Concept language with number restrictions and fixpoints, and its relationship with mu-calculus

Giuseppe De Giacomo[1] and **Maurizio Lenzerini**[2]

Abstract. Many recent works point out that there are several possibilities of assigning a meaning to a concept definition containing some sort of recursion. In this paper, we argue that, instead of choosing a single style of semantics, we achieve a better result by adopting a formalism allowing for different semantics to coexist. In order to demonstrate the feasibility of our proposal, we present a knowledge representation language with the above characteristics. The language is a powerful concept language where, besides the usual constructs for conjunction, disjunction, negation, and quantifiers, both qualified number restrictions, and recursive definitions are allowed. Notably, these features make our formalism one of the most powerful concept languages proposed in literature, in which the usual frame-based descriptions can be combined with definitions of data structures such as lists, directed acyclic graphs, streams, etc. We show that reasoning in our language is decidable, and we characterize its computational complexity by resorting to a correspondence with the modal mu-calculus interpreted over deterministic structures.

1 INTRODUCTION

Many of the modern formalisms developed in Artificial Intelligence for describing an application domain, allow for using the notions of concept (sometime called class or frame) and relationship among concepts. Indeed the notion of concept and that of link among concepts are provided by all structured languages for knowledge representation (frame-based languages, semantic networks, terminological languages, etc.), by the type system of several programming languages, and by the most recent database models, specially those based on the object-oriented paradigm.

There are basically two ways of using and describing classes. In the first one (prescriptive approach) the description formalism allows for expressing several properties of a class, thus prescribing constraints that instances of the class must satisfy. In the second one (definitional approach) the formalism allows for providing the definition of a class, i.e., a set of properties that precisely characterize the instances of the class. While the prescriptive approach is quite well understood and established, the definitional approach is still the subject of an interesting debate, regarding both its nature and its semantic foundation. In particular, it is well known that there are various possibilities of assigning a meaning to a definition of a class, in particular when it contains some sort of recursion ([1, 10, 3, 2]).

In this paper, we are concerned with the semantical problems related to the definitional approach, arguing that, instead of choosing a single style of semantics for the knowledge representation formalism, we achieve a better result by adopting a more general formalism allowing for different semantics to coexist. To demonstrate the feasibility of our proposal, we present a knowledge representation language with the above characteristics, discuss its properties, and describe a method for effectively reasoning with knowledge bases expressed in the language. Our language belongs to the family of concept languages, that have been introduced with the aim of providing a logical reconstruction of frame-based languages. It includes constructs for conjunction, disjunction, negation, quantifiers, qualified number restrictions, and recursive definitions. Notably, these features make our formalism one of the most powerful concept languages proposed in literature, in which the usual frame-based descriptions can be combined with definitions of data structures such as lists, directed acyclic graphs, streams, etc. We show that reasoning in this logic is decidable, and we precisely characterize the computational complexity of the reasoning process by resorting to a correspondence with the modal mu-calculus interpreted over deterministic structures. Several recent papers (see, for example, [4, 5]) advocate the use of concept languages as a unifying framework for different types of database and knowledge representation formalisms. Indeed, it is possible to show that, based on both the constructors and the definitional facilities, our language can capture several database models and programming language type systems. Therefore, the results presented in this paper are not merely confined to concept languages, but are applicable to other representation formalisms.

The paper is organized as follows. In Section 2, we recall some preliminary notions on concept languages and fixpoints. In Section 3, we discuss the different semantics of concept definitions that have been considered in the literature, and we argue for a formalism in which the various semantics coexist. In Section 4, we introduce our proposal, namely the language $\mu\mathcal{ALCN}$. In particular, we present the syntax and the semantics of the language, and we derive decidability and computational complexity of reasoning in the language. We conclude the paper discussing our results and comparing them to other proposals in the literature.

2 PRELIMINARIES

In this section, we briefly recall some preliminary notions regarding concept languages, and fixpoints. The reader is referred to [10] and [6] for an introduction to the subjects.

In this paper, we concentrate our attention on a concept language, called $\mathcal{ALCN}$, in which concepts are composed inductively according to the following abstract syntax:

$$C, D \quad ::= \quad A \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C \mid (\leq n\ R.C) \mid (\geq n\ R.C)$$

[1] Dipartimento di Informatica e Sistemistica, Univ. di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italia. degiacom@assi.dis.uniroma1.it

[2] Dipartimento di Informatica e Sistemistica, Univ. di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italia. lenzerini@assi.dis.uniroma1.it

ECAI 94. *11th European Conference on Artificial Intelligence* Edited by A. Cohn
Published in 1994 by John Wiley & Sons, Ltd.

where A is an atomic concept and R is an atomic 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 $\cdot^{\mathcal{I}}$ mapping every atomic concept to a subset of $\Delta^{\mathcal{I}}$ and every atomic role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ is extended to complex concepts of $\mathcal{ALCN}$ in such a way that the meaning of the construct is preserved (e.g., $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$, $(\exists R.C)^{\mathcal{I}} = \{s \in \Delta^{\mathcal{I}} \mid \exists s'.\ (s, s') \in R^{\mathcal{I}}$ and $s' \in C^{\mathcal{I}}\}$, $(\leq n\, R.C)^{\mathcal{I}} = \{s \in \Delta^{\mathcal{I}} \mid \sharp\{s'.\ (s, s') \in R^{\mathcal{I}}$ and $s' \in C^{\mathcal{I}}\} \leq n\}$). A concept C is satisfiable if there exists an interpretation such that $C^{\mathcal{I}} \neq \emptyset$, unsatisfiable otherwise.

A knowledge base as a finite set (possibly empty) of inclusion statements (or simply inclusion) of the form $C \doteq D$. A pair of inclusions $\{C \doteq D, D \doteq C\}$ is often written as $C \doteq D$ and is called equivalence statement. An interpretation $\mathcal{I}$ satisfies an inclusion $C \doteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of a knowledge base $\mathcal{T}$ if $\mathcal{I}$ satisfies all inclusions in $\mathcal{T}$. A concept C is satisfiable in $\mathcal{T}$, if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$, unsatisfiable otherwise. A concept C is subsumed by a concept D in $\mathcal{T}$, written $\mathcal{T} \models C \sqsubseteq D$, if for every model $\mathcal{I}$ of $\mathcal{T}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

Next we turn to fixpoints. Consider the equation $X = f(X)$ where f is an operator from 2^S to 2^S (2^S denotes the set of all subsets of a set $\mathcal{S}$). The solutions $\mathcal{E}$ of such an equation are called *fixpoints* of the operator f. In general an equation as the one above may have either no solution, a finite number of solutions, or infinite number of them. Among the various solutions, the smallest and the greatest solutions (with respect to set-inclusion) have a prominent position, if they exist. A fundamental result due to Tarski ([18]) guarantees the existence and the uniqueness of both such solutions in case f is monotonic wrt set-inclusion ($\subseteq$), where f is *monotonic* wrt $\subseteq$ whenever $\mathcal{E}_1 \subseteq \mathcal{E}_2$ implies $f(\mathcal{E}_1) \subseteq f(\mathcal{E}_2)$.

Theorem 1 (Tarski) *Let $\mathcal{S}$ be a set, and f an operator from 2^S to 2^S that is monotonic wrt $\subseteq$. Then, there is a unique least fixpoint of f given by $\bigcap\{\mathcal{E} \subseteq \mathcal{S} \mid f(\mathcal{E}) \subseteq \mathcal{E}\}$, and a unique greatest fixpoint of f given by $\bigcup\{\mathcal{E} \subseteq \mathcal{S} \mid \mathcal{E} \subseteq f(\mathcal{E})\}$.*

3 CONCEPT DEFINITIONS AS EQUATIONS

It is widely recognized that the notion of knowledge base as introduced in Section 2 can be made more powerful if we allow some sort of concept definitions to be expressed. Let us call *definition statement* (or simply definition), statements of the form:

$$A =_{def} C$$

where A is an atomic concept and C is a concept expression in $\mathcal{ALCN}$ (A can not occur in the left-hand part of more then one definitions). Intuitively, the above definition statement is intended to provide a precise account of A in terms of C. When we specify the semantics of definitions, we need to distinguish between two different types of atomic concepts, namely, *primitive concepts* and *defined concepts*: given a set $\mathcal{D}$ of definitions, primitive concepts are the atomic concepts that do not appear on the left of any definition of $\mathcal{D}$, whereas defined concepts are those that have an associated definition in $\mathcal{D}$. An interpretation $\mathcal{I}$ satisfies a set of definitions if, for each $A =_{def} C$ in the set, $\mathcal{I}$ assigns the same subset of $\Delta^{\mathcal{I}}$ to the defined concept A and to concept C.

We call *recursive definition statements*[3] (or simply recursive definitions), definition statements of the form

$$A =_{def} F(A),$$

where $F(A)$ stands for a concept that has A as a subconcept[4]. From a semantical point of view, a recursive definition $A =_{def} F(A)$ is a sort of equation specifying that, for any interpretation $\mathcal{I}$, the subset of $\Delta^{\mathcal{I}}$ that can be tied to the concept A must satisfy the equation $A^{\mathcal{I}} = (F(A))^{\mathcal{I}}$, i.e. must be one of its *solutions*. Notice that, in general, either none, one, or several subsets of $\Delta^{\mathcal{I}}$ may exist which are solutions of the above equation. For example, it is easy to see that two interpretations that satisfy the statement $A =_{def} P \sqcap \forall R.A$ and that agree on both the concept P and the role R, may differ in the extension assigned to the defined concept A. Notice also that we can associate to a definition statement an operator from subsets of $\Delta^{\mathcal{I}}$ to subsets of $\Delta^{\mathcal{I}}$, such that the solutions of the equation correspond to the fixpoints of the operator. For example to the definition $A =_{def} P \sqcap \forall R.A$ we can associate, for any interpretation $\mathcal{I}$, the operator $\lambda\mathcal{S}.\{s \in \Delta^{\mathcal{I}} \mid s \in P^{\mathcal{I}}$ and $\forall t.(s, t) \in R^{\mathcal{I}}$ implies $t \in \mathcal{S}\}$.

In the literature on concept languages, three semantics for recursive definitions, have been proposed: the descriptive semantics, the least fixpoint semantics, and the greatest fixpoint semantics ([10]). Let us remind their behavior on some examples. According to the descriptive semantics, a recursive definition $A =_{def} F(A)$ is a *constraint* stating that, for any $\mathcal{I}$ satisfying the definition, $A^{\mathcal{I}}$ has to be *any* solution of the equation $A^{\mathcal{I}} = (F(A))^{\mathcal{I}}$. That is, to $A =_{def} F(A)$ it is assigned the same meaning as to the equivalence statement $A \doteq F(A)$. In our example, $A =_{def} P \sqcap \forall R.A$ states that the individuals in the class A are those in the class P that are related by means of R to individuals in A itself, and vice versa, *where A is no better specified*. In fact the descriptive semantics is not appropriate to properly define recursive concepts. Instead, it is suitable to specify a set of necessary and sufficient conditions that individuals must satisfy in order to be instances of a concept. For example ([10]), we can express the fact that humans are mammals having two parents that are humans, and, on the converse, that mammals having two parents that are humans are humans themselves, in terms of the equivalence statement

$$human \doteq mam \sqcap (\leq 2\, par.\top) \sqcap (\geq 2\, par.\top) \sqcap \forall par.human.$$

It is interesting to observe that we may state an analogous property for horses $horse \doteq mam \sqcap (\leq 2\, par.\top) \sqcap (\geq 2\, par.\top) \sqcap \forall par.horse$ without implying any mutual relationship between $human$ and $horse$. We will see later that this is not true if we use a fixpoint semantics for defining these two concepts.

According to the least (greatest) fixpoint semantics, a definition statement of the form $A =_{def} F(A)$ specifies that, in any interpretation $\mathcal{I}$, A is to be interpreted as the smallest (greatest) solution, if it exists, of $A^{\mathcal{I}} = (F(A))^{\mathcal{I}}$. In other words, in order to consider an interpretation $\mathcal{I}$ adequate to give a meaning to $A =_{def} F(A)$, each other interpretation $\mathcal{J}$, that agrees with $\mathcal{I}$ on the primitive concepts and roles, must assigns to A a superset (subset) of $A^{\mathcal{I}}$. Let us consider some examples, illustrating the differences in the two fixpoint semantics. In our running example $A =_{def} P \sqcap \forall R.A$, the least fixpoint semantics leads to identify A with $\bot$, (indeed the empty set satisfies the statement, and it is obviously the smallest solution), while the greatest fixpoint semantics interprets A as the largest class satisfying the definition, which can be proved to be equivalent to $\forall R^*.P$, where R^* denotes the reflexive and transitive closure of R.

Although the least fixpoint semantics does not help in the above example, it is particularly suitable for providing *inductive definitions* of concepts. Consider the case of a single source finite directed acyclic graph (DAG) defined as follows: an EMPTY-DAG is a DAG (base

[3] Terminological cycles in[10].

[4] A *subconcept* of a concept C is any substring of C (including C itself) that is a concept, according to the syntax rules.

step); a NODE that has connections and all connections are DAGs, is a DAG (inductive step); nothing else is a DAG.[5] We can write a natural definition statement modeling the class of DAGs, namely

$$dag =_{def} emptydag \sqcup (node \sqcap \exists arc.\top \sqcap \forall arc.dag),$$

as long as we interpret it according to the least fixpoint semantics. Similarly, we can model the class of LISTs (defined inductively as: a EMPTY-LIST is a LIST; a NODE that has exactly one successor that is a LIST is a LIST; nothing else is a LIST) by

$$list =_{def} emptylist \sqcup (node \sqcap (\leq 1 succ.\top) \sqcap \exists succ.list).$$

The greatest fixpoint semantics is well suited for defining classes of individuals whose structure is *non-well-founded*. An example is the class of STREAMs, modeling the well-known linear data structure having a NODE as first element and such that the rest of the structure is a STREAM itself. Note that streams, differently from lists, are infinite sequences of nodes. A natural statement for the definition of stream is

$$stream =_{def} node \sqcap (\leq 1\, succ.\top) \sqcap \exists succ.stream$$

with the proviso that, for every $\mathcal{I}$, we need to associate to $stream^{\mathcal{I}}$ the greatest solution of the corresponding equation. Notice however that, if we interpret the definition statements

$$\begin{aligned} human &=_{def} mam \sqcap (\leq 2\, par.\top) \sqcap (\geq 2\, par.\top) \sqcap \forall par.human, \\ horse &=_{def} mam \sqcap (\leq 2\, par.\top) \sqcap (\geq 2\, par.\top) \sqcap \forall par.horse \end{aligned}$$

by the greatest fixpoint semantics, as well as with least fixpoint semantics, we obtain a rather non-intuitive result: for any interpretation $\mathcal{I}$ satisfying the above definition statements, $human^{\mathcal{I}} = horse^{\mathcal{I}}$.

The above considerations show that the three semantics capture different intuitions, and hence we may need all of them in the same knowledge base in order to properly model different concepts. Our proposal in this paper is exactly in the direction of reconciling the various semantics in the same knowledge base. This is pursued by means of a language, called $\mu\mathcal{ALCN}$, that incorporates two constructors, $\mu X.F(X)$ and $\nu X.F(X)$ (the symbols $X, Y, \ldots$ stand for concept variables), denoting, respectively, the least fixpoint and the greatest fixpoint of the operator associated with the definition $X =_{def} F(X)$, that is, for every $\mathcal{I}$ satisfying the definition, the smallest solution and the greatest solution of the equation $X^{\mathcal{I}} = (F(X))^{\mathcal{I}}$.

In our approach, definition statements will never appear in a knowledge base. Instead, a knowledge base will be simply a set of inclusion statements over $\mu\mathcal{ALCN}$ concepts. For example, in order to specify the properties of the concepts of $human$, $horse$, dag, $list$ and $stream$, we can use the equivalence statements:[6]

$$\begin{aligned} dag &\doteq \mu X\,.\, emptydag \sqcup (node \sqcap \exists arc.\top \sqcap \forall arc.X) \\ list &\doteq \mu X\,.\, emptylist \sqcup (node \sqcap (\leq 1\, succ.\top) \sqcap \exists succ.X) \\ stream &\doteq \nu X.node \sqcap (\leq 1\, succ.\top) \sqcap \exists succ.X \\ human &\doteq mam \sqcap (\leq 2\, par.\top) \sqcap (\geq 2\, par.\top) \sqcap \forall par.human \\ horse &\doteq mam \sqcap (\leq 2\, par.\top) \sqcap (\geq 2\, par.\top) \sqcap \forall par.horse. \end{aligned}$$

[5] We assume that a leaf of a DAG is a NODE with all arcs leading to a special node called EMPTY-DAG, as opposed to a NODE having no connection at all. Indeed, in the latter case, the definition of dag would simplify to $dag =_{def} node \sqcap \forall arc.dag$, hiding the general form of inductive definitions (i.e., base case and inductive case).

[6] Notice that, if we add to this knowledge base the equivalence statement $sm \doteq \nu X\,.\, mam \sqcap (\leq 2\, par.\top) \sqcap (\geq 2\, par.\top) \sqcap \forall par.X$, defining the concept sm (sexual mammal), then it turns out that both $human$ and $horse$ are subsumed by sm.

The availability of least and greatest fixpoint constructors not only allows different semantics to be used in the same knowledge base, but also increases the expressive power of concept definitions. On one hand, it makes it possible to model not only abstract classes, but also inductively and co-inductively defined data structures, such as dags, lists and streams. This is particularly important if we aim at effectively integrating class-based representation formalisms and programming systems (declarative or procedural), in order to make these formalisms more useful in practice. On the other hand, we have the possibility of embedding fixpoint constructors within each other, thus going beyond the simple equational format by which we motivated their introduction.[7]

4 ADDING FIXPOINTS TO $\mathcal{ALCN}$

In this section, we describe syntax and semantics of the language $\mu\mathcal{ALCN}$, providing a formal account of the meaning of the fixpoint constructors, and we study the decidability and the computational complexity of reasoning in $\mu\mathcal{ALCN}$ knowledge bases.

We make use of notions of scope, bound and free occurrences of variables, closed formulas, etc. The definitions of these notions are the same as the analogues in first-order logic, treating μ and ν as quantifiers. We also make use of the symbol σ as an abstraction for either μ or ν.

Concepts in $\mu\mathcal{ALCN}$ are formed inductively according to the following abstract syntax:

$$\begin{aligned} C, D \quad ::= \quad & A \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C \mid \\ & (\leq n\, R.C) \mid (\geq n\, R.C) \mid \mu X.C \mid \nu X.C \mid X \end{aligned}$$

where A denotes an atomic concept, R an atomic role, X a variable. We implicitly assume the restriction that every free occurrence of variables X is in the scope of an even number of negation, considering concepts C in $(\leq n\, R.C)$ in the scope of one negation. Not all the constructors introduced are independent, in particular $\nu X.C = \neg\mu X.\neg C[X/\neg X]$ (where $C[X/\neg X]$ is the concept obtained by substituting all free occurrences of X with $\neg X$).

As usual, an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a 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 role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. However, the presence of free variables does not allow us to extend directly the interpretation function $\cdot^{\mathcal{I}}$ to every concept of the language. For this reason we introduce valuations. A *valuation* ρ on an interpretation $\mathcal{I}$, is a mapping from 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 $\rho[X/\mathcal{E}](X) = \mathcal{E}$.

[7] As an example consider the following: Among the inhabitants of the planet "Plonk", a disease called "foo" is quite common. Such a disease manifests in two forms: a "visible" one and a "latent" (not visible) one, and it has a rather intricate hereditary pattern. Individuals that have the visible form transmit the visible form to at least one direct descendant (obviously, if there is any direct descendant), these ill descendants in turn do the same, and so on, *until* someone transmits the latent form of the disease. All direct descendants (if any) of an individual that has the latent form inherit the visible form. The pattern goes on like this, generation after generation, *forever*. Notice that, along any chain of descendants, the visible form of the disease sooner or latter is interrupted, because either an individual has no direct descendant or an individual transmits to some descendant the latent form. The hereditary pattern (foo_hp) of the above disease can be defined as follows:

$$\begin{aligned} foo\_hp \doteq \nu X.\mu Y.((visible \sqcap (\exists child.Y \sqcup \forall child.\bot)) \sqcup \\ (\neg visible \sqcap \forall child.(visible \sqcap X))) \end{aligned}$$

where $visible$ denotes the visible form of the disease, while $\neg visible$ denotes the latent form.

Let $\mathcal{I}$ be an interpretation and ρ a valuation on $\mathcal{I}$. We assign meaning to concepts of the language by associating to $\mathcal{I}$ and ρ an *extension function* $\cdot^{\mathcal{I}}_{\rho}$, mapping concepts to subsets of $\Delta^{\mathcal{I}}$, defined as follows:

$$
\begin{array}{l}
X^{\mathcal{I}}_{\rho} = \rho(X) \subseteq \Delta^{\mathcal{I}}, \quad A^{\mathcal{I}}_{\rho} = A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}, \\
\top^{\mathcal{I}}_{\rho} = \Delta^{\mathcal{I}}, \quad \bot^{\mathcal{I}}_{\rho} = \emptyset, \quad (\neg C)^{\mathcal{I}}_{\rho} = \Delta^{\mathcal{I}} - C^{\mathcal{I}}_{\rho}, \\
(C \sqcap D)^{\mathcal{I}}_{\rho} = C^{\mathcal{I}}_{\rho} \cap D^{\mathcal{I}}_{\rho}, \quad (C \sqcup D)^{\mathcal{I}}_{\rho} = C^{\mathcal{I}}_{\rho} \cup D^{\mathcal{I}}_{\rho}, \\
(\exists R.C)^{\mathcal{I}}_{\rho} = \{s \in \Delta^{\mathcal{I}} \mid \exists s'.\ (s,s') \in R^{\mathcal{I}} \text{ and } s' \in C^{\mathcal{I}}_{\rho}\}, \\
(\forall R.C)^{\mathcal{I}}_{\rho} = \{s \in \Delta^{\mathcal{I}} \mid \forall s'.\ (s,s') \in R^{\mathcal{I}} \text{ implies } s' \in C^{\mathcal{I}}_{\rho}\}, \\
(\leq n\, R.C)^{\mathcal{I}}_{\rho} = \{s \in \Delta^{\mathcal{I}} \mid \sharp\{s'.\ (s,s') \in R^{\mathcal{I}} \text{ and } s' \in C^{\mathcal{I}}_{\rho}\} \leq n\}, \\
(\geq n\, R.C)^{\mathcal{I}}_{\rho} = \{s \in \Delta^{\mathcal{I}} \mid \sharp\{s'.\ (s,s') \in R^{\mathcal{I}} \text{ and } s' \in C^{\mathcal{I}}_{\rho}\} \geq n\}, \\
(\mu X.C)^{\mathcal{I}}_{\rho} = \bigcap\{\mathcal{E} \subseteq \Delta^{\mathcal{I}} \mid\ C^{\mathcal{I}}_{\rho[X/\mathcal{E}]} \subseteq \mathcal{E}\}, \\
(\nu X.C)^{\mathcal{I}}_{\rho} = \bigcup\{\mathcal{E} \subseteq \Delta^{\mathcal{I}} \mid\ \mathcal{E} \subseteq\ C^{\mathcal{I}}_{\rho[X/\mathcal{E}]}\}.
\end{array}
$$

We remark that, in the last two cases $C^{\mathcal{I}}_{\rho[X/\mathcal{E}]}$ is interpreted as an operator from subsets $\mathcal{E}$ of $\Delta^{\mathcal{I}}$ to subsets of $\Delta^{\mathcal{I}}$. By the syntactic restriction enforced on variables, such operator is ensured to be monotonic wrt $\subseteq$. Notice also that free variables appearing in a concept are interpreted as atomic concepts, though by means of the valuation ρ instead of the interpretation $\mathcal{I}$ directly.

A concept C is satisfiable if there exists an interpretation $\mathcal{I}$ and a valuation ρ on $\mathcal{I}$ such that $C^{\mathcal{I}}_{\rho} \neq \emptyset$, unsatisfiable otherwise. $C^{\mathcal{I}}_{\rho} \subseteq D^{\mathcal{I}}_{\rho}$. A $\mu\mathcal{ALCN}$ knowledge base is a finite set (possibly empty) of inclusion statements $C \doteq D$ where C and D are *closed* concepts of $\mu\mathcal{ALCN}$. As before, we use equivalence statements of the form $C \doteq D$ as an abbreviation for $\{C \doteq D, D \doteq C\}$. An interpretation $\mathcal{I}$ satisfies an inclusion statement $C \doteq D$ if $C^{\mathcal{I}}_{\rho} \subseteq D^{\mathcal{I}}_{\rho}$, where ρ is any valuation on $\mathcal{I}$ (being C and D closed, and hence independent from valuations). $\mathcal{I}$ is a model of a knowledge base $\mathcal{T}$ if $\mathcal{I}$ satisfies all inclusion statements in $\mathcal{T}$.[8] We say that a concept C is satisfiable in $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ and a valuation ρ on $\mathcal{I}$ such that $C^{\mathcal{I}}_{\rho} \neq \emptyset$, unsatisfiable otherwise. We say that a concept C is subsumed by a concept D in $\mathcal{T}$, written $\mathcal{T} \models C \sqsubseteq D$, if for every model $\mathcal{I}$ of $\mathcal{T}$ and every valuation ρ on $\mathcal{I}$, $C^{\mathcal{I}}_{\rho} \subseteq D^{\mathcal{I}}_{\rho}$.

We investigate the decidability and the complexity of both subsumption and satisfiability of $\mu\mathcal{ALCN}$ knowledge bases. Actually these two reasoning services are tightly related. Indeed, it is easy to check that a concept C is subsumed by a concept D in a knowledge base $\mathcal{T}$ iff the concept $C \sqcap \neg D$ is unsatisfiable in $\mathcal{T}$, and that a concept C is unsatisfiable in $\mathcal{T}$ iff C is subsumed by $\bot$ in $\mathcal{T}$. Furthermore we can prove the following result.

Theorem 2 *Let $\mathcal{T} = \{C_1 \doteq D_1, \ldots, C_n \doteq D_n\}$ be a $\mu\mathcal{ALCN}$ knowledge base, and C and D two $\mu\mathcal{ALCN}$ concepts. Then C is subsumed by D in $\mathcal{T}$ (i.e., $\mathcal{T} \models C \sqsubseteq \mathcal{D}$) if and only if the $\mu\mathcal{ALCN}$ concept $\nu X.(\forall R_1.X \sqcap \ldots \sqcap \forall R_m.X \sqcap C_{\mathcal{T}}) \sqcap C \sqcap \neg D$ is unsatisfiable, where $R_1, \ldots, R_m$ are all the roles appearing in $\mathcal{T}$, and $C_{\mathcal{T}} = (\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$.*

That is, given a $\mu\mathcal{ALCN}$ knowledge base $\mathcal{T}$, subsumption between concepts in $\mathcal{T}$ (thus also unsatisfiability of a concept in $\mathcal{T}$) is reducible to unsatisfiability of a single $\mu\mathcal{ALCN}$ concept.

In order to devise a (effective) method for checking a $\mu\mathcal{ALCN}$ concept for unsatisfiability, we exhibit a correspondence (similarly to [14, 7]) between $\mu\mathcal{ALCN}$ and a powerful logic of programs called *modal mu-calculus* ([9, 16, 17]), which subsumes most Propositional Dynamic Logics, and most temporal logics for reactive/parallel processes (for references, see [15]).

[8] Notice that inclusions statements in $\mathcal{T}$ are interpreted according to the descriptive semantics.

The abstract syntax of modal mu-calculus is as follows:

$$
\Phi, \Psi \quad ::= \quad A \mid \top \mid \bot \mid \neg\Phi \mid \Phi \wedge \Psi \mid \Phi \vee \Psi \mid < a > \Phi \mid [a]\Phi \mid \mu X.\Phi \mid \nu X.\Phi \mid X
$$

where, A is an atomic formula, a is the generic element of a set of labels $\mathcal{L}$, X a variable, and every occurrence of any variable must be in the scope of an even number of negation. We assign meaning of a formula Φ through a Kripke structure and a valuation. A *Kripke structure* $\mathcal{M}$ is a triple $(\mathcal{S}, \{\mathcal{R}_i \mid i \in \mathcal{L}\}, \mathcal{V})$, where $\mathcal{S}$ is a set of states, each $\mathcal{R}_i$ is a binary relation, and $\mathcal{V}$ is a mapping from atomic formulas to subsets of $\mathcal{S}$. A *valuation* ρ on $\mathcal{M}$ is a mapping from variables to subsets of $\mathcal{S}$. To a structure $\mathcal{M}$ and a valuation ρ on $\mathcal{M}$, we associate an extension function $\cdot^{\mathcal{M}}_{\rho}$ defined inductively as follows: $X^{\mathcal{M}}_{\rho} = \rho(X) \subseteq \mathcal{S}$, $A^{\mathcal{M}}_{\rho} = \mathcal{V}(A) \subseteq \mathcal{S}$, $\top^{\mathcal{M}}_{\rho} = \mathcal{S}$, $\bot^{\mathcal{M}}_{\rho} = \emptyset$, $(\neg\Phi)^{\mathcal{M}}_{\rho} = \mathcal{S} - \Phi^{\mathcal{M}}_{\rho}$, $(\Phi \wedge \Psi)^{\mathcal{M}}_{\rho} = \Phi^{\mathcal{M}}_{\rho} \cap \Psi^{\mathcal{M}}_{\rho}$, $(\Phi \vee \Psi)^{\mathcal{M}}_{\rho} = \Phi^{\mathcal{M}}_{\rho} \cup \Psi^{\mathcal{M}}_{\rho}$, $(< a > \Phi)^{\mathcal{M}}_{\rho} = \{s \in \mathcal{S} \mid \exists s'.(s,s') \in \mathcal{R}_a \text{ and } s' \in \Phi^{\mathcal{M}}_{\rho}\}$ $([a]\Phi)^{\mathcal{M}}_{\rho} = \{s \in \mathcal{S} \mid \forall s'.(s,s') \in \mathcal{R}_a \text{ implies } s' \in \Phi^{\mathcal{M}}_{\rho}\}$, $(\mu X.\Phi)^{\mathcal{M}}_{\rho} = \bigcap\{\mathcal{E} \subseteq \mathcal{S} \mid \Phi^{\mathcal{M}}_{\rho[X/\mathcal{E}]} \subseteq \mathcal{E}\}$, $(\nu X.\Phi)^{\mathcal{M}}_{\rho} = \bigcup\{\mathcal{E} \subseteq \mathcal{S} \mid \mathcal{E} \subseteq \Phi^{\mathcal{M}}_{\rho[X/\mathcal{E}]}\}$. A formula Φ is satisfiable if there exists a Kripke structure $\mathcal{M}$ and a valuation ρ on $\mathcal{M}$ such that $\Phi^{\mathcal{M}}_{\rho} \neq \emptyset$.

In fact, we are interested in a variant of modal mu-calculus, called *deterministic modal mu-calculus*, which has the same syntax as the one above, but is interpreted only on *deterministic Kripke structures*, that is Kripke structures in which the relations $\mathcal{R}_i$ are partial functions ([16]).[9]

We show that there is a function t mapping concepts of $\mu\mathcal{ALCN}$ to deterministic modal mu-calculus formulae, such that C is satisfiable if and only if $t(C)$ is satisfiable. The function t is defined inductively. The mapping form A, X, $C \sqcap D$, $C \sqcup D$, $\neg C$, and $\sigma X.C$ is simply $t(A) = A$, $t(X) = X$, $t(C \sqcap D) = t(C) \wedge t(D)$, $t(C \sqcup D) = t(C) \vee t(D)$, $t(\neg C) = \neg t(C)$, $t(\sigma X.C) = \sigma X.t(C)$. The mapping form $\forall R.C$ and $\exists R.C$ is based on a technique developed in Propositional Dynamic Logic (PDL) to map non-deterministic PDL formulae to deterministic PDL formulae preserving satisfiability ([11, 16]), namely:

$$
\begin{array}{lcl}
t(\exists R.C) & = & < R > (\mu X.(t(C) \vee < R_{new} > X)), \\
t(\forall R.C) & = & [R](\nu X.(t(C) \wedge [R_{new}]X)),
\end{array}
$$

where R_{new} is a new role. Finally, $(\leq n\, R.C)$ and $(\geq n\, R.C)$ are mapped to the following formulae (we use the abbreviation $[R^*]\Phi$ for $\nu X.(\Phi \wedge [R]X)$, and $< R^* > \Phi$ for $\mu X.(\Phi \vee < R > X)$):

$$
\begin{array}{l}
t((\leq n\, R.C)) = [R][R^*_{new}](t(C) \Rightarrow [R^*_{new}](t(C) \Rightarrow \\
\qquad\qquad [R^*_{new}](\ldots(t(C) \Rightarrow [R^*_{new}]\neg t(C))\ldots))
\end{array}
$$

where the number of nested formulae of the form $t(C) \Rightarrow [R^*_{new}]\Phi$ is n, and

$$
\begin{array}{l}
t((\geq n\, R.C)) = < R >< R^*_{new} > (t(C) \wedge < R^*_{new} > (t(C) \wedge \\
\qquad\qquad < R^*_{new} > (\ldots(t(C) \wedge < R^*_{new} > t(C))\ldots))
\end{array}
$$

where the number of nested formulae of the form $t(C) \wedge < R^*_{new} > \Phi$ is $n - 1$. These formulae express constraints on the number of states satisfying C along the chain $R \circ R^*_{new}$. For example, consider the concept $(\leq 2\, R.A)$, where A is an atomic concept, $t((\leq 2\, R.A)) = [R][R^*_{new}](A \Rightarrow [R^*_{new}](A \Rightarrow [R^*_{new}]\neg A))$ that means "everywhere along the chain $R \circ R^*_{new}$ there are at most two states where A holds".

[9] Note that deterministic modal mu-calculus can be considered as an extension of the non-deterministic one, since it holds that every (non-deterministic) modal mu-calculus formula is satisfiable if and only if a polynomially related formula of deterministic modal mu-calculus is satisfiable.

Theorem 3 *Let C be a $\mu\mathcal{ALCN}$ concept, and t the function defined above. Then, C is satisfiable if and only if $t(C)$ is satisfiable.*

It is known that satisfiability in deterministic modal mu-calculus is an EXPTIME-complete problem ([16, 8, 12]). Since $t(C)$ is clearly polynomial in the size of C (assuming numbers in C coded in unary), from the above theorem we can derive the decidability and the computational complexity of reasoning with $\mu\mathcal{ALCN}$ knowledge bases.

Corollary 4 *Let $\mathcal{T}$ be a $\mu\mathcal{ALCN}$ knowledge base. Subsumption between concepts in $\mathcal{T}$ (and hence satisfiability of concepts in $\mathcal{T}$) is decidable in deterministic exponential time (tight bound).*

It is worth noting that in spite of the big increase in expressive power, the computational complexity of reasoning in $\mu\mathcal{ALCN}$ knowledge bases is the same as that of reasoning in $\mathcal{ALC}$ knowledge bases.

5 DISCUSSION

The language introduced in this paper features a novel combination of constructors, namely, