In Proceedings of the **16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)**

Reasoning in Expressive Description Logics with Fixpoints based on Automata on Infinite Trees

Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
lastname@dis.uniroma1.it
http://www.dis.uniroma1.it/∼*lastname*

Abstract

In the last years, the investigation on Description Logics (DLs) has been driven by the goal of applying them in several areas, such as, software engineering, information systems, databases, information integration, and intelligent access to the web. The modeling requirements arising in the above areas have stimulated the need for very rich languages, including fixpoint constructs to represent recursive structures. We study a DL comprising the most general form of fixpoint constructs on concepts, all classical concept forming constructs, plus inverse roles, n-ary relations, qualified number restrictions, and inclusion assertions. We establish the EXPTIME decidability of such logic by presenting a decision procedure based on a reduction to nonemptiness of alternating automata on infinite trees. We observe that this is the first decidability result for a logic combining inverse roles, number restrictions, and general fixpoints.

1 Introduction

Description Logics (DLs) allow one to represent a domain of interest in terms of *concepts* and *roles*, where concepts model classes of individuals, and roles model relationships between classes [Woods and Schmolze, 1992; Donini *et al.*, 1996; Borgida and Patel-Schneider, 1994]. A knowledge base expressed in a DL is constituted by inclusion assertions that state the properties of concepts and roles. Various reasoning tasks can be carried out on a knowledge base. The most fundamental one consists in checking whether a certain assertion is logically implied by a knowledge base. A DL is characterized by three aspects: the language used to form complex concepts and roles, the kind of assertions that are used to express properties of concepts and roles, and the inference mechanisms provided for reasoning on the knowledge bases expressible in the system.

In the last years, the investigation on DLs has been driven by the goal of applying them in several areas, such as planning [Weida and Litman, 1992], action representation [Artale and Franconi, 1994], software engineering [Devanbu and Jones, 1997], information systems [Catarci and Lenzerini, 1993], databases [Borgida, 1995; Bergamaschi and Sartori, 1992; Sheth *et al.*, 1993], information integration [Calvanese *et al.*, 1998c], and intelligent access to the web [Levy *et al.*, 1996; Blanco *et al.*, 1994]. The modeling requirements arising in the above areas have stimulated the need for incorporating increasingly expressive representation mechanisms:

- The goal of capturing the semantics of database models and reasoning about data schemas has stressed the importance of number restrictions, n-ary relations, and cyclic assertions in the knowledge base [Calvanese *et al.*, 1994].
- Information integration systems require inclusion assertions not only on concepts, but also on relations [Ullman, 1997].
- Semi-structured data, used in applications such as digital libraries, internet information systems, etc., require the ability to represent data whose structure is not rigid and strictly typed as in conventional database systems. Models for semi-structured data represent data as graphs with labeled edges, and adopt flexible typing schemes in order to classify data [Buneman, 1997]. A special case of such models is XML [Bray *et al.*, 1998], which is becoming the standard for exchanging data on the web. In general, correctly modeling such typing schemes calls for the use of fixpoints in the representation formalism [Calvanese *et al.*, 1998b].
- UML [Booch *et al.*, 1998] is nowadays the standard language for the analysis phase of software and information system development. CASE tools that perform automated reasoning on UML schemas (for example, to test consistency or redundancy) would be of great interest. Fully capturing UML schemas in DLs requires inverse roles, n-ary relations, number restrictions, and general fixpoints on concepts for modeling recursive structures (both inductive and coinductive), such as lists, trees, streams, etc..

DLs that capture all requirements above except fixpoints are known (see e.g. [Calvanese *et al.*, 1998c]). However, fully capturing fixpoints in DLs has been an open problem for a long time. Fixpoints incorporated directly in the semantics have been first studied in [Nebel, 1991; Baader, 1996] for simple DLs. DLs with regular expressions, which can be seen as a form of fixpoints, have been studied in [Baader, 1991], and exploiting the correspondence with Propositional Dynamic Logics in [Schild, 1991; De Giacomo and Lenzerini, 1994]. In [Calvanese *et al.*, 1995] another form of

fixpoints, capturing well-foundedness, has been considered. While such logics got increasingly expressive, they all include fixpoint of a limited form only. Fixpoints on concepts in their full generality have been investigated in [Schild, 1994; De Giacomo and Lenzerini, 1997] developing a correspondence with modal μ-calculus [Kozen, 1983]. However these logics lack inverse roles (and number restrictions on them) which are essential to deal with n-ary relations.

The work presented in this paper closes the gap between the two lines of research, presenting a logic with general fixpoints on concepts that includes all the constructs mentioned above. Specifically, we consider a DL, called $\mathcal{DLR}_\mu$, that includes:

- a very rich language, comprising all classical concept forming constructs, plus inverse roles, n-ary relations, and the most general form of number restrictions;
- the most general form of inclusion assertions, without any limitations on the presence of cycles;
- the most general form of fixpoint on concepts.

We characterize reasoning in such a DL as EXPTIME-complete[1], by presenting a decision procedure based on reducing inference to nonemptiness of two-way alternating automata on infinite trees [Vardi, 1998]. We observe that this is the first decidability result for a logic combining inverse roles, number restrictions, and general fixpoints.

2 The Description Logic $\mathcal{DLR}_\mu$

Traditionally, description logics (DLs) allow one to represent a domain of interest in terms of concepts and roles, which model classes of individuals and binary relationships between classes, respectively. More recently DLs comprising relations of arbitrary arity have been introduced, e.g., $\mathcal{DLR}$ [Calvanese *et al.*, 1998c]. We present the DL $\mathcal{DLR}_\mu$ which extends $\mathcal{DLR}$ by least and greatest fixpoint constructs.

