Boosting the correspondence between description logics and propositional dynamic logics

Giuseppe De Giacomo and **Maurizio Lenzerini**
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italia
{degiacom,lenzerini}@assi.dis.uniroma1.it

Abstract

One of the main themes in the area of Terminological Reasoning has been to identify description logics (DLs) that are both very expressive and decidable. A recent paper by Schild showed that this issue can be profitably addressed by relying on a correspondence between DLs and propositional dynamic logics (PDL). However Schild left open three important problems, related to the translation into PDLs of functional restrictions on roles (both direct and inverse), number restrictions, and assertions on individuals. The work reported in this paper presents a solution to these problems. The results have a twofold significance. From the standpoint of DLs, we derive decidability and complexity results for some of the most expressive logics appeared in the literature, and from the standpoint of PDLs, we derive a general methodology for the representation of several forms of program determinism and for the specification of partial computations.

Introduction

The research in Artificial Intelligence and Computer Science has always paid special attention to formalisms for the structured representation of information. In Artificial Intelligence, the investigation of 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 represent knowledge in terms of objects (individuals) grouped into classes (concepts), and offer structuring mechanisms for both characterizing the relevant properties of classes in terms of relations (roles), and establishing several interdependencies among classes (e.g. is-a).

Two main advantages in using structured formalisms for knowledge representation were advocated, namely, epistemological adequacy, and computational effectiveness. In the last decade, many efforts have been devoted to an analysis of these two aspects. In particular, starting with (Brachman & Levesque 1984), the research on the computational complexity of the reasoning tasks associated with DLs has shown that in order to ensure decidability and/or efficiency of reasoning in all cases, one must renounce to some of the expressive power (Levesque & Brachman 1987, Nebel 1988, Nebel 1990a, Donini et al. 1991a, Donini et al. 1991b, Donini et al. 1992). These results have led to a debate on the trade-off between expressive power of representation formalisms and worst-case efficiency of the associated reasoning tasks. This issue has been one of the main themes in the area of DLs, and has led to at least four different approaches to the design of knowledge representation systems.

- In the first approach, the main goal of a DL is to offer powerful mechanisms for structuring knowledge, as well as sound and complete reasoning procedures, while little attention has to be paid to the (worst-case) computational complexity of the reasoning procedures. Systems like OMEGA (Attardi & Simi 1981), LOOM (MacGregor 1991) and KL-ONE (Brachman & Schmolze 1985), can be considered as following this approach.
- The second approach advocates a careful design of the DLs so as to offer as much expressive power as possible while retaining the possibility of sound, complete, and efficient (often polynomial in the worst case) inference procedures. Much of the research on CLASSIC (Brachman et al. 1991) follows this approach.
- The third approach, similarly to the first one, advocates very expressive languages, but, in order to achieve efficiency, accepts incomplete reasoning procedures. No general consensus exists on what kind of incompleteness is acceptable. Perhaps, the most interesting attempts are those resorting to a non-standard semantics for characterizing the form of incompleteness (Patel-Schneider 1987, Borgida & Patel-Schneider 1993, Donini et al. 1992).

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

- Finally, the fourth approach is based on what we can call "the expressiveness and decidability thesis", and aims at defining DLs that are both very expressive and decidable, i.e. designed in such a way that sound, complete, and terminating procedures exist for the associated reasoning tasks. Great attention is given in this approach to the complexity analysis for the various sublogics, so as to devise suitable optimization techniques and to single out tractable subcases. This approach is the one followed in the design of KRIS (Baader & Hollunder 1991).

The work presented in this paper adheres to the fourth approach, and aims at both identifying the most expressive DLs with decidable associated decision problems, and characterizing the computational complexity of reasoning in powerful DLs. In order to clearly describe this approach, let us point out that by "very expressive DL" we mean:

1. The logic offers powerful constructs in order to form concept and role descriptions. Besides the constructs corresponding to the usual boolean connectives (union, intersection, complement), and existential and universal quantification on roles, three important types of construct must be mentioned, namely, those for building complex role descriptions, those for expressing functional restrictions (i.e. that a role is functional for a given concept), and those for expressing number restrictions (a generalization of functional restrictions stating the minimum and the maximum number of links between instances of classes and instances of roles).
2. Besides the possibility of building sophisticated class descriptions, the logic provides suitable mechanisms for stating necessary and/or sufficient conditions for the objects to belong to the extensions of the classes. The basic mechanism for this feature is the so-called inclusion assertion, stating that every instance of a class is also an instance of another class. Much of the work done in DLs assumes that all the knowledge on classes is expressed through the use of class descriptions, and rules out the possibility of using this kind of assertions (note the power of assertions vanishes with the usual assumption of acyclicity of class definitions).
3. The logic allows one to assert properties of single individuals, in terms of the so-called membership assertions. Two membership assertions are taken into account, one for stating that an object is an instance of a given class, and another one for stating that two objects are related to by means of a given role.

Note that, among the constructs for role description, the one for inverse of roles has a special importance, in particular because it makes DLs powerful enough to subsume most frame-based representation systems, semantic data models and object-oriented database models proposed in the literature. Also, functional restrictions on atomic roles and their inverse are essential for real world modeling, specially because the combined use of functional restrictions and inverse of atomic roles allows n-ary relations to be correctly represented.

Two main approaches have been developed following the "expressiveness and decidability thesis". The first approach relies on the tableau-based technique proposed in (Schmidt-Schauß & Smolka 1991, Donini et al. 1991a), and led to the identification of a decision procedure for a logic which fully covers points (2) and (3) above, and only partially point (1) in that it does not include the construct for inverse roles (Buchheit, Donini, & Schaerf 1993). The second approach is based on the work by Schild, 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 correspond to programs. The correspondence is extremely useful for at least 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, Kozen & Tiuryn 1990) can be exploited in the setting of DLs, and, on the converse, the various works on tractability/intractability of DLs (see, for example, Donini et al. 1991b) can be used in the setting of PDL.

However, in order to fully exploit this correspondence, we need to solve at least three problems left open in (Schild 1991), concerning how to fit functional restrictions (on both atomic roles and their inverse), number restrictions, and assertions on individuals, respectively, into the correspondence. Note that these problems refer to points (1) and (3) above.

In this paper we present a solution to each of the three problems, for several very expressive DLs. 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, number restrictions and assertions on individuals, special "constraints" are added to the PDL formulae. The results have a twofold significance. From the standpoint of DLs, we derive decidability and complexity results for some of the most expressive languages appeared in the literature (the only language which is not subsumed by ours is the one studied in (Buchheit, Donini, & Schaerf 1993), whose expressive power is incomparable with respect to the DLs studied here), and from the standpoint of PDLs, we derive a general methodology for the representation of several forms of program determinism corresponding to functional[2] and number restrictions, and for the

[2]Note that no decidability results were known for a PDL

specification of partial computations (assertions on individuals).

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 atomic roles and their inverse. In Section 4, we outline the generalization to the case of number restrictions, and in Section 5 we deal with the problem of representing assertions on individuals. In particular, we analyze two languages and show that reasoning in knowledge bases consisting on both assertions on classes and assertions on individuals in these two languages can be again reduced to satisfiability checking of particular PDL formulae. Finally, in Section 6, we present examples of modeling with the powerful and decidable DLs introduced in the paper, and outline possible extensions of our work. For the sake of brevity all proofs are omitted.

Preliminaries

We base our work on two logics, namely the DL $\mathcal{C}$, and the PDL $\mathcal{D}$, whose basic characteristics are recalled in this section.

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

$$\begin{array}{rcl} C & \longrightarrow & \top \mid \bot \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \\ & & C_1 \Rightarrow C_2 \mid \exists R.C \mid \forall R.C \\ R & \longrightarrow & P \mid R_1 \sqcup R_2 \mid R_1 \circ R_2 \mid R^* \mid id(C) \end{array}$$

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 $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$ interprets concepts as subsets of $\Delta^{\mathcal{I}}$ and roles as binary relations over $\Delta^{\mathcal{I}}$, in such a way that the meaning of the constructs is preserved (for example, $(C_1 \Rightarrow C_2)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid d \notin C_1^{\mathcal{I}} \text{ or } d \in C_2^{\mathcal{I}}\}$, where $C^{\mathcal{I}}$ denotes the set of elements of $\Delta^{\mathcal{I}}$ assigned to C by $\mathcal{I}$). Note that $\mathcal{C}$ is a very expressive language, comprising the constructs for union of roles $R_1 \sqcup R_2$, chaining of roles $R_1 \circ R_2$, transitive closure of roles R^*, and the identity role $id(C)$ projected on C.

A $\mathcal{C}$-intensional knowledge base ($\mathcal{C}$-TBox) 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 $\mathcal{C}$-concepts. The 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 $\mathcal{K}$.

As pointed out in (Schild 1991), there is a direct correspondence between $\mathcal{C}$ and a PDL, here called $\mathcal{D}$, whose syntax is as follows:

$$\begin{array}{rcl} \phi & \longrightarrow & true \mid false \mid A \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \\ & & \phi_1 \Rightarrow \phi_2 \mid < r > \phi \mid [r]\phi \\ r & \longrightarrow & P \mid r_1 \cup r_2 \mid r_1 ; r_2 \mid r^* \mid \phi? \end{array}$$

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 $\mathcal{D}$ is based on the notion of 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 $\mathcal{C}$ and $\mathcal{D}$ is realized through a mapping δ from $\mathcal{C}$-concepts to $\mathcal{D}$-formulae, and from $\mathcal{C}$-roles to $\mathcal{D}$-programs. The mapping δ maps the constructs of $\mathcal{C}$ in the obvious way. For example:

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

In the rest of this section, we introduce several notions and notations that will be used in the sequel. Some of them are concerned with extensions of $\mathcal{D}$ that include the construct r^-, denoting the converse of a program r (see Section 3).

