, which we believe has its own value: the inference in DLs is formulated in the setting of PDL, and in order to represent functional restrictions on relations (both direct and inverse), special "constraints" are added to the PDL formulae. The solution to the problem of expressing functional restrictions on both direct and inverse roles directly leads to a method for incorporating n-ary relation in DLs.

The results have a twofold significance. From the standpoint of DLs, we derive decidability and complexity results for one of the most expressive DLs appeared in the literature, and from the standpoint of PDLs, we define a very powerful PDL (it subsumes Converse Deterministic PDL), and establish its decidability and complexity by a methodology that can be exploited to derive reasoning procedures for many extensions of known PDLs - e.g. PDLs including several forms of program determinism.

The paper is organized as follows. In Section 2, we recall the basic notions of both DLs and PDLs. In Section 3, we present the result on functional restrictions, showing that Converse PDL is powerful enough to allow the representation of functional restrictions on both direct and inverse roles. In Section 4, we deal with the problem of representing n-ary relations in DLs. Finally, Section 5 ends the paper with some conclusions. For the sake of brevity, all proves are omitted.

2 Preliminaries

We base our work on two logics, namely the DL $\mathcal{CI}$, and the PDL $\mathcal{DI}$ (traditionally called Converse PDL), whose basic characteristics are recalled in this section.

The formation rules of $\mathcal{CI}$ are specified by the following abstract syntax

$$C \longrightarrow \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid C_1 \Rightarrow C_2 \mid \neg C \mid \exists R.C \mid \forall R.C$$
$$R \longrightarrow P \mid R_1 \sqcup R_2 \mid R_1 \circ R_2 \mid R^* \mid R^- \mid id(C)$$

where A denotes an atomic concept, C (possibly with subscript) denotes a concept, P denotes an atomic role, and R (possibly with subscript) denotes a role. The semantics of concepts is the usual one: an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a domain $\Delta^{\mathcal{I}}$, and an interpretation function $\cdot^{\mathcal{I}}$ that assigns subsets of $\Delta^{\mathcal{I}}$ to concepts and binary relations over $\Delta^{\mathcal{I}}$ to roles as follows:

$$\begin{array}{l}
A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}, \\
\top^{\mathcal{I}} = \Delta^{\mathcal{I}}, \\
\bot^{\mathcal{I}} = \emptyset, \\
(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} - C^{\mathcal{I}}, \\
(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}, \\
(C_1 \sqcup C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}, \\
(C_1 \Rightarrow C_2)^{\mathcal{I}} = (\neg C_1)^{\mathcal{I}} \cup C_2^{\mathcal{I}}, \\
(\exists R.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists d'.(d,d') \in R^{\mathcal{I}} \text{ and } d' \in C^{\mathcal{I}}\}, \\
(\forall R.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \forall d'.(d,d') \in R^{\mathcal{I}} \text{ implies } d' \in C^{\mathcal{I}}\}, \\
P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}, \\
(R_1 \sqcup R_2)^{\mathcal{I}} = R_1^{\mathcal{I}} \cup R_2^{\mathcal{I}}, \\
(R_1 \circ R_2)^{\mathcal{I}} = R_1^{\mathcal{I}} \circ R_2^{\mathcal{I}}, \\
(R^*)^{\mathcal{I}} = (R^{\mathcal{I}})^*, \\
(R^-)^{\mathcal{I}} = \{(d_1,d_2) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (d_2,d_1) \in R^{\mathcal{I}}\}, \\
id(C)^{\mathcal{I}} = \{(d,d) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid d \in C^{\mathcal{I}}\}.
\end{array}$$

Note that $\mathcal{CI}$ is a very expressive language, comprising all usual concept constructs, and a rich set of role constructs, namely: union of roles $R_1 \sqcup R_2$, chaining of roles $R_1 \circ R_2$, transitive closure of roles R^*, inverse roles R^-, and the identity role $id(C)$ projected on C.

A ($\mathcal{CI}$) TBox (i.e., intensional knowledge base in $\mathcal{CI}$) is defined as a finite set $\mathcal{K}$ of inclusion assertions of the form $C_1 \sqsubseteq C_2$, where C_1, C_2 are $\mathcal{CI}$-concepts. An assertion $C_1 \sqsubseteq C_2$ is satisfied by an interpretation $\mathcal{I}$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, and $\mathcal{I}$ is a model of $\mathcal{K}$ if every assertion of $\mathcal{K}$ is satisfied by $\mathcal{I}$. A TBox $\mathcal{K}$ logically implies an assertion $C_1 \sqsubseteq C_2$, written $\mathcal{K} \models C_1 \sqsubseteq C_2$, if $C_1 \sqsubseteq C_2$ is satisfied by every model of $\mathcal{K}$.

There is a direct correspondence between $\mathcal{CI}$ and the PDL $\mathcal{DI}$, whose syntax is as follows:

$$\phi \longrightarrow true \mid false \mid A \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \Rightarrow \phi_2 \mid \neg\phi_1 \mid < r > \phi_1 \mid [r]\phi_1$$
$$r \longrightarrow P \mid r_1 \cup r_2 \mid r_1; r_2 \mid r^* \mid r^- \mid \phi?$$

where A denotes a propositional letter, ϕ (possibly with subscript) denotes a formula, P denotes an atomic program, and r (possibly with subscript) denotes a program. The semantics of $\mathcal{DI}$ is based on the notion of (Kripke) structure, which is defined as a triple $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$, where $\mathcal{S}$ denotes a set of states, $\{R_P\}$ is a family of binary relations over $\mathcal{S}$ such that each atomic program P is given

a meaning through R_P, and Π is a mapping from $\mathcal{S}$ to propositional letters such that $\Pi(s)$ determines the letters that are true in the state s. Given M, the family $\{\mathcal{R}_P\}$ can be extended in the obvious way so as to include, for every program r, the corresponding relation $\mathcal{R}_r$ (for example, $\mathcal{R}_{r_1;r_2}$ is the composition of $\mathcal{R}_{r_1}$ and $\mathcal{R}_{r_2}$). For this reason, we often denote a structure by $(\mathcal{S}, \{\mathcal{R}_r\}, \Pi)$, where $\{\mathcal{R}_r\}$ includes a binary relations for every program (atomic or non-atomic). A structure M is called a model of a formula ϕ if there exists a state s in M such that $M, s \models \phi$. A formula ϕ is satisfiable if there exists a model of ϕ, unsatisfiable otherwise.

The correspondence between $\mathcal{CI}$ and $\mathcal{DI}$, first pointed out by Schild [26], is based on the similarity between the interpretation structures of the two logics: at the extensional level, individuals (members of $\Delta^{\mathcal{I}}$) in DLs correspond to states in PDLs, whereas connections between two individuals correspond to state transitions. At the intensional level, classes correspond to propositions, and roles corresponds to programs. The correspondence is realized through a (one-to-one and onto) mapping δ from $\mathcal{CI}$-concepts to $\mathcal{DI}$-formulae, and from $\mathcal{CI}$-roles to $\mathcal{DI}$-programs. The mapping δ is defined inductively as follows (we assume $\sqcup, \Rightarrow$ to be expressed by means of $\sqcap, \neg$):

$$
\begin{array}{ll}
\delta(A) = A & \delta(P) = P \\
\delta(C_1 \sqcap C_2) = \delta(C_1) \wedge \delta(C_2) & \delta(\neg C) = \neg\delta(C) \\
\delta(\exists R.C) =< \delta(R) > \delta(C) & \delta(\forall R.C) = [\delta(R)]\delta(C) \\
\delta(R_1 \sqcup R_2) = \delta(R_1) \cup \delta(R_2) & \delta(R_1 \circ R_2) = \delta(R_1); \delta(R_2) \\
\delta(R^*) = \delta(R)^* & \delta(id(C)) = \delta(C)? \\
\delta(R^-) = \delta(R)^- . &
\end{array}
$$

From δ one can easily obtain a mapping δ^+ from $\mathcal{CI}$-TBoxes to $\mathcal{DI}$-formulae. Namely, if $\mathcal{K} = \{K_1, \cdots, K_n\}$ is a TBox in $\mathcal{CI}$, and $P_1, \ldots, P_m$ are all atomic roles appearing in $\mathcal{K}$, then

$$
\begin{array}{l}
\delta^+(\mathcal{K}) = [(P_1 \cup \cdots \cup P_m \cup P_1^- \cdots \cup P_m^-)^*]\ \delta^+(\{K_1\}) \wedge \cdots \wedge \delta^+(\{K_n\}), \\
\delta^+(\{C_1 \sqsubseteq C_2\}) = (\delta(C_1) \Rightarrow \delta(C_2)).
\end{array}
$$

Observe that $\delta^+(\mathcal{K})$ exploits the power of program constructs (union, converse, and transitive closure) and the "connected model property"[2] of PDLs in order to represent inclusion assertions of DLs. Based on this correspondence, we can state the following: if $\mathcal{K}$ is a TBox, then $\mathcal{K} \models C_1 \sqsubseteq C_2$ (where atomic concepts and roles in C_1, C_2 are also in $\mathcal{K}$) iff the $\mathcal{DI}$-formula

$$\delta^+(\mathcal{K}) \wedge \delta(C_1) \wedge \delta(\neg C_2)$$

is unsatisfiable. Note that the size of the above formula is polynomial with respect to the size of $\mathcal{K}, C_1$ and C_2.

By virtue of δ and δ^+, respectively, both satisfiability of $\mathcal{CI}$ concepts, and logical implication for $\mathcal{CI}$-TBoxes, can be (polynomially) reduced to satisfiability of $\mathcal{DI}$-formulae. Being satisfiability for $\mathcal{DI}$ an EXPTIME-complete problem,

[2] That is, if a formula has a model, it has a model which is connected.

so are satisfiability of $\mathcal{CI}$-concepts and logical implication for $\mathcal{CI}$-TBoxes. It is straightforward to extend the correspondence, and hence both δ and δ^+, to other DLs and PDLs.

In the rest of this section, we introduce several notions and notations that will be used in the sequel.

The *Fisher-Ladner closure* ([10]) of a $\mathcal{DI}$-formula Φ, denoted $CL(\Phi)$, is the least set F such that $\Phi \in F$ and such that (we assume, without loss of generality, $\vee, \Rightarrow, [\cdot]$ to be expressed by means of $\neg, \wedge, < \cdot >$, and the converse operator to be applied to atomic programs only[3]):

$$\begin{array}{ll}
\phi_1 \wedge \phi_2 \in F & \Rightarrow \phi_1, \phi_2 \in F, \\
\neg\phi \in F & \Rightarrow \phi \in F, \\
< r > \phi \in F & \Rightarrow \phi \in F, \\
< r_1; r_2 > \phi \in F & \Rightarrow < r_1 >< r_2 > \phi \in F, \\
< r_1 \cup r_2 > \phi \in F & \Rightarrow < r_1 > \phi, < r_2 > \phi \in F, \\
< r^* > \phi \in F & \Rightarrow < r >< r^* > \phi \in F, \\
< \phi'? > \phi \in F & \Rightarrow \phi' \in F.
\end{array}$$

The notion of Fisher-Ladner closure can be easily extended to formulae of other PDLs.

A *path* in a structure M (sometimes called *trajectory*) is a sequence $(s_0, \ldots, s_q)$ of states of M, such that for each $i = 1, \ldots, q$, $(s_{i-1}, s_i) \in \mathcal{R}_a$ for some $a = P \mid P^-$. The length of $(s_0, \ldots, s_q)$ is q. We inductively define the set of paths $Paths(r)$ of a program r in a structure M, as follows (we assume, without loss of generality, that in r all occurrences of the converse operator are moved all way in):

$$\begin{array}{ll}
Paths(a) & = \mathcal{R}_a \ (a = P \mid P^-), \\
Paths(r_1 \cup r_2) & = Paths(r_1) \cup Paths(r_2), \\
Paths(r_1; r_2) & = \{(s_0, \ldots, s_u, \ldots, s_q) \mid (s_0, \ldots, s_u) \in Paths(r_1) \\
 & \quad \text{and } (s_u, \ldots, s_q) \in Paths(r_2)\}, \\
Paths(r^*) & = \{(s) \mid s \in \mathcal{S}\} \cup (\bigcup_{i>0} Paths(r^i)), \\
Paths(\phi'?) & = \{(s) \mid M, s \models \phi'\}.
\end{array}$$

We say that a path (s_0) in M *satisfies* a formula ϕ which is not of the form $< r > \phi'$ if $M, s_0 \models \phi$. We say that a path $(s_0, \ldots, s_q)$ in M *satisfies* a formula ϕ of the form $< r_1 > \cdots < r_l > \phi'$, where ϕ' is not of the form $< r' > \phi''$, if $M, s_q \models \phi'$ and $(s_0, \ldots s_q) \subseteq Paths(r_1; \cdots; r_l)$.

Finally, if a denotes the atomic program P (resp. the inverse of an atomic program P^-), then we write a^- to denote P^- (resp. P).

3 Functional Restrictions

In this section, we consider an extension of $\mathcal{CI}$, called $\mathcal{CIF}$, which is obtained from $\mathcal{CI}$ by adding the concept construct $(\leq 1\ a)$, where $a = P \mid P^-$. The meaning of the construct in an interpretation $\mathcal{I}$ is the following:

[3] We recall that the following equations hold: $(r_1; r_2)^- = r_1^-; r_2^-$, $(r_1 \cup r_2)^- = r_1^- \cup r_2^-$, $(r_1^*)^- = (r_1^-)^*$, $(\phi?)^- = (\phi?)$.

$$(\leq 1\ a)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \text{there exists } \textit{at most one } d' \text{ such that } (d, d') \in a^{\mathcal{I}}\}.$$

The corresponding PDL will be called $\mathcal{DIF}$, and is obtained from $\mathcal{DI}$ by adding the same construct $(\leq 1\ a)$, where, again, $a = P \mid P^-$, whose meaning in $\mathcal{DIF}$ can be immediately derived by the semantics of $\mathcal{CIF}$. Observe that the construct $(\leq 1\ a)$ allows the notion of local determinism for both atomic programs and their converse to be represented in PDL. With this construct, we can denote states from which the running of an atomic program or the converse of an atomic program is deterministic, i.e., it leads to at most one state. It is easy to see that this possibility allows one to impose the so-called global determinism too, i.e., that a given atomic program, or the converse of an atomic program, is (globally) deterministic. Therefore, $\mathcal{DIF}$ subsumes the logic studied in [30], called Converse Deterministic PDL where atomic programs, *not their converse*, are (globally) deterministic.

From the point of view of DLs, as mentioned in the Introduction, the fact that in the $(\leq 1\ a)$ construct, a can be either an atomic role or the inverse of an atomic role, greatly enhances the expressive power of the logic, and makes $\mathcal{CIF}$ one of the most expressive DLs among those studied in the literature.

The decidability and the complexity of both satisfiability of $\mathcal{CIF}$-concepts and logical implication for $\mathcal{CIF}$-TBox, can be derived by exploiting the correspondence between $\mathcal{CIF}$ and $\mathcal{DIF}$. This is realized through the mappings δ and δ^+ described in Section 2, suitably extended in order to deal with functional restrictions.

Note however that the decidability and the complexity of satisfiability in $\mathcal{DIF}$ are to be established, yet. We establish them below by showing an encoding of $\mathcal{DIF}$-formulae in $\mathcal{DI}$. More precisely we show that, for any $\mathcal{DIF}$-formula Φ, there is a $\mathcal{DI}$-formula, denoted $\gamma(\Phi)$, whose size is polynomial with respect to the size of Φ, and such that Φ is satisfiable iff $\gamma(\Phi)$ is satisfiable. Since satisfiability in $\mathcal{DI}$ is EXPTIME-complete, this ensures us that satisfiability in $\mathcal{DIF}$, and therefore both satisfiability of $\mathcal{CIF}$-concepts and logical implication for $\mathcal{CIF}$-TBoxes, are EXPTIME-complete too.[4] In what follows, we assume, without loss of generality, that $\mathcal{DIF}$-formula Φ is in negation normal form (i.e., negation is pushed inside as much as possible). We define the *$\mathcal{DI}$-counterpart* $\gamma(\Phi)$ of the $\mathcal{DIF}$-formula Φ as the conjunction of two formulae, $\gamma(\Phi) = \gamma_1(\Phi) \wedge \gamma_2(\Phi)$, where:

- $\gamma_1(\Phi)$ is obtained from Φ by replacing each $(\leq 1\ a)$ with a new propositional letter $A_{(\leq 1\ a)}$, and each $\neg(\leq 1\ a)$ with $(< a > H_{(\leq 1\ a)}) \wedge (< a > \neg H_{(\leq 1\ a)})$, where $H_{(\leq 1\ a)}$ is again a new propositional letter.
- $\gamma_2(\Phi) = [(P_1 \cup \cdots \cup P_m \cup P_1^- \cdots \cup P_m^-)^*]\gamma_2^1 \wedge \cdots \wedge \gamma_2^q$, where $P_1, \ldots, P_m$ are all atomic roles appearing in Φ, and with one conjunct γ_2^i of the form

$$((A_{(\leq 1\ a)} \wedge < a > \phi) \Rightarrow [a]\phi)$$

  for every $A_{(\leq 1\ a)}$ occurring in $\gamma_1(\Phi)$ and every $\phi \in CL(\gamma_1(\Phi))$.

[4] Indeed $\gamma(\delta^+(\mathcal{K}) \wedge \delta(C_1) \wedge \delta(\neg C_2))$ is the $\mathcal{DIF}$-formula corresponding to the implication problem $\mathcal{K} \models C_1 \sqsubseteq C_2$ for $\mathcal{CIF}$-TBoxes.

Intuitively $\gamma_2(\Phi)$ constrains the models M of $\gamma(\Phi)$ so that: for every state s of M, if $A_{(\leq\ 1\ a)}$ holds in s, and there is an a-transition from s to t_1 and an a-transition from s to t_2, then t_1 and t_2 are equivalent wrt the formulae in $CL(\gamma_1(\Phi))$. We show that this allow us to actually collapse t_1 and t_2 into a single state.

To prove that a $\mathcal{DIF}$-formula is satisfiable iff its $\mathcal{DI}$-counterpart is, we proceed as follows. Given a model $M = (\mathcal{S}, \{R_r\}, \Pi)$ of $\gamma(\Phi)$, we build a tree-like structure $M^t = (\mathcal{S}^t, \{R_r^t\}, \Pi^t)$ such that $M^t, root \models \gamma(\Phi)$ ($root \in S^t$ is the root of the tree-structure), and the local determinism requirements are satisfied. From such M^t, a model $M^t_{\mathcal{F}}$ of Φ can easily be derived. In order to construct M^t we make use of the following notion: For each state s in M, we call by $ES(s)$ the smallest set of states in M such that

- $s \in ES(s)$, and
- if $s' \in ES(s)$, then for every s'' such that $(s', s'') \in \mathcal{R}_{a;A_{(\leq 1\ a^-)}?;a^-}$, $ES(s'') \subseteq ES(s)$.

The set $ES(s)$ is the set of states of M that are going to be collapsed into a single state of M^t. Note that, by $\gamma_2(\Phi)$, all the states in $ES(s)$ satisfy the same formulae in $CL(\gamma_1(\Phi))$. The construction of M^t is done in three stages.

Stage 1. Let $< a_1 > \psi_1, \ldots, < a_h > \psi_h$ be all the formulas of the form $< a > \phi'$ included in $CL(\Phi)$.[5] We consider an infinite h-ary tree $\mathcal{T}$ whose root is $root$ and such that every node x has h children $child_i(x)$, one for each formula $< a_i > \psi_i$. We write $father(x)$ to denote the father of a node x in $\mathcal{T}$. We define two partial mappings m and l: m maps nodes of $\mathcal{T}$ to states of M, and l is used to label the arcs of $\mathcal{T}$ by either atomic programs, converse of atomic programs, or a special symbol 'undefined'. For the definition of m and l, we proceed level by level. Let $s \in \mathcal{S}$ be any state such that $M, s \models \gamma(\Phi)$. We put $m(root) = s$, and for all arcs $(root, child_i(root))$ corresponding to a formula $< a_i > \psi_i$ such that $M, s \models < a_i > \psi_i$ we put $l((root, child_i(root))) = a_i$. Suppose we have defined m and l up to level k, let x be a node at level $k+1$, and let $l((father(x), x)) = a_j$. Then $M, m(father(x)) \models < a_j > \psi_j$, and therefore, there exists a path $(s_o, s_1, \ldots, s_q)$, with $s_o = m(father(x))$ satisfying $< a_j > \psi_j$. Among the states in $ES(s_1)$ we choose a state t such that there exists a *minimal* path (i.e., a path with minimal length) from t satisfying ψ_j. We put $m(x) = t$ and for every $< a_i > \psi_i \in CL(\Phi)$ such that $M, t \models < a_i > \psi_i$ we put $l((x, child_i(x))) = a_i$.

Stage 2. We change the labeling l, proceeding level by level. If $M, m(root) \models A_{(\leq\ 1\ a)}$, then for each arc $(root, child_i(root))$ labeled a, except for one randomly chosen, we put $l((root, child_i(root)) =$ 'undefined'. Assume we have modified l up to level k, and let x be a node at level $k+1$. Suppose $M, m(x) \models A_{(\leq\ 1\ a)}$. Then if $l((father(x), x)) = a^-$, for each arc $(x, child_i(x))$ labeled a, we put $l((x, child_i(x)) =$ 'undefined', otherwise (i.e. $l((father(x), x)) \neq a^-$) we put $l((x, child_i(x)) =$ 'undefined' for every arc $(x, child_i(x))$ labeled a, except for one randomly chosen.

Stage 3. For each P, let $\mathcal{R}'_P = \{(x, y) \in \mathcal{T} \mid l((x, y)) = P \text{ or } l((y, x)) = P^-\}$. We define the structure $M^t = (\mathcal{S}^t, \{\mathcal{R}_r^t\}, \Pi^t)$ as follows: $\mathcal{S}^t = \{x \in$

[5] Notice that the formulas ψ_i may be of the form $< r > \phi$, and that $\psi_i \in CL(\Phi)$.

$\mathcal{T} \mid (root, x) \in (\bigcup_P (\mathcal{R}'_P \cup \mathcal{R}'^{-}_P))^*\}$, $\mathcal{R}^t_P = \mathcal{R}'_P \cap (\mathcal{S}^t \times S^t)$, and $\Pi^t(x) = \Pi(m(x))$ $(\forall x.x \in \mathcal{S}^t)$. From $\{\mathcal{R}^t_P\}$ we get all $\{\mathcal{R}^t_r\}$ as usual.

The basic property of M^t is stated in the following lemma.

Lemma 1. *Let Φ be a $\mathcal{DIF}$-formula, and let M be a model of $\gamma(\Phi)$. Then, for every formula $\phi \in CL(\gamma_1(\Phi))$ and every $x \in \mathcal{S}^t$, $M^t, x \models \phi$ iff $M, m(x) \models \phi$.*

From M^t, we can define a new structure $M^t_{\mathcal{F}} = (\mathcal{S}^t_{\mathcal{F}}, \{\mathcal{R}^t_{\mathcal{F}r}\}, \Pi^t_{\mathcal{F}})$ where, $\mathcal{S}^t_{\mathcal{F}} = \mathcal{S}^t$, $\{\mathcal{R}^t_{\mathcal{F}r}\} = \{\mathcal{R}^t_r\}$, and $\Pi^t_{\mathcal{F}}(x) = \Pi^t(x) - \{A_{(\leq 1\ a)}, H_{(\leq 1\ a)}\}$, for each $x \in \mathcal{S}^t_{\mathcal{F}}$. The structure $M^t_{\mathcal{F}}$ has the following property.

Lemma 2. *Let Φ be a $\mathcal{DIF}$-formula, and let $M^t, M^t_{\mathcal{F}}$ be obtained from a model M of $\gamma(\Phi)$ as specified above. Then $M^t, root \models \gamma_1(\Phi)$ implies $M^t_{\mathcal{F}}, root \models \Phi$.*

Considering that every model of Φ can be easily transformed in a model of $\gamma(\Phi)$ we can state the main result of this section.

Theorem 3. *A $\mathcal{DIF}$-formula Φ is satisfiable iff its $\mathcal{DI}$-counterpart $\gamma(\Phi)$ is satisfiable.*

Corollary 4. *Satisfiability in $\mathcal{DIF}$ and both satisfiability of $\mathcal{CIF}$-concepts and logical implication for $\mathcal{CIF}$-TBoxes are EXPTIME-complete problems.*

The fact that $\mathcal{DIF}$-formulae can be encoded in $\mathcal{DI}$, calls for some comments. Notice that $\mathcal{DI}$-formulae have always a finite model M (finite model property) while $\mathcal{DIF}$-formulae don't - e.g. the formula $A \wedge [(P^-)^*]((\leq 1\ P) \wedge < P^- > \neg A)$ does not have any finite model [6]. Indeed, M^t, and thus $M^t_{\mathcal{F}}$, built from a finite model M are not finite in general.

It is also interesting to observe that, since $\mathcal{DIF}$ subsumes Converse Deterministic PDL, also formulae of that logic can be encoded in $\mathcal{DI}$. This fact gives us procedures to decide satisfiability of Converse Deterministic PDL formulae that do not rely on techniques based on automata on infinite structures as those in [30].

Finally, the construction above can be easily modified/restricted to encode Deterministic PDL formulae in PDL. In fact, the original construction, used in [3] to study satisfiability of Deterministic PDL, is similar in the spirit, though not in the development, to such a restricted version of the our construction.

In concluding the section we would like to present some examples of $\mathcal{CIF}$ concepts, that demonstrate the power of this DL. The examples concern the definition of concepts denoting common data structures in computer science. The first example regards lists. A LIST can be (inductively) defined as: a NIL is a LIST, a NODE that as exactly one successor that is a LIST, is a LIST. From this definition it follows that a list is a chain of NODEs of any length, that

[6] This formula is a variant of the Converse Deterministic PDL formula $A \wedge [(P^-)^*] < P^- > \neg A$ (see [30]).

terminates with a NIL. Therefore, we can denote the class of LISTs as (we use $C_1 \doteq C_2$ as a shorthand for $C_1 \sqsubseteq C_2, C_2 \sqsubseteq C_1$):

$$List \doteq \exists(id(Node \sqcap (\leq\ 1\ succ)) \circ succ)^*.Nil.$$

The second example concerns (possible infinite) trees. A TREE is formed by a single NODE, that has no father (the root), whose all children are inner NODEs of a TREE, where an inner NODE of a TREE is a NODE having exactly one father, whose children are themselves all inner NODEs of a TREE. This definition implies that TREEs are formed by a NODE that has no father and such that all its descendants are NODEs having exactly one father. Note that infinite TREEs are allowed. The $\mathcal{CIF}$ concept corresponding to this definition of TREE is

$$Tree \doteq \forall child^-.\bot \sqcap (\forall child^*.(Node \sqcap (\leq\ 1\ child^-))).$$

As last example, we specialize the above definition of TREE, to BINARY-TREE where left and right subtrees are identified through different roles. That is, BINARY-TREEs are TREEs such that each NODE has at most one LEFT child and at most one RIGHT child. The corresponding $\mathcal{CIF}$-concept is

$$\begin{aligned} BinTree \doteq\ & (\forall left^-.\bot) \sqcap (\forall right^-.\bot) \sqcap \\ & (\forall (left \sqcup right)^*.(Node \sqcap (\leq\ 1\ left) \sqcap (\leq\ 1\ right) \sqcap \\ & (\leq\ 1\ left^-) \sqcap (\leq\ 1\ right^-) \sqcap ((\forall left^-.\bot) \sqcup (\forall right^-.\bot)))). \end{aligned}$$

Observe that, in order to fully capture the latter two concepts, we need to make use of functional restrictions on both atomic roles and inverse of atomic roles. To the best of our knowledge, $\mathcal{CIF}$ is the only DL allowing for a correct and precise definition of TREE and BINARY-TREE.

4 N-ary Relations

In this section we extend $\mathcal{CIF}$ by means of suitable mechanisms to aggregate individuals in *tuples*. Each tuple has an associated *arity* which is the number of individuals constituting the tuple. Tuples of the same arity n can be grouped into sets forming *n-ary relations*.

An n-ary relation is described by a *name* and n relation roles (r-roles in the following). Each r-role names a component of the relation, i.e., a component of each of its tuples. For each relation $\mathbf{R}$ the set of its r-roles is denoted by $rol(\mathbf{R})$. The cardinality of this set is greater or equal to 2, and implicitly determinates the arity of $\mathbf{R}$. We call "U-component" the component of $\mathbf{R}$ named by the r-role $U \in rol(\mathbf{R})$.

We present a DL, called $\mathcal{CIFR}$, with suitable constructs to deal with relations, having the following abstract syntax:

$$\begin{aligned} C \longrightarrow\ & \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid C_1 \Rightarrow C_2 \mid \neg C \mid \forall R.C \mid \exists R.C \mid \\ & (\leq\ 1\ P) \mid (\leq\ 1\ P^-) \mid (\leq\ 1\ \mathbf{R}[U]) \mid \\ & \forall \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m \mid \exists \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m \\ R \longrightarrow\ & P \mid \mathbf{R}[U, U'] \mid R_1 \sqcup R_2 \mid R_1 \circ R_2 \mid R^* \mid R^- \mid id(C). \end{aligned}$$

The intuitive meaning of the new constructs is explained below (the other constructs have the usual meaning).

- $\mathbf{R}[U]$ denotes the relation between individuals d and tuples of $\mathbf{R}$ that have d as U-component - i.e., the inverse of the function projecting $\mathbf{R}$ onto its U-component.
- $\mathbf{R}[U, U']$ denotes the function projecting the relation $\mathbf{R}$ onto its U, U' components.
- $(\leq\ 1\ \mathbf{R}[U])$ represents the individuals d that occur at most once as U-component of the relation $\mathbf{R}$.[7]
- $\forall \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m$ represents the set of individuals x such that for each tuple r of $\mathbf{R}$ with x as U-component, the T_i-component of r belongs to the extension of C_i $(i = 1, \ldots, m)$.
- $\exists \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m$ represents the set of individuals x such that there exists a tuple r of $\mathbf{R}$ with x as U-component and x_i $(i = 1, \ldots, m)$ as T_i-component such that x_i belongs to the extension of C_i.

The semantics of $\mathcal{CIFR}$ is given, as usual, through an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, now extended to interpret relations and the new constructs. In particular, if $\mathbf{R}$ is a (n-ary) relation whose set of r-roles is $rol(\mathbf{R}) = \{U_1, \ldots, U_n\}$, then $\mathbf{R}^{\mathcal{I}}$ is a set of labeled tuples of the form $< U_1 : d_1, \ldots, U_n : d_n >$ where $d_1, \ldots, d_n \in \Delta^{\mathcal{I}}$. We write $r[U]$ to denote the value associated with the U-component of the tuple r. The new constructs are interpreted as follows:

- $\mathbf{R}[U]^{\mathcal{I}} = \{(d, r) \in \Delta^{\mathcal{I}} \times \mathbf{R}^{\mathcal{I}} \mid d = r[U]\}$.
- $\mathbf{R}[U, U']^{\mathcal{I}} = \{(d, d') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \exists r \in \mathbf{R}^{\mathcal{I}}.d = r[U] \wedge d' = r[U']\}$.
- $(\leq\ 1\ \mathbf{R}[U])^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid$ there exists at most one $r \in \mathbf{R}^{\mathcal{I}}$ such that $r[U] = d\}$.
- $(\forall \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \forall r \in \mathbf{R}^{\mathcal{I}}.r[U] = d \Rightarrow (r[T_1] \in C_1^{\mathcal{I}} \wedge \cdots \wedge r[T_m] \in C_m^{\mathcal{I}})\}$.
- $(\exists \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists r \in \mathbf{R}^{\mathcal{I}}.r[U] = d \wedge r[T_1] \in C_1^{\mathcal{I}} \wedge \cdots \wedge r[T_m] \in C_m^{\mathcal{I}}\}$.

$\mathcal{CIFR}$-TBoxes are defined as a finite set of inclusion assertions $C_1 \sqsubseteq C_2$, where C_1, C_2 are $\mathcal{CIFR}$-concepts. Satisfiability of $\mathcal{CIFR}$-concepts, as well as logical implication, in $\mathcal{CIFR}$-TBoxes is defined as usual.

Let us show some examples of use of $\mathcal{CIFR}$. Consider the relation **Parents**, with $rol(\mathbf{Parents}) = \{child, father, mother\}$, denoting the set of tuples child and his/her (natural) parents (father and mother). An inclusion assertion regarding this relation can be:

$$Human \sqsubseteq \forall \mathbf{Parents}[child].father : Human, mother : Human$$

stating that both the father and the mother of a child, who is human, must be human as well (more precisely, every individual who is $Human$ is such that, if

[7] Note that $\mathcal{CIFR}$ does not include the concept construct $(\leq\ 1\ \mathbf{R}[U]^-)$, because, by definition, $\mathbf{R}[U]^-$ is always functional.

(s)he participates, as *child*-component, in a tuple r of the relation **Parents**, then both the $father$-component of r and the $mother$-component of r are $Human$). Note that, in order to represent the (natural) parents of a child, the relation **Parent** must be so that child has exactly one father and one mother in the relation **Parents** - that is, individuals may occur as *child*-component in at most one tuple of the relation. This fact can easily be represented in $\mathcal{CIFR}$ by asserting that:

$$\top \sqsubseteq (\leq\ 1\ \mathbf{Parents}[child]).$$

Next we investigate the decidability and the complexity of the reasoning tasks for $\mathcal{CIFR}$. For ease of exposition, we concentrate on satisfiability of $\mathcal{CIFR}$-TBoxes. In fact, it is easy to check that, satisfiability of $\mathcal{CIFR}$-TBoxes and logical implication in $\mathcal{CIFR}$-TBoxes are (linearly) reducible one into the other, and satisfiability of $\mathcal{CIFR}$-concepts is a subcase of both of them. We show that there exists a one-to-one mapping from $\mathcal{CIFR}$-TBoxes $\mathcal{K}$ to $\mathcal{CIF}$-TBoxes $\mathcal{K}'$ such that $\mathcal{K}$ is satisfiable if and only if $\mathcal{K}'$ is satisfiable. To define this mapping we make use of an auxiliary mapping t.

The mapping t is defined inductively, in the obvious way for the common constructs, and as follows, for the new ones:

$t(\mathbf{R}[U]) = P^{-}_{\mathbf{R}[U]}$ ($P_{\mathbf{R}[U]}$ is a new atomic role)
$t(\mathbf{R}[U,U']) = P^{-}_{\mathbf{R}[U]} \circ P_{\mathbf{R}[U']}$
$t((\leq\ 1\ \mathbf{R}[U])) = (\leq\ 1\ P^{-}_{\mathbf{R}[U]})$
$t(\forall \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m) = \forall P^{-}_{\mathbf{R}[U]}.\exists P_{\mathbf{R}[T_1]}.t(C_1) \sqcap \ldots \sqcap \exists P_{\mathbf{R}[T_m]}.t(C_m)$
$t(\exists \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m) = \exists P^{-}_{\mathbf{R}[U]}.\exists P_{\mathbf{R}[T_1]}.t(C_1) \sqcap \ldots \sqcap \exists P_{\mathbf{R}[T_m]}.t(C_m).$

Inclusion assertions $C_1 \sqsubseteq C_2$ are mapped to $t(C_1) \sqsubseteq t(C_2)$.

Let us call $t(\mathcal{K})$ the TBox thus obtained. From $t(\mathcal{K})$ we obtain $\mathcal{K}'$ by adding to it the following inclusion assertions:

1. $\top \sqsubseteq (\leq\ 1\ P_{\mathbf{R}[U]})$ for all roles $P_{\mathbf{R}[U]}$.
2. $\exists P_{\mathbf{R}[U]}.\top \sqsubseteq \exists P_{\mathbf{R}[U_1]}.\top \sqcap \ldots \sqcap \exists P_{\mathbf{R}[U_n]}.\top$ where $U \in rol(\mathbf{R})$ and $rol(\mathbf{R}) = \{U_1, \ldots, U_n\}$, for all roles $P_{\mathbf{R}[U]}$.

The inclusion assertions (1) constrain the models of $\mathcal{K}'$ so that each $P_{\mathbf{R}[U]}$ is (globally) functional. The inclusion assertions (2) constrain the models of $\mathcal{K}'$ so that if an individual has a link that is an instance of $P_{\mathbf{R}[U]}$ then it also has links that are instance of $P_{\mathbf{R}[U_i]}$ (for all $U_i \in rol(\mathbf{R})$). Indeed, (1) and (2) allow us to represent a n-ary relation $\mathbf{R}$ by the concept $\exists P_{\mathbf{R}[U_1]}.\top \sqcap \ldots \sqcap \exists P_{\mathbf{R}[U_n]}.\top$, i.e., the tuples of $\mathbf{R}$ are represented by instances of $\exists P_{\mathbf{R}[U_1]}.\top \sqcap \ldots \sqcap \exists P_{\mathbf{R}[U_n]}.\top$. Observe that this representation is accurate only in the models $\mathcal{I}$ of $\mathcal{K}'$ where tuples of $\mathbf{R}$ corresponds to a single individual, otherwise, in $\mathcal{I}$ there would be two individuals representing the same tuple. However, we can show that if $\mathcal{K}'$ admits a model, then it admits a model satisfying the above condition. Formally, the following lemma holds.

Lemma 5. *The $\mathcal{CIF}$-TBox $\mathcal{K}'$ obtained by the above construction is satisfiable if and only if it has a model $\mathcal{I}$ satisfying the constraint:*

$$d, d' \in (\exists P_{\mathbf{R}_{[U_1]}}.\top \sqcap \ldots \sqcap \exists P_{\mathbf{R}_{[U_n]}}.\top)^{\mathcal{I}} \Rightarrow \\ \neg((d, d_1) \in (P_{\mathbf{R}_{[U_1]}})^{\mathcal{I}} \wedge (d', d_1) \in (P_{\mathbf{R}_{[U_1]}})^{\mathcal{I}} \wedge \\ \ldots \wedge (d', d_n) \in (P_{\mathbf{R}_{[U_n]}})^{\mathcal{I}} \wedge (d', d_n) \in (P_{\mathbf{R}_{[U_n]}})^{\mathcal{I}})$$

for every n-ary relation $\mathbf{R}$ with $rol(\mathbf{R}) = \{U_1, \ldots, U_n\}$.

Now, we are ready to state the desired result.

Theorem 6. *A $\mathcal{CIFR}$-TBox $\mathcal{K}$ is satisfiable if and only if the $\mathcal{CIF}$-TBox $\mathcal{K}'$ defined as above is satisfiable.*

Considering that $\mathcal{K}'$ is polynomially bounded to $\mathcal{K}$, the decidability and the complexity of reasoning in $\mathcal{CIFR}$ are an immediate consequence of the results in the previous section.

Corollary 7. *Satisfiability of $\mathcal{CIFR}$-TBoxes, logical implication for $\mathcal{CIFR}$-TBoxes, satisfiability of $\mathcal{CIFR}$-concepts, are EXPTIME-complete problems.*

5 Discussion and Conclusion

By exploiting the correspondence between DLs and PDLs, we have analyzed the decidability and the complexity of a very expressive DL, $\mathcal{CIF}$, which includes functional restrictions on both atomic roles and their inverse. On top of $\mathcal{CIF}$ we have been able to design constructs involving n-ary relations, thus presenting a DL, $\mathcal{CIFR}$, whose characteristics are quite unusual in the context of Frame Based Languages, and more typical of other class-based formalisms such as Semantics Data Models or Object-Oriented Data Models.

It is possible to show that our results on functional restrictions extend to full *qualified number restrictions* (generalizations of functional restrictions stating the minimum and the maximum number of links between instances of classes and instances of another concept through a specified role or relation), by which general cardinality constrains on components of relations can be expressed.

We conclude remarking that, the issues presented in this paper can be relevant also in the setting of Modal Mu-Calculus, a logic of programs which includes explicit constructs for least and greatest fixpoints of formulae (PDL is a fragment of it), that has been recently used to model, in a single framework, terminological cycles interpreted according to Least Fixpoint Semantics, Greatest Fixpoint Semantics, and Descriptive Semantics (see [23, 27, 11]).

References

1. G. Attardi and M. Simi. Consistency and completeness of omega, a logic for knowledge representation. In *Proc. of the Int. Joint Conf. on Artificial Intelligence IJCAI-81*, pages 504–510, 1981.

2. F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithm. In *Proc. of the Workshop on Processing Declarative Knowledge, PDK-91*, Lecture Notes in Artificial Intelligence, pages 67–86. Springer-Verlag, 1991.
3. M. Ben-Ari, J. Y. Halpern, and A. Pnueli. Deterministic propositional dynamic logic: Finite models, complexity, and completeness. *Journal of Computer and System Sciences*, 25:402–417, 1982.
4. A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. Submitted for publication, 1993.
5. R. J. Brachman and H.J. Levesque. The tractability of subsumption in frame-based description languages. In *Proc. of the 4th Nat. Conf. on Artificial Intelligence AAAI-84*, pages 34–37, 1984.
6. R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. Alperin Resnick, and A. Borgida. Living with CLASSIC: when and how to use a KL-ONE-like language. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, 1991.
7. R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.
8. M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. In *Proc. of the 13th Int. Joint Conf. on Artificial Intelligence IJCAI-93*, pages 704–709, 1993.
9. D. Calvanese, M. Lenzerini, D. Nardi. A unified framework for class-based representation languages. In *Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning KR-94*, 1994.
10. M. J. Fisher and R. E. Ladner. Propositional Dynamic Logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.
11. G. De Giacomo and M. Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with mu-calculus. In *Proc. of the 11th Eur. Conf. on Artificial Intelligence ECAI-94*, pages 355–360, 1994.
12. F. M. Donini, B. Hollunder, M. Lenzerini, A. Marchetti Spaccamela, D. Nardi, and W. Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 2–3:309–327, 1992.
13. F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proc. of the 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning KR-91*, pages 151–162, 1991.
14. F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence IJCAI-91*, pages 458–463, 1991.
15. F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt, and A. Schaerf. Adding epistemic operators to concept languages. In *Proc. of the 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning KR-92*, 18–3:201–260, 1987.
16. R. Hull and R. King. Semantic database modeling: Survey, applications, and research Issues. *ACM Computing Surveys*, pages 342–353, 1992.
17. D. Kozen and J. Tiuryn. Logics of programs. In *Handbook of Theoretical Computer Science – Formal Models and Semantics*, pages 789–840. Elsevier, 1990.
18. H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.
19. R. MacGregor. Inside the LOOM description classifier. *SIGART Bulletin*, 2(3):88–92, 1991.

20. B. Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34(3):371–383, 1988.
21. B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.
22. B. Nebel. *Reasoning and revision in hybrid representation systems.* Springer-Verlag, 1990.
23. B. Nebel. Terminological cycles: Semantics and computational properties. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, 1991.
24. R. Parikh. Propositional dynamic logics of programs: a survey. In *Proc. of 1st Workshop on Logic of Programs*, Lecture Notes in Computer Science 125, pages 102–144. Springer-Verlag, 1981.
25. P. F. Patel-Schneider. A hybrid, decidable, logic-based knowledge representation system. *Computational Intelligence*, 3(2):64–77, 1987.
26. K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence IJCAI-91*, pages 466–471, 1991.
27. K. Schild. Terminological cycles and the propositional μ-calculus. In *Proc. of the 4th Int. Conf. on Knowledge Representation and Reasoning KR-94*, 1994.
28. M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
29. G. Schreiber, B. Wielinga, J. Breuker. *KADS: A principled approach to knowledge-based system development.* Academic Press, 1993.
30. M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.