We make use of the standard first-order notions of scope, bound and free occurrences of variables, closed formulae, etc., treating μ and ν as quantifiers.

Concepts and relations (of arity between 2 and n_{max}) are built according to the following syntax:

$$\begin{array}{rcl} R & ::= & \top_n \mid P \mid (\$i/n\!:\!C) \mid \neg R \mid R_1 \sqcap R_2 \\ C & ::= & \top_1 \mid A \mid X \mid \neg C \mid C_1 \sqcap C_2 \mid \\ & & \exists[\$i]R \mid (\leq k\,[\$i]R) \mid \mu X.C \end{array}$$

where P and A denote *atomic relations* and *atomic concepts* respectively, R and C denote arbitrary *relations* and *concepts*, i denotes components of relations, i.e., an integer between 1 and n_{max}, n denotes the arity of a relation, i.e., an integer between 2 and n_{max}, k denotes a nonnegative integer, $\top_1$ denotes the top concept, $\top_n$, for $n = 2, \ldots, n_{max}$, denotes the top relation of arity n, X denotes a concept variable, and the restriction is made that every free occurrence of X in $\mu X.C$ is in the scope of an even number of negations ($(\leq k\,[\$i]R)$ counts as one negation).

Concepts and relations must be *well-typed*, which means that (i) only relations of the same arity n can be combined to form expressions of type $R_1 \sqcap R_2$ (which inherit the arity n), and (ii) $i \leq n$ whenever i denotes a component of a relation of arity n.

[1] The same computational complexity of reasoning with inclusion assertions in the basic DL $\mathcal{ALC}$.

$$\begin{array}{rl} (\top_n)^{\mathcal{I}}_\rho = \top^{\mathcal{I}}_n \subseteq (\Delta^{\mathcal{I}})^n & \qquad (\neg R)^{\mathcal{I}}_\rho = \top^{\mathcal{I}}_n \setminus R^{\mathcal{I}}_\rho \\ P^{\mathcal{I}}_\rho = P^{\mathcal{I}} \subseteq \top^{\mathcal{I}}_n & \qquad (R_1 \sqcap R_2)^{\mathcal{I}}_\rho = (R_1)^{\mathcal{I}}_\rho \cap (R_2)^{\mathcal{I}}_\rho \\ (\$i/n\!:\!C)^{\mathcal{I}}_\rho = & \{(d_1, \ldots, d_n) \in \top^{\mathcal{I}}_n \mid d_i \in C^{\mathcal{I}}_\rho\} \end{array}$$

$$\begin{array}{rl} (\top_1)^{\mathcal{I}}_\rho = \Delta^{\mathcal{I}} & \qquad X^{\mathcal{I}}_\rho = \rho(X) \subseteq \Delta^{\mathcal{I}} \\ A^{\mathcal{I}}_\rho = A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} & \qquad (\neg C)^{\mathcal{I}}_\rho = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}_\rho \\ (C_1 \sqcap C_2)^{\mathcal{I}}_\rho = & (C_1)^{\mathcal{I}}_\rho \cap (C_2)^{\mathcal{I}}_\rho \\ (\exists[\$i]R)^{\mathcal{I}}_\rho = & \{d \mid \exists(d_1, \ldots, d_n) \in R^{\mathcal{I}}_\rho . d_i = d\} \\ (\leq k\,[\$i]R)^{\mathcal{I}}_\rho = & \{d \mid \#\{(d_1, \ldots, d_n) \in R^{\mathcal{I}}_\rho \mid d_i = d\} \leq k\} \\ (\mu X.C)^{\mathcal{I}}_\rho = & \bigcap\{\mathcal{E} \subseteq \Delta^{\mathcal{I}} \mid C^{\mathcal{I}}_{\rho[X/\mathcal{E}]} \subseteq \mathcal{E}\} \end{array}$$

P, R, R_1, and R_2 have arity n

Figure 1: Semantic rules for $\mathcal{DLR}_\mu$

We make use of the standard abbreviations, including $\nu X.C$ for $\neg\mu X.\neg C[X/\neg X]$, where $C[X/C']$ denotes the concept obtained from C by substituting all free occurrences of X with C'. We use λ to denote either μ or ν.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of an *interpretation domain* $\Delta^{\mathcal{I}}$, and an *interpretation function* $\cdot^{\mathcal{I}}$, which maps every atomic concept to a subset of $\Delta^{\mathcal{I}}$, and every atomic relation of arity n to a subset of $(\Delta^{\mathcal{I}})^n$. The presence of free variables does not allow us to extend $\cdot^{\mathcal{I}}$ directly to every concept and relation. For this reason we introduce valuations. A *valuation* ρ on $\mathcal{I}$ is a mapping from concept variables to subsets of $\Delta^{\mathcal{I}}$. Given a valuation ρ, we denote by $\rho[X/\mathcal{E}]$ the valuation identical to ρ except for $\rho[X/\mathcal{E}](X) = \mathcal{E}$.

Let $\mathcal{I}$ be an interpretation and ρ a valuation on $\mathcal{I}$. We assign meaning to concepts and relations of the logic by associating to $\mathcal{I}$ and ρ an *extension function* $\cdot^{\mathcal{I}}_\rho$, mapping concepts to subsets of $\Delta^{\mathcal{I}}$ and relations of arity n to subsets of $(\Delta^{\mathcal{I}})^n$, as shown in Figure 1. Observe that the semantics assigned to $\nu X.C$ is

$$(\nu X.C)^{\mathcal{I}}_\rho = \bigcup\{\mathcal{E} \subseteq \Delta^{\mathcal{I}} \mid \mathcal{E} \subseteq C^{\mathcal{I}}_{\rho[X/\mathcal{E}]}\}$$

The expression $C^{\mathcal{I}}_{\rho[X/\mathcal{E}]}$ can be seen as an operator from subsets $\mathcal{E}$ of $\Delta^{\mathcal{I}}$ to subsets of $\Delta^{\mathcal{I}}$, and, by the syntactic restriction enforced on variables, such an operator is guaranteed to be monotonic wrt $\subseteq$. The constructs $\mu X.C$ and $\nu X.C$ denote respectively the *least fixpoint* and the *greatest fixpoint* of the operator (see [De Giacomo and Lenzerini, 1997] for a discussion on the use of fixpoints in DLs). The extension of closed concepts and relations is independent of the valuation, and therefore for closed concepts and relations we do not consider the valuation explicitly. A closed concept or relation L is *satisfiable* if there exists an interpretation $\mathcal{I}$ such that $L^{\mathcal{I}} \neq \emptyset$.

A $\mathcal{DLR}_\mu$ *knowledge base* is a finite set of *assertions* of the form $L_1 \sqsubseteq L_2$ where L_1 and L_2 are either two closed concepts of $\mathcal{DLR}_\mu$ or two closed relations of the same arity. We use $L_1 \equiv L_2$ as an abbreviation for the assertions $L_1 \sqsubseteq L_2$ and $L_2 \sqsubseteq L_1$. An interpretation $\mathcal{I}$ *satisfies an assertion* $L_1 \sqsubseteq L_2$, if $L^{\mathcal{I}}_1 \subseteq L^{\mathcal{I}}_2$. $\mathcal{I}$ is a *model* of a knowledge base

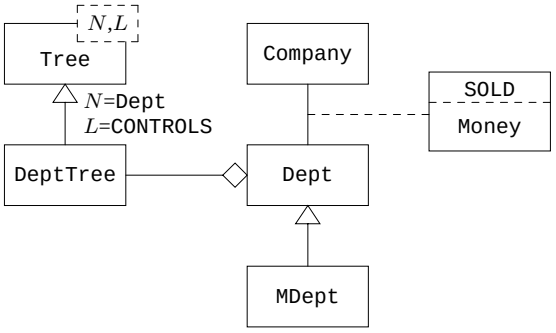

Figure 2: An UML diagram

$\mathcal{K}$, if it satisfies all assertions in $\mathcal{K}$. An assertion $L_1 \sqsubseteq L_2$ is *logically implied* by a knowledge base $\mathcal{K}$ if $L_1^{\mathcal{I}} \subseteq L_2^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{K}$.

Example 2.1 Figure 2 shows an UML diagram which is part of a Telecom Italia application monitoring departments. Departments other than Main Departments are controlled by other departments, in a purely hierarchical fashion (see the use of the concept `Tree`). Moreover, Departments can be sold to companies for a certain amount of money. There are further constraints in the application (not shown in the diagram): First, if a Main Department is sold, then all Departments directly or indirectly controlled by it are also sold. Second, if a Department is sold, then its controlling Main Department is also sold.

We provide the formalization in $\mathcal{DLR}_\mu$ of the UML diagram in Figure 2. $\texttt{Tree}[N, L]$ represents a concept parameterized on N and L, to be used as a template, according to the following inductive definition of tree: (i) an empty tree is a tree; (ii) a node with at most one predecessor, at least one successor, and such that all successors are trees, is a tree; (iii) nothing else is a tree. $\texttt{Tree}[\texttt{Dept}, \texttt{CONTROLS}]$ represents the concept obtained by syntactically substituting `Dept` and `CONTROLS` for the parameters N and L in $\texttt{Tree}[N, L]$.

$$\begin{aligned}
\texttt{Tree}[N,L] &\stackrel{def}{=} \mu X.(\texttt{EmptyTree} \sqcup \\
&\qquad (N \sqcap (\leq 1\,[\$2]L) \sqcap \exists[\$1]L \sqcap \\
&\qquad \neg\exists[\$1](L \sqcap (\$2/2{:}\,\neg X))))\\
\texttt{DeptTree} &\equiv \texttt{Tree}[\texttt{Dept}, \texttt{CONTROLS}]\\
\texttt{SOLD} &\sqsubseteq (\$1{:}\,\texttt{Dept}) \sqcap (\$2{:}\,\texttt{Company}) \sqcap (\$3{:}\,\texttt{Money})\\
\texttt{CONTROLS} &\sqsubseteq (\$1{:}\,\texttt{Dept}) \sqcap (\$2{:}\,\texttt{Dept})\\
\texttt{MDept} &\sqsubseteq \texttt{Dept} \sqcap \neg\exists[\$2]\texttt{CONTROLS}
\end{aligned}$$

The additional constraints mentioned above are formalized as follows:

$$\begin{aligned}
\texttt{MDept} \sqcap \exists[\$1]\texttt{SOLD} &\sqsubseteq \nu X.(\exists[\$1]\texttt{SOLD} \sqcap \\
&\qquad \neg\exists[\$1](\texttt{CONTROLS} \sqcap (\$2{:}\,\neg X)))\\
\texttt{Dept} \sqcap \exists[\$1]\texttt{SOLD} &\sqsubseteq \mu X.((\texttt{MDept} \sqcap \exists[\$1]\texttt{SOLD}) \sqcup \\
&\qquad \exists[\$2](\texttt{CONTROLS} \sqcap (\$1{:}\,X)))
\end{aligned}$$

3 The DLs $\mu\mathcal{ALCQI}$ and $\mu\mathcal{ALCI}_f$

Below we also consider the DL $\mu\mathcal{ALCQI}$, which extends $\mu\mathcal{ALCQ}$, studied in [De Giacomo and Lenzerini, 1997], by the inverse operator on roles. Concepts in $\mu\mathcal{ALCQI}$ are built as follows (R is an atomic or inverse atomic role):

$$C ::= A \mid X \mid \neg C \mid C_1 \sqcap C_2 \mid \exists R.C \mid (\leq k\, R.C) \mid \mu X.C$$

$$\begin{aligned}
\alpha(\top_n) &= A_{\top_n} & \alpha(\top_1) &= A_{\top_1}\\
\alpha(P) &= A_P & \alpha(A) &= A\\
\alpha((i/n{:}\,C)) &= A_{\top_n} \sqcap \forall F_i.\alpha(C) & \alpha(X) &= X\\
\alpha(\neg R) &= A_{\top_n} \sqcap \neg\alpha(R) & \alpha(\neg C) &= A_{\top_1} \sqcap \neg\alpha(C)\\
\alpha(R_1 \sqcap R_2) &= \alpha(R_1) \sqcap \alpha(R_2) & \alpha(C_1 \sqcap C_2) &= \alpha(C_1) \sqcap \alpha(C_2)
\end{aligned}$$

$$\begin{aligned}
\alpha(\exists[\$i]R) &= \exists F_i^-.\alpha(R)\\
\alpha((\leq k\,[\$i]R)) &= (\leq k\, F_i^-.\alpha(R))\\
\alpha(\mu X.C) &= \mu X.\alpha(C)\\
\alpha(L_1 \sqsubseteq L_2) &= \alpha(L_1) \sqsubseteq \alpha(L_2)
\end{aligned}$$

Figure 3: Mapping $\alpha(\cdot)$ from $\mathcal{DLR}_\mu$ to $\mu\mathcal{ALCQI}$

$\mu\mathcal{ALCQI}$ can be viewed as a syntactic variant of *modal μ-calculus* [Kozen, 1983] extended both with *graded modalities* (see e.g., [Van der Hoek and De Rijke, 1995]) and with *backward modalities* [Vardi, 1985].

We observe that $\mu\mathcal{ALCQI}$ can also be considered as a sublanguage of $\mathcal{DLR}_\mu$ by restricting relations to be binary and allowing their use only according to the following abbreviations:

$$\begin{aligned}
\exists P.C \quad &\text{for} \quad \exists[\$1](P \sqcap (\$2/2{:}\,C))\\
\exists P^-.C \quad &\text{for} \quad \exists[\$2](P \sqcap (\$1/2{:}\,C))\\
(\leq k\, P.C) \quad &\text{for} \quad (\leq k\,[\$1](P \sqcap (\$2/2{:}\,C)))\\
(\leq k\, P^-.C) \quad &\text{for} \quad (\leq k\,[\$2](P \sqcap (\$1/2{:}\,C)))
\end{aligned}$$

Finally, we call $\mu\mathcal{ALCI}_f$ the restriction of $\mu\mathcal{ALCQI}$ obtained by forcing all atomic and *inverse* roles to be functional.

4 Encoding $\mathcal{DLR}_\mu$ into $\mu\mathcal{ALCI}_f$

Next we turn to reasoning in $\mathcal{DLR}_\mu$. In particular, we present a technique to decide logical implication in $\mathcal{DLR}_\mu$. In this section we show how to encode $\mathcal{DLR}_\mu$ into $\mu\mathcal{ALCQI}$ and then into $\mu\mathcal{ALCI}_f$. In Section 5 we study reasoning in $\mu\mathcal{ALCI}_f$ by adopting automata theoretic techniques.

Since we can define an atomic relation to be equivalent to any complex relation, we assume wlog that all qualified number restrictions are of the form $(\leq k\,[\$i]P)$, where P is an atomic relation. We also use the standard abbreviations.

To reduce logical implication in $\mathcal{DLR}_\mu$ to logical implication in $\mu\mathcal{ALCQI}$ we extend the technique in [Calvanese *et al.*, 1998a]. We make use of the mapping $\alpha(\cdot)$ defined in Figure 3, and define the $\mu\mathcal{ALCQI}$ knowledge base $\alpha(\mathcal{K})$ by applying α to all assertions in $\mathcal{K}$ and adding:

$$\begin{aligned}
\top &\sqsubseteq A_{\top_1} \sqcup \cdots \sqcup A_{\top_{n_{max}}}\\
\top &\sqsubseteq (\leq 1\, F_i.\top) \quad \text{for each } i \in \{1, \ldots, n_{max}\}\\
\forall F_i.\bot &\sqsubseteq \forall F_{i+1}.\bot \quad \text{for each } i \in \{1, \ldots, n_{max}\}\\
A_{\top_n} &\equiv \exists F_1.A_{\top_1} \sqcap \cdots \sqcap \exists F_n.A_{\top_1} \sqcap \forall F_{n+1}.\bot\\
&\qquad \text{for each } n \in \{2, \ldots, n_{max}\}\\
A_P &\sqsubseteq A_{\top_n} \quad \text{for each atomic relation } P \text{ of arity } n\\
A &\sqsubseteq A_{\top_1} \quad \text{for each atomic concept } A
\end{aligned}$$

Intuitively, $\alpha(\mathcal{K})$ makes use of *reification* of n-ary relations, i.e. a tuple in a model of $\mathcal{K}$ is represented in a model of $\alpha(\mathcal{K})$ by an individual having one functional role F_i for each tuple component $\$i$.

Although atomic roles in $\alpha(\mathcal{K})$ are functional their inverses are not. Next we further transform $\alpha(\mathcal{K})$ to get a $\mu\mathcal{ALCI}_f$

$$
\begin{aligned}
\beta(\top) &= \top & \beta(\neg C) &= \neg\beta(C)\\
\beta(A) &= A & \beta(C_1 \sqcap C_2) &= \beta(C_1) \sqcap \beta(C_2)\\
\beta(X) &= X & \beta(\mu X.C) &= \mu X.\beta(C)\\
\beta(\exists F_i^-.C) &= \exists f_i.\exists g_i^*.\beta(C)\\
\beta(\exists F_i.C) &= \exists f_i^-.\exists (g_i^-)^*.\beta(C)\\
\beta((\leq 1\, F_i.\top)) &= \top\\
\beta((\leq k\, F_i^-.A)) &= \forall f_i.\forall g_i^*.(\neg\beta(A) \sqcup \forall g_i^+.\neg\beta(A) \sqcup\\
&\qquad \forall g_i^+.(\cdots(\neg\beta(A) \sqcup \forall g_i^+.\neg\beta(A))\cdots))\\
\beta(C_1 \sqsubseteq C_2) &= \beta(C_1) \sqsubseteq \beta(C_2)
\end{aligned}
$$

where in the second last equation the number of nested concepts of the form $\neg\beta(A) \sqcup \forall g_i^+.C$ is k, and the following abbreviations are used: $\forall g_i^*.C$ for $\nu X.(C \sqcap \forall g_i.X)$, $\forall g_i^+.C$ for $\forall g_i.\forall g_i^*.C$, $\exists g_i^*.C$ for $\mu X.(C \sqcup \exists g_i.X)$, and $\exists (g_i^-)^*.C$ for $\mu X.(C \sqcup \exists g_i^-.X)$.

Figure 4: Mapping $\beta(\cdot)$ from $\mu\mathcal{ALCQI}$ to $\mu\mathcal{ALCI}_f$

knowledge base $\beta(\alpha(\mathcal{K}))$ (in which also all inverse roles are functional). Intuitively, following [De Giacomo and Lenzerini, 1995], we represent the role F_i^-, $i = 1, \ldots, n_{max}$, by the role $f_i \circ g_i^*$, where f_i, g_i are new functional roles and g_i^* is the reflexive-transitive closure of g_i. Now qualified number restrictions can be encoded as constraints on the chain $f_i \circ g_i^*$. Formally, we make use of the mapping $\beta(\cdot)$ defined in Figure 4.

We define $\beta(\alpha(\mathcal{K}))$ as the $\mu\mathcal{ALCI}_f$ knowledge base obtained by applying β to all assertions in $\alpha(\mathcal{K})$ and adding the assertion $\top \sqsubseteq \neg(\exists f_i^-.\top \sqcap \exists g_i^-.\top)$.

Theorem 4.1 *Given a $\mathcal{DLR}_\mu$ knowledge base $\mathcal{K}$ and a $\mathcal{DLR}_\mu$ assertion $L_1 \sqsubseteq L_2$,*

$$\mathcal{K} \models L_1 \sqsubseteq L_2 \quad \textit{iff} \quad \beta(\alpha(\mathcal{K})) \models \beta(\alpha(L_1 \sqsubseteq L_2)).$$

Since the mappings α and β are polynomial we get the following result.

Theorem 4.2 *Logical implication in $\mathcal{DLR}_\mu$ can be polynomially reduced to logical implication in $\mu\mathcal{ALCI}_f$.*[2]

Finally we observe, that since $\mu\mathcal{ALCI}_f$ has the *connected-model property*, we can internalize assertions and polynomially reduce logical implication to concept satisfiability. Namely, $\mathcal{K} \models C_1 \sqsubseteq C_2$ iff

$$C_1 \sqcap \neg C_2 \sqcap \nu X.(C_{\mathcal{K}} \sqcap (\sqcap_{i=1}^{q}(\forall P_i.X \sqcap \forall P_i^-.X)))$$

is unsatisfiable, where $C_{\mathcal{K}} = \sqcap_{[C \sqsubseteq C' \in \mathcal{K}]}(\neg C \sqcup C')$ and $P_1, \ldots, P_q$ are the atomic roles in $\mathcal{K}$, C_1 and C_2. Therefore, in the following we concentrate on concept satisfiability in $\mu\mathcal{ALCI}_f$.

5 Automata Techniques for $\mu\mathcal{ALCI}_f$

We now study concept satisfiability in $\mu\mathcal{ALCI}_f$ following the techniques based on *two-way alternating automata on infinite trees* (TWAA) introduced in [Vardi, 1998]. Indeed, Vardi used TWAAs to derive a decision procedure for modal μ-calculus with backward modalities. Here we exploit them to derive a reasoning procedure for $\mu\mathcal{ALCI}_f$, which corresponds to a modal μ-calculus with backward modalities in which both forward and backward modalities are functional.

5.1 Automata on Infinite Trees

Infinite trees are represented as prefix closed (infinite) sets of words over $\mathbb{N}$ (the set of positive natural numbers). Formally, an *infinite tree* is a set of words $T \subseteq \mathbb{N}^*$, such that if $x{\cdot}c \in T$, where $x \in \mathbb{N}^*$ and $c \in \mathbb{N}$, then also $x \in T$. The tree is *full* if also $x{\cdot}c' \in T$ for all $0 < c' < c$. The elements of T are called *nodes*, the empty word ε is the *root* of T, and for every $x \in T$, the nodes $x{\cdot}c$, with $c \in \mathbb{N}$, are the *successors* of x. By convention we take $x{\cdot}0 = x$, and $x{\cdot}i{\cdot}{-1} = x$. The *branching degree* $d(x)$ denotes the number of successors of x. If $d(x) = k$ for all nodes x, then we say that the tree is *k-ary*. An *infinite path* P of T is a prefix-closed set $P \subseteq T$ such that for every $i \geq 0$ there exists a unique node $x \in P$ with $|x| = i$. A *labeled tree* over an alphabet Σ is a pair $\langle T, V\rangle$ where T is a tree and $V : T \to \Sigma$.

Alternating automata on infinite trees are a generalization of nondeterministic automata on infinite trees, introduced in [Muller and Schupp, 1987]. They allow for an elegant reduction of decision problems for temporal and program logics [Emerson and Jutla, 1991; Bernholtz *et al.*, 1994]. Let $\mathcal{B}^+(I)$ be the set of positive boolean formulas over I, including also $\mathbf{true}$ and $\mathbf{false}$. For a set $J \subseteq I$ and a formula $\varphi \in \mathcal{B}^+(I)$, we say that J *satisfies* φ iff assigning $\mathbf{true}$ to the elements in J and $\mathbf{false}$ to those in $I \setminus J$ makes φ true. Let $[k] = \{-1, 0, 1, \ldots, k\}$. A *two-way alternating automaton* over infinite k-ary trees is a tuple $\mathbf{A} = \langle \Sigma, Q, \delta, q_0, F\rangle$, where Σ is the input alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \to \mathcal{B}^+([k] \times Q)$ is the transition function, $q_0 \in Q$ is the initial state, and F specifies the acceptance condition.

The transition function maps a state $q \in Q$ and an input letter $\sigma \in \Sigma$ to a positive boolean formula over $[k] \times Q$. Intuitively, if $\delta(q, \sigma) = \varphi$, then each pair (c, q') appearing in φ corresponds to a new copy of the automaton going to the direction suggested by c and starting in state q'. For example, if $k = 2$ and $\delta(q_1, \sigma) = (1, q_2) \wedge (1, q_3) \vee (-1, q_1) \wedge (0, q_3)$, when the automaton is in the state q_1 and is reading the node x labeled by the letter σ, it proceeds either by sending off two copies, in the states q_2 and q_3 respectively, to the first successor of x (i.e., $x{\cdot}1$), or by sending off one copy in the state q_1 to the predecessor of x (i.e., $x{\cdot}{-1}$) and one copy in the state q_3 to x itself (i.e., $x{\cdot}0$).

A run of a TWAA $\mathbf{A}$ over a labeled tree $\langle T, V\rangle$ is a labeled tree $\langle T_r, r\rangle$ in which every node is labeled by an element of $T \times Q$. A node in T_r labeled by $\langle x, q\rangle$ describes a copy of $\mathbf{A}$ that is in the state q and reads the node x of T. The labels of adjacent nodes have to satisfy the transition function of $\mathbf{A}$. Formally, a run $\langle T_r, r\rangle$ is a $T \times Q$-labeled tree satisfying:

1. $\varepsilon \in T_r$ and $r(\varepsilon) = \langle \varepsilon, q_0\rangle$.
2. Let $y \in T_r$, with $r(y) = \langle x, q\rangle$ and $\delta(q, V(x)) = \varphi$. Then there is a (possibly empty) set $S = \{\langle c_1, q_1\rangle, \ldots, \langle c_n, q_n\rangle\} \subseteq [k] \times Q$ such that:
   - S satisfies φ and
   - for all $1 \leq i \leq n$, we have that $y{\cdot}i \in T_r$, $x{\cdot}c_i$ is defined, and $r(y{\cdot}i) = \langle x{\cdot}c_i, q_i\rangle$.

[2] Under the usual assumption that numbers in number restrictions are coded in unary.

A run $\langle T_r, r\rangle$ is *accepting* if all its infinite paths satisfy the acceptance condition. Given an infinite path $P \subseteq T_r$, let $inf(P) \subseteq Q$ be the set of states that appear infinitely often in P (as second components of node labels). We consider here *parity* acceptance conditions. A parity condition over a state set Q is a finite sequence $F = (G_1, \ldots, G_m)$ with $G_1 \subseteq G_2 \subseteq \cdots \subseteq G_m = Q$, and a path P satisfies F if there is an even i for which $inf(P) \cap G_i \neq \emptyset$ and $inf(P) \cap G_{i-1} = \emptyset$.

5.2 Reasoning in $\mu\mathcal{ALCI}_f$

First we observe that $\mu\mathcal{ALCI}_f$ has the *tree model property*, which states that if a $\mu\mathcal{ALCI}_f$ concept C is satisfiable then it is satisfied in an interpretation which has the structure of an infinite tree of bounded degree. In particular, the degree is bounded by $2 \cdot n$, where n is the number of atomic roles appearing in C. The tree model property can be shown following the lines of the proof in [Vardi, 1998] for the modal μ-calculus with backward modalities. Next we define a TWAA that accepts exactly the trees that are models of a concept.

The closure $cl(C)$ of a $\mu\mathcal{ALCI}_f$ concept C (which extends the one in [Kozen, 1983] for the modal μ-calculus) is defined as the smallest set $cl(C)$ of closed concepts that satisfies:

$C \in cl(C)$
$C' \in cl(C)$ implies $\neg C' \in cl(C)$ (we identify $\neg\neg C$ and C)
$C_1 \sqcap C_2, C_1 \sqcup C_2 \in cl(C)$ implies $C_1 \in cl(C)$ and $C_2 \in cl(C)$
$\exists R.C', \forall R.C' \in cl(C)$ implies $C' \in cl(C)$
$\lambda X.C' \in cl(C)$ implies $C'[X/\lambda X.C'] \in cl(C)$

Note that the cardinality of $cl(C)$ is linear in the length of C.

Let C be the $\mu\mathcal{ALCI}_f$ concept we want to check for satisfiability, which wlog we assume to be in negation normal form. Let $\mathcal{A}$ be the set of atomic concepts, and $\mathcal{P} = \{P_1, \ldots, P_n\}$ the set of atomic roles appearing in C. We construct from C a TWAA $\mathbf{A}_C$ which checks that C is satisfied at the root of the input tree. For technical reasons it is useful to consider trees where all nodes have the same branching degree $2n$. To this end we introduce dummy nodes in the tree. We use the symbols A_g and $\neg A_g$ to distinguish nodes that correspond to elements of the model from those that do not. We also represent in the nodes of the tree the information about the labeling of the edges by introducing for each role P_i four symbols A_i, $\neg A_i$, A_i^-, and $\neg A_i^-$. Intuitively, A_i labels $x{\cdot}i$ if $(x, x{\cdot}i) \in P_i^{\mathcal{I}}$ and $\neg A_i$ labels $x{\cdot}i$ if not. Similarly A_i^- labels $x{\cdot}i$ if $(x{\cdot}i, x) \in P_i^{\mathcal{I}}$ and $\neg A_i^-$ labels $x{\cdot}i$ if not.

Since all roles (both direct and inverse) are deterministic, we can assume that for each node x, each P_i and each P_i^- successor appears in a fixed position. In particular, $x{\cdot}i$ is labeled with A_i and $x{\cdot}(i{+}n)$ is labeled with A_i^-. Let det and ini be two new symbols, and $\mathcal{A}_{aux} = \{A_g, \neg A_g\} \cup \bigcup_{i=1}^{n}\{A_i, \neg A_i, A_i^-, \neg A_i^-\} \cup \{det\}$.

The automaton $\mathbf{A}_C = \langle \Sigma, S, \delta, ini, F\rangle$, where $\Sigma = 2^{\mathcal{A}\cup\mathcal{A}_{aux}}$, $S = cl(C) \cup \mathcal{A}_{aux} \cup \{ini, det\}$, the acceptance condition F is as in [Vardi, 1998] and the transition function δ is defined as follows. For all $\sigma \in \Sigma$: for all $A \in \mathcal{A} \cup \mathcal{A}_{aux}$ we have $\delta(A, \sigma) = \mathbf{true}$ if $A \in \sigma$, $\delta(A, \sigma) = \mathbf{false}$ if $A \notin \sigma$, $\delta(\neg A, \sigma) = \mathbf{true}$ if $A \notin \sigma$, $\delta(\neg A, \sigma) = \mathbf{false}$ if $A \in \sigma$, and

$\delta(C_1 \sqcap C_2, \sigma) = (0, C_1) \wedge (0, C_2)$
$\delta(C_1 \sqcup C_2, \sigma) = (0, C_1) \vee (0, C_2)$
$\delta(\lambda X.C_1, \sigma) = (0, C[X/\lambda X.C_1])$
$\delta(\exists P_i.C_1, \sigma) = ((-1, C_1) \wedge (0, A_i^-)) \vee ((i, A_g) \wedge (i, C_1))$
$\delta(\exists P_i^-.C_1, \sigma) = ((-1, C_1) \wedge (0, A_i)) \vee ((i{+}n, A_g) \wedge (i{+}n, C_1))$
$\delta(\forall P_i.C_1, \sigma) = ((-1, C_1) \vee (0, \neg A_i^-)) \wedge ((i, \neg A_g) \vee (i, C_1))$
$\delta(\forall P_i^-.C_1, \sigma) = ((-1, C_1) \vee (0, \neg A_i)) \wedge ((i{+}n, \neg A_g) \vee (i{+}n, C_1)$
$\delta(det, \sigma) = \bigwedge_{i=1}^{2n}(i, det) \wedge \bigwedge_{i=1}^{n}((i, \neg A_g) \vee (i, A_i) \wedge (i{+}n, \neg A_g) \vee (i{+}n, A_i^-)) \wedge \bigwedge_{i=1}^{n}((0, \neg A_i) \vee (n{+}i, \neg A_g) \wedge (0, \neg A_i^-) \vee (i, \neg A_g))$
$\delta(ini, \sigma) = (0, det) \wedge (0, C)$

Intuitively, the automaton starts in the initial state ini and spawns two copies of itself: one verifies that the tree has the right structure wrt functionality, and one checks C on such structure.

Theorem 5.1 *A $\mu\mathcal{ALCI}_f$ concept C is satisfiable iff the set of trees accepted by $\mathbf{A}_C$ in not empty.*

Since nonemptiness of TWAA can be decided in EXPTIME [Vardi, 1998] we get the following upper bound.

Corollary 5.2 *Concept satisfiability in $\mu\mathcal{ALCI}_f$ can be decided in EXPTIME.*

Since the reduction in the previous section is polynomial, we get a worst case deterministic exponential time decision procedure for logical implication in $\mathcal{DLR}_\mu$. Moreover, since logical implication in $\mathcal{DLR}_\mu$ is EXPTIME-hard (it is so already for $\mathcal{ALC}$) we get the following tight complexity bound.

Theorem 5.3 *Logical implication in $\mathcal{DLR}_\mu$ is EXPTIME-complete.*

6 Conclusions

By addressing general fixpoints on concepts, in addition to more standard constructs, DLs finally meet the modeling requirements of advanced applications. The EXPTIME reasoning procedure for $\mathcal{DLR}_\mu$ is the first decidability result for a logic combining inverse roles, number restrictions, and general fixpoints. In particular, since modal μ-calculus extended both with graded and backward modalities corresponds to $\mu\mathcal{ALCQI}$, the result here applies to such logic as well.

We observe that reasoning in the presence of extensional information (ABox) remains an open problem for $\mathcal{DLR}_\mu$.

References

[Artale and Franconi, 1994] A. Artale and E. Franconi. A computational account for a description logic of time and action. In *KR-94*, pages 3–14, 1994.

[Baader, 1991] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *IJCAI-91*, 1991.

[Baader, 1996] F. Baader. Using automata theory for characterizing the semantics of terminological cycles. *Ann. of Math. and AI*, 18:175–219, 1996.

[Bergamaschi and Sartori, 1992] S. Bergamaschi and C. Sartori. On taxonomic reasoning in conceptual design. *ACM TODS*, 17(3):385–422, 1992.

[Bernholtz *et al.*, 1994] O. Bernholtz, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In *CAV-94*, LNCS 818, pages 142–155, 1994.

[Blanco *et al.*, 1994] J. L. Blanco, A. Illarramendi, and A. Goñi. Building a federated relational database system: An approach using a knowledge-based system. *J. of Intelligent and Cooperative Information Systems*, 3(4):415–455, 1994.

[Booch *et al.*, 1998] G. Booch, J. Rumbaugh, and I. Jacobson. *Unified Modeling Language User Guide*. Addison Wesley, 1998.

[Borgida and Patel-Schneider, 1994] A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *JAIR*, 1:277–308, 1994.

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

[Bray *et al.*, 1998] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. *Extensible Markup Language (XML) 1.0 – W3C Recommendation*, 1998.

[Buneman, 1997] P. Buneman. Semistructured data. In *PODS-97*, pages 117–121, 1997.

[Calvanese *et al.*, 1994] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class based representation formalisms. In *KR-94*, pages 109–120, 1994.

[Calvanese *et al.*, 1995] D. Calvanese, G. De Giacomo, and M. Lenzerini. Structured objects: Modeling and reasoning. In *DOOD-95*, LNCS 1013, pages 229–246, 1995.

[Calvanese *et al.*, 1998a] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *PODS-98*, pages 149–158, 1998.

[Calvanese *et al.*, 1998b] D. Calvanese, G. De Giacomo, and M. Lenzerini. What can knowledge representation do for semi-structured data? In *AAAI-98*, pages 205–210, 1998.

[Calvanese *et al.*, 1998c] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *KR-98*, pages 2–13, 1998.

[Catarci and Lenzerini, 1993] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *J. of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.

[De Giacomo and Lenzerini, 1994] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *AAAI-94*, pages 205–212, 1994.

[De Giacomo and Lenzerini, 1995] G. De Giacomo and M. Lenzerini. What's in an aggregate: Foundations for description logics with tuples and sets. In *IJCAI-95*, pages 801–807, 1995.

[De Giacomo and Lenzerini, 1997] G. De Giacomo and M. Lenzerini. A uniform framework for concept definitions in description logics. *JAIR*, 6:87–110, 1997.

[Devanbu and Jones, 1997] P. Devanbu and M. A. Jones. The use of description logics in KBSE systems. *ACM Trans. on Software Engineering and Methodology*, 6(2):141–172, 1997.

[Donini *et al.*, 1996] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In *Principles of Knowledge Representation*, pages 193–238. 1996.

[Emerson and Jutla, 1991] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *FOCS-91*, pages 368–377, 1991.

[Kozen, 1983] D. Kozen. Results on the propositional μ-calculus. *Theor. Comp. Sci.*, 27:333–354, 1983.

[Levy *et al.*, 1996] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Query answering algorithms for information agents. In *AAAI-96*, pages 40–47, 1996.

[Muller and Schupp, 1987] D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *Theor. Comp. Sci.*, 54:267–276, 1987.

[Nebel, 1991] B. Nebel. Terminological cycles: Semantics and computational properties. In *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, 1991.

[Schild, 1991] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *IJCAI-91*, pages 466–471, 1991.

[Schild, 1994] K. Schild. Terminological cycles and the propositional μ-calculus. In *KR-94*, pages 509–520, 1994.

[Sheth *et al.*, 1993] A. P. Sheth, S. K. Gala, and S. B. Navathe. On automatic reasoning for schema integration. *J. of Intelligent and Cooperative Information Systems*, 2(1):23–50, 1993.

[Ullman, 1997] J. D. Ullman. Information integration using logical views. In *ICDT-97*, LNCS 1186, pages 19–40, 1997.

[Van der Hoek and De Rijke, 1995] W. Van der Hoek and M. De Rijke. Counting objects. *J. of Log. and Comp.*, 5(3):325–345, 1995.

[Vardi, 1985] M. Y. Vardi. The taming of converse: Reasoning about two-way computations. In LNCS 193, pages 413–424, 1985.

[Vardi, 1998] M. Y. Vardi. Reasoning about the past with two-way automata. In *ICALP'98*, LNCS 1443, pages 628–641, 1998.

[Weida and Litman, 1992] R. Weida and D. Litman. Terminological reasoning with constraint networks and an application to plan recognition. In *KR-92*, pages 282–293, 1992.

[Woods and Schmolze, 1992] W. A. Woods and J. G. Schmolze. The KL-ONE family. In *Semantic Networks in Artificial Intelligence*, pages 133–178. Pergamon Press, 1992.