The *Fisher-Ladner closure* of a $\mathcal{D}$-formula Φ, denoted $CL(\Phi)$, is the least set F such that $\Phi \in F$ and such that (we assume $\vee, \Rightarrow, [\cdot]$ to be expressed by means of $\neg, \wedge, < \cdot >$ as usual):

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

Note that, the size of $CL(\Phi)$ is linear with respect to the size of Φ. The notion of Fisher-Ladner closure can be easily extended to formulae of other PDLs.

We introduce the notion of path in a structure M, which extends the one of *trajectory* defined in (Ben-Ari, Halpern, & Pnueli 1982) in order to deal with the

where both atomic programs and their converse can be made (locally) deterministic.

converse of an atomic programs. A *path* in a structure M is a sequence $(s_0, \ldots, s_q)$ of states of M, such that $(s_{i-1}, s_i) \in \mathcal{R}_a$ for some $a = P \mid P^-$, where $i = 1, \ldots, q$. 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 the way in):

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

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).

Functional restrictions

In this section, we study an extension of $\mathcal{C}$, called $\mathcal{CIF}$, which is obtained from $\mathcal{C}$ by adding both the role construct R^- and the concept construct $(\leq 1\ a)$, where $a = P \mid P^-$. The meaning of the two constructs in an interpretation $\mathcal{I}$ is as follows:

$$\begin{array}{l} (R^-)^{\mathcal{I}} = \{(d_1, d_2) \mid (d_2, d_1) \in R^{\mathcal{I}}\}, \\ (\leq 1\ a)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \begin{array}{l} \text{there exists } \textit{at most one } d' \\ \text{such that} (d, d') \in a^{\mathcal{I}} \end{array} \}. \end{array}$$

The corresponding PDL will be called $\mathcal{DIF}$, and is obtained from $\mathcal{D}$ by adding the programs of the form r^-, and the formulae of the form $(\leq 1\ a)$, where, again, $a = P \mid P^-$. The meaning of the two constructs in $\mathcal{DIF}$ can be easily derived by the semantics of $\mathcal{CIF}$. Observe that the r^- construct allows one to denote the converse of a program, and the $(\leq 1\ a)$ construct allows the notion of local determinism for both atomic programs and their converse to be represented in PDL. With the latter construct, we can denote states from which the running of an atomic program (symmetrically, 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 certain atomic programs and converse of atomic programs are globally deterministic. Therefore, $\mathcal{DIF}$ subsumes the logic studied in (Vardi & Wolper 1986), called Converse Deterministic PDL, in which atomic programs (but not their converse) are globally deterministic.

From the point of view of DLs, as mentioned in the Introduction, the presence of inverse roles and of functional restrictions on both atomic roles and their inverse, makes $\mathcal{CIF}$ one of the most expressive DLs among those studied in the literature.

The correspondence between $\mathcal{CIF}$ and $\mathcal{DIF}$ is realized through the mapping δ described in Section 2, suitably extended in order to deal with inverse roles and functional restrictions. From δ we easily obtain the mapping δ^+ from $\mathcal{CIF}$-TBoxes to $\mathcal{DIF}$-formulae. In particular, if $\mathcal{K} = \{K_1, \cdots, K_n\}$ is a TBox in $\mathcal{CIF}$, and $P_1, \ldots, P_m$ are all atomic roles appearing in $\mathcal{K}$ then (we abbreviate $(P_1 \cup \cdots \cup P_m \cup P_1^- \cup \cdots \cup P_m^-)^*$ by $\mathbf{u}$, for notational convenience)

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

Observe that $\delta^+(\mathcal{K})$ exploits the power of program constructs (union, converse, and transitive closure) and the "connected model property" 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 $\mathcal{DIF}$-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.

Let $\mathcal{DI}$ be the PDL obtained from $\mathcal{D}$ by adding the r^- construct only. We are going to 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 logical implication for $\mathcal{CIF}$-TBoxes, are EXPTIME-complete too.[3] In what follows, we assume without loss of generality that 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 a $\mathcal{DIF}$-formula Φ as the conjunction of two formulae, $\gamma(\Phi) = \gamma_1(\Phi) \wedge \gamma_2(\Phi)$, where:

- $\gamma_1(\Phi)$ is obtained from the original formula 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.
- $\gamma_2(\Phi) = \gamma_2^1 \wedge \cdots \wedge \gamma_2^q$, with one conjunct γ_2^i of the form (we use the abbreviation $\mathbf{u}$ for $(P_1 \cup \cdots \cup P_m \cup P_1^- \cdots \cup P_m^-)^*$, where $P_1, \ldots, P_m$ are all the atomic roles appearing in Φ):

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

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

[3] 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 with respect to the formulae in $CL(\gamma_1(\Phi))$. We show that this allows us to actually collapse t_1 and t_2 into a single state. Note that the size of $\gamma(\Phi)$ is polynomial with respect to the size of Φ.

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, one can easily derive a model $M^t_{\mathcal{F}}$ of Φ. 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 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 $CL(\Phi)$.[4] 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). 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 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 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 labelling l, proceeding again level by level. If $M, m(root) \models A_{(\leq 1\ a)}$, then for each arc $(root, child_i(root))$ labelled 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 $(x, child_i(x))$ labelled 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 $(x, child_i(x))$ labelled 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 $M^t = (\mathcal{S}^t, \{\mathcal{R}^t_r\}, \Pi^t)$ as follows: $\mathcal{S}^t = \{x \in \mathcal{T} \mid (root, x) \in (\bigcup_P (\mathcal{R}'_P \cup \mathcal{R'}^-_P))^*\}$, $\mathcal{R}^t_P = \mathcal{R}'_P \cap (\mathcal{S}^t \times S^t)$, and $\Pi^t(x) = \Pi(m(x))$ for all $x \in \mathcal{S}^t$. 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, M a model of $\gamma(\Phi)$, and M^t a structure derived from M as specified above. Then, for every formula $\phi \in CL(\gamma_1(\Phi))$ and every $x \in \mathcal{S}^t$, $M^t, x \models \phi$ iff $M, m(x) \models \phi$.*

Once we have obtained 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 derived 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 logical implication for $\mathcal{CIF}$-TBoxes are EXPTIME-complete problems.*

Number restrictions

In this section, we briefly outline a method that allows us to polynomially encode number restrictions into $\mathcal{CIF}$. Let us call $\mathcal{CIN}$ the language obtained from $\mathcal{CIF}$ by adding the constructs $(\geq n\ a)$ and $(\leq n\ a)$ for number restrictions, where n is a non-negative integer, and $a := P \mid P^-$. The meaning of $(\geq n\ a)$ (resp. $(\leq n\ a)$) in an interpretation $\mathcal{I}$ is given by the set of individuals that are related to at least (at most) n instances of a.

Let $\mathcal{K}$ be a $\mathcal{CIN}$-TBox. We, first, introduce for each atomic role P in $\mathcal{K}$ a new primitive concept A_P and two atomic roles F_P and G_P, imposing that each individual in the class A_P is related to exactly one instance of F^-_P and G^-_P. In this way the original P can be represented by means of the role $F_P \circ id(A_P) \circ G^-_P$. Then we replace F_P by $f_P \circ id(A_P) \circ (f'_P \circ id(A_P))^*$ and G_P by $g_P \circ id(A_P) \circ (g'_P \circ id(A_P))^*$, making the

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

atomic roles f_P, f'_P, g_P, g'_P and their inverse, globally functional, and requiring that no individual is linked to others by means of both f_P^- and f'^-_P, or g_P^- and g'^-_P. In this way the concept $(\leq n\ P)$ can be obtained simply by imposing that there are at most n states in the chain $f_P \circ id(A_P) \circ (f'_P \circ id(A_P))^*$, and the concept $(\leq n\ P^-)$ can be obtained by imposing that there are at most n states in the chain $g_P \circ id(A_P) \circ (g'_P \circ id(A_P))^*$. These constraints are easily expressible in $\mathcal{CIF}$. Analogous considerations hold both for $(\geq n\ a)$ and for qualified number restrictions, where a qualified number restriction is a concept of the form $(\leq n\ a.C)$ (resp. $(\geq n\ a.C)$), which is interpreted as the set of individuals that are related to at most (resp. at least) n instances of C by means of a.

Membership assertions

In this section, we study reasoning involving knowledge on single individuals expressed in terms of membership assertions. Given an alphabet $\mathcal{O}$ of symbols for individuals, a membership assertion is of one of the following forms:

$$C(\alpha_1), \quad R(\alpha_1, \alpha_2)$$

where C is a concept, R is a role, and α_1, α_2 belong to $\mathcal{O}$. The semantics of such assertions is stated as follows. An interpretation $\mathcal{I}$ is extended so as to assign to each $\alpha \in \mathcal{O}$ an element $\alpha^\mathcal{I} \in \Delta^\mathcal{I}$ in such a way that different elements are assigned to different symbols in $\mathcal{O}$. Then, $\mathcal{I}$ satisfies $C(\alpha)$ if $\alpha^\mathcal{I} \in C^\mathcal{I}$, and $\mathcal{I}$ satisfies $R(\alpha_1, \alpha_2)$ if $(\alpha_1^\mathcal{I}, \alpha_2^\mathcal{I}) \in R^\mathcal{I}$. An extensional knowledge base (ABox) $\mathcal{M}$ is a finite set of membership assertions, and an interpretation $\mathcal{I}$ is called a model of $\mathcal{M}$ if $\mathcal{I}$ satisfies every assertion in $\mathcal{M}$.

A knowledge base is a pair $\mathcal{B} = (\mathcal{K}, \mathcal{M})$, where $\mathcal{K}$ is a TBox, and $\mathcal{M}$ is a ABox. An interpretation $\mathcal{I}$ is called a model of $\mathcal{B}$ if it is a model of both $\mathcal{K}$ and $\mathcal{M}$. $\mathcal{B}$ is satisfiable if it has a model, and $\mathcal{B}$ logically implies an assertion β ($\mathcal{B} \models \beta$), where β is either an inclusion or a membership assertion, if every model of $\mathcal{B}$ satisfies β. Since logical implication can be reformulated in terms of unsatisfiability (e.g. if $\beta = C(\alpha)$, then $\mathcal{B} \models \beta$ iff $\mathcal{B} \cup \{\neg C(\alpha)\}$ is unsatisfiable), we only need a procedure for checking satisfiability of a knowledge base.

It is worth noting that, from the point of view of PDLs, an ABox is a sort of specification of partial computations, and that no technique is known for integrating such a form of specification with PDLs' formulae.

We study the satisfiability problem for knowledge bases expressed in two extensions of the basic language $\mathcal{C}$. The first extension regards the language $\mathcal{CF}$, obtained from $\mathcal{C}$ by adding the construct $(\leq 1\ P)$. We show that satisfiability of a $\mathcal{CF}$-knowledge base $\mathcal{B}$ can be polynomially reduced to satisfiability of a $\mathcal{DF}$-formula $\varphi(\mathcal{B})$, where $\mathcal{DF}$ is the PDL obtained from $\mathcal{D}$ by including the construct $(\leq 1\ P)$.

We start by defining $\varphi_0(\mathcal{B})$ to be the $\mathcal{DF}$-formula resulting from the conjunction of the following formulae (there is a new letter A_i in $\varphi_0(\mathcal{B})$ for each individual α_i in $\mathcal{B}$): for every individual α_i, $A_i \Rightarrow \wedge_{j \neq i} \neg A_j$; for every membership assertion of the form $C(\alpha_i)$, $A_i \Rightarrow \delta(C)$ (δ is the mapping introduced in Section 2); for every membership assertion of the form $R(\alpha_i, \alpha_j)$, $A_i \Rightarrow < R > A_j$; for every inclusion assertion $C_1 \sqsubseteq C_2$ in $\mathcal{K}$, $\delta(C_1) \Rightarrow \delta(C_2)$.

Let *create* be a new atomic program, and $\mathbf{u}$ an abbreviation for $(P_1 \cup \ldots \cup P_m)^*$, where $P_1, \ldots, P_m$ are all the atomic roles in $\mathcal{B}$. We define the *$\mathcal{DF}$-counterpart* of $\mathcal{B}$ as $\varphi(\mathcal{B}) = \varphi_1(\mathcal{B}) \wedge \varphi_2(\mathcal{B})$, where:

- $\varphi_1(\mathcal{B}) = \varphi_1^1(\mathcal{B}) \wedge \cdots \wedge \varphi_1^n(\mathcal{B}) \wedge [create]([\mathbf{u}]\varphi_0(\mathcal{B}))$, with one $\varphi_1^i(\mathcal{B}) =< create > A_i$ for each individual α_i in $\mathcal{B}$.
- $\varphi_2(\mathcal{B})$ is the conjunction of the following formulae:
  - For all A_i, for all $\phi \in CL([\mathbf{u}]\varphi_0(\mathcal{B}))$:
  $$[create](< \mathbf{u} > (A_i \wedge \phi) \Rightarrow [\mathbf{u}](A_i \Rightarrow \phi)). \quad (1)$$
  - For all A_i, for all $\phi \in CL([\mathbf{u}]\varphi_0(\mathcal{B}))$, for all programs $r \in CL([\mathbf{u}]\varphi_0(\mathcal{B}))$:
  $$\begin{array}{c} [create](<\mathbf{u}> (A_i \wedge < r_{\neg ind} > \phi) \Rightarrow \\ [\mathbf{u}](A_i \Rightarrow < r_{\neg ind} > \phi)), \end{array} \quad (2)$$
  where $r_{\neg ind}$ denotes the program obtained from the program r by chaining the test $(\wedge_{j \neq i} A_j)?$ after each atomic program in r.
  - For all A_i, A_j, for all programs $r' \in Pre(r)$, $r \in CL([\mathbf{u}]\varphi_0(\mathcal{B}))$:
  $$\begin{array}{c} [create](<\mathbf{u}> (A_i \wedge < r'_{\neg ind} > A_j) \Rightarrow \\ [\mathbf{u}](A_i \Rightarrow < r'_{\neg ind} > A_j)), \end{array} \quad (3)$$
  where $Pre(r)$ for a program r, is defined inductively as follows (ε is the empty sequence of programs): $Pre(P) = \{\varepsilon\}$; $Pre(r_1; r_2) = \{r_1; r'_2 \mid r'_2 \in Pre(r_2)\}$; $Pre(r_1 \cup r_2) = Pre(r_1) \cup Pre(r_2)$; $Pre(r^*) = \{r^*; r' \mid r' \in Pre(r)\}$; $Pre(\phi?) = \{\varepsilon\}$.[5]

The role of (1),(2) and (3) is to allow us to collapse all the states where a certain A_i holds, so as to be able to transform them into a single state corresponding to the individual α_i.

In the following we call states t of a model M of $\varphi(\mathcal{B})$, *individual-aliases* of an individual α_i iff $M, t \models A_i$. The formulae (2) and (3) allow us to prove the technical lemma below.

Lemma 5 *Let M be a model of $\varphi(\mathcal{B})$, let t be an individual-alias of α_i, and let $< r > \phi \in CL([\mathbf{u}]\varphi_0(\mathcal{B}))$. If there is a path from t that satisfies $< r > \phi$, containing N individual-aliases $t_1, \ldots, t_N$ of $\alpha_1, \ldots, \alpha_N$ respectively, then from every individual-alias t' of α_i in M, there is a path that satisfies $< r > \phi$, containing N individual-aliases $t'_1, \ldots, t'_N$ for $\alpha_1, \ldots, \alpha_N$ (in the same order as $t_1, \ldots, t_N$).*

[5] Notice that $< \varepsilon > \phi \equiv \phi$ and $[\varepsilon]\phi \equiv \phi$.

Given a model $M = (\mathcal{S}, \{\mathcal{R}_r\}, \Pi)$ of $\varphi(\mathcal{B})$, we can obtain a new model $M' = (\mathcal{S}', \{\mathcal{R}'_r\}, \Phi')$ of $\varphi(\mathcal{B})$ in which there is exactly one individual-alias, for each individual in $\mathcal{B}$. Let $s \in \mathcal{S}$ be such that $M, s \models \varphi(\mathcal{B})$. For every individual α_i, we randomly choose, among its individual-aliases x such that $(s, x) \in \mathcal{R}_{create}$, a distinguished one denoted by s_{α_i}. We define a set of relations $\{\mathcal{R}''_P\} \cup \{\mathcal{R}''_{create}\}$ as follows: $\mathcal{R}''_{create} = \{(s, s_{\alpha_i}) \in \mathcal{R}_{create} \mid \alpha_i \text{ is an individual}\}$, and $\mathcal{R}''_P = (\mathcal{R}_P - \{(x,y) \in \mathcal{R}_P \mid M, y \models A_j \text{ for some } A_j\}) \cup \{(x, s_{\alpha_j}) \mid (x,y) \in \mathcal{R}_P \text{ and } M, y \models A_j \text{ for some } A_j\}$. The structure M' is defined as: $\mathcal{S}' = \{x \in \mathcal{S} \mid (s,x) \in ((\bigcup_P \mathcal{R}''_P) \cup \mathcal{R}''_{create})^*\}$, $\mathcal{R}'_P = \mathcal{R}''_P \cap (\mathcal{S}' \times \mathcal{S}')$ and $\mathcal{R}'_{create} = \mathcal{R}''_{create} \cap (\mathcal{S}' \times \mathcal{S}')$, and $\Pi'(x) = \Pi(x)$, for each state $x \in \mathcal{S}'$ (from $\{\mathcal{R}'_P\}$ and $\mathcal{R}'_{create}$ we get $\{\mathcal{R}'_r\}$ as usual). Observe that the transformation from M to M' *does not change* the number of "out-going edges" for those states of M which are also states of M'. The following two lemmas concern M'.

Lemma 6 *Let M be a model of $\varphi(\mathcal{B})$, and M' a structure derived from M as specified above. Then for every formula $\phi \in CL(\varphi_1(\mathcal{B}))$, for every state x of M': $M, x \models \phi$ iff $M', x \models \phi$.*

Lemma 7 *Let M be a model of $\varphi(\mathcal{B})$ such that $M, s \models \varphi(\mathcal{B})$, and let M' be a structure derived from M as specified above. Then $M', s \models \varphi(\mathcal{B})$.*

We can now state the main theorem on reasoning in $\mathcal{CF}$-knowledge bases.

Theorem 8 *A $\mathcal{CF}$-knowledge base $\mathcal{B}$ is satisfiable iff its $\mathcal{DF}$-counterpart $\varphi(\mathcal{B})$ is satisfiable.*

Corollary 9 *Satisfiability and logical implication for $\mathcal{CF}$-knowledge bases (TBox and ABox) are EXPTIME-complete problems.*

The second extension regards the language $\mathcal{CI}$, obtained from $\mathcal{C}$ by adding the construct for inverse of roles. Analogously to the case of $\mathcal{CF}$, satisfiability of a $\mathcal{CI}$-knowledge base $\mathcal{B}$ can be polynomially reduced to satisfiability of a $\mathcal{DI}$-formula $\eta(\mathcal{B})$, where $\mathcal{DI}$ is the PDL obtained from $\mathcal{D}$ by allowing converse programs. Let $\eta_0(\mathcal{B})$ be a $\mathcal{DI}$-formula defined similarly to $\varphi_0(\mathcal{B})$ in the case of $\mathcal{CF}$, *create* a new atomic program, and $\mathbf{u}$ an abbreviation for $(P_1 \cup \ldots \cup P_m \cup P_1^- \cup \ldots \cup P_m^-)^*$, where $P_1, \ldots, P_m$ are all the atomic roles in $\mathcal{B}$. We define the *$\mathcal{DI}$-counterpart* of $\mathcal{B}$ as $\eta(\mathcal{B}) = \eta_1(\mathcal{B}) \wedge \eta_2(\mathcal{B})$, where:

- $\eta_1(\mathcal{B}) = \eta_1^1(\mathcal{B}) \wedge \cdots \wedge \eta_1^n(\mathcal{B}) \wedge [create]([\mathbf{u}]\eta_0(\mathcal{B}))$, with each $\eta_1^i(\mathcal{B}) = < create > A_i$ for each individual α_i in $\mathcal{B}$.
- $\eta_2(\mathcal{B}) = \eta_2^1(\mathcal{B}) \wedge \cdots \wedge \eta_2^p(\mathcal{B})$, where we have one $\eta_2^i(\mathcal{B})$ of the form

$$[create](< \mathbf{u} > (A_i \wedge \phi) \Rightarrow [\mathbf{u}](A_i \Rightarrow \phi)), \qquad (4)$$

for each A_i, and for each $\phi \in CL([\mathbf{u}]\eta_0(\mathcal{B}))$.

Again, the role of (4) is to make all the states where a certain A_i holds, equivalent, so as to be able to collapse them into a single state corresponding to the individual α_i. By reasoning similarly to the case of $\mathcal{CF}$, we derive the result below.[6]

Theorem 10 *A $\mathcal{CI}$-knowledge base $\mathcal{B}$ is satisfiable iff its $\mathcal{DI}$-counterpart $\eta(\mathcal{B})$ is satisfiable.*

Corollary 11 *Satisfiability and logical implication for $\mathcal{CI}$-knowledge bases (TBox and ABox) are EXPTIME-complete problems.*

We remark that, in establishing the satisfiability of $\mathcal{CF}$-knowledge bases, the satisfiability of $\mathcal{CI}$-knowledge bases, and the satisfiability of a $\mathcal{CIF}$ concepts, we resorted to a transformation of their models. Unfortunately the kind of transformation used in the first two cases cannot be composed with the one used in the latter. This results in the impossibility of extending the constructions carried out in this section to $\mathcal{CIF}$-knowledge bases.

Discussion and conclusion

The work by Schild on the correspondence between DLs and PDLs provides an invaluable tool for devising decision procedures for very expressive DLs. In this paper we included into this correspondence, notions such as functional restrictions on both atomic roles and their converse, number restrictions, and assertions on individuals, that typically arise in modeling structured knowledge. We made use of the correspondence to determine decision procedures and establish the decidability and the complexity of some of the most expressive DLs appeared in the literature. It is worth noticing that the PDLs defined in this paper are novel and of interest in their own right.

Space limitations have prevented us to demonstrate the full power of the results presented. We mention here that they form the basis to derive suitable decision procedures both for extensions of $\mathcal{CIF}$ that include n-ary relation and qualified number restrictions, and for knowledge bases (TBox and ABox) based on $\mathcal{CF}$ extended with qualified number restrictions. Moreover, some of these results can also be formulated in the setting of the μ-calculus, that has been used to model in single framework terminological cycles interpreted according to Least and Greatest Fixpoint Semantics (Nebel 1991, Schild 1994, De Giacomo & Lenzerini 1994).

In concluding the paper, we would like to show two salient examples of use of the powerful DLs introduced here. They concern the definition of concepts for the representation of lists, and n-ary trees. Consider the following inductive definition of list: *nil* is a *list*; a *node* that has exactly one successor that is a *list*, is a *list*; nothing else in a *list*. This is equivalent to define a list as a chain (of any finite length) of nodes that terminates with nil. Assuming *node* and *nil* to be concepts

[6] The proof is much simpler in this case, witness the absence of constraints analogous to (2) and (3) above.

of our language, we can denote the concept *list* as (we use $C_1 \doteq C_2$ as a shorthand for $C_1 \sqsubseteq C_2, C_2 \sqsubseteq C_1$):

$$list \doteq \exists(id(node \sqcap (\leq 1\ succ)) \circ succ)^*.nil$$

Similarly we can denote the class of (possibly infinite) n-ary trees as:

$$n\_tree \doteq \forall child^-.\bot \sqcap \forall child^*.(node \sqcap (\leq 1\ child^-) \sqcap (\leq n\ child))$$

which defines a *n_tree* as a node having no father and at most n children, and such that all descendents are nodes having one father and at most n children.

Observe that, in order to fully capture the above concepts, we make use of inverse roles, functional restrictions on both atomic and inverse roles, and number restrictions.

References

Attardi, G., and Simi, M. 1981. Consistency and completeness of omega, a logic for knowledge representation. In Proceedings of the International Joint Conference on Artificial Intelligence, 504–510.

Baader, F., and Hollunder, B. 1991. A terminological knowledge representation system with complete inference algorithm. In Proceedings of the Workshop on Processing Declarative Knowledge, Lecture Notes in Artificial Intelligence, pages 67–86: Springer-Verlag.

Ben-Ari, M.; Halpern, J. Y.; and Pnueli, A. 1982. Deterministic propositional dynamic logic: Finite models, complexity, and completeness. *Journal of Computer and System Sciences*, 25:402–417.

Borgida, A., and Patel-Schneider, P. F. 1993. A semantics and complete algorithm for subsumption in the CLASSIC description logic. Forthcoming.

Brachman, R. J., and Levesque, H. J. 1984. The tractability of subsumption in frame-based description languages. In Proceedings of the Fourth National Conference on Artificial Intelligence, 34–37.

Brachman, R. J.; McGuinness, D. L.; Patel-Schneider, P. F.; Alperin Resnick, L.; and Borgida, A. 1991. Living with CLASSIC: when and how to use a KL-ONE-like language. In John F. Sowa, editor, *Principles of Semantic Networks*, 401–456: Morgan Kaufmann.

Brachman, R. J., and Schmolze J. G. 1985. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216.

Buchheit M.; Donini F. M.; and Schaerf, A. 1993. Decidable reasoning in terminological knowledge representation systems. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, 704–709.

De Giacomo, G., and Lenzerini, M. 1994. Concept language with number restrictions and fixpoints, and its relationship with mu-calculus. In Proceedings of Eleventh European Conference on Artificial Intelligence.

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

Donini, F. M.; Lenzerini, M.; Nardi, D.; and Nutt, W. 1991a. The complexity of concept languages. In Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning, 151–162.

Donini, F. M.; Lenzerini, M.; Nardi, D.; and Nutt, W. 1991b. Tractable concept languages. In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, 458–463.

Donini, F. M.; Lenzerini, M.; Nardi, D.; Nutt, W; and Schaerf, A. 1992. Adding epistemic operators to concept languages. In Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning, 342–353.

Kozen, D., and Tiuryn, J. 1990. Logics of programs. In *Handbook of Theoretical Computer Science – Formal Models and Semantics*, 789–840: Elsevier.

Levesque, H. J., and Brachman, R. J. 1987. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93.

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

Nebel, B. 1988. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34(3):371–383.

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

Nebel, B. 1991. Terminological cycles: Semantics and computational properties. In John F. Sowa, editor, *Principles of Semantic Networks*, 331–361: Morgan Kaufmann.

Patel-Schneider, P. F. 1987. A hybrid, decidable, logic-based knowledge representation system. *Computational Intelligence*, 3(2):64–77.

Schild, K. 1991. A correspondence theory for terminological logics: Preliminary report. In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, 466–471.

Schild, K. 1994. Terminological cycles and the propositional μ-calculus. In Proceedings of the Fourth International Conference on Knowledge Representation and Reasoning.

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

Vardi, M. Y., and Wolper, P. 1986. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32:183–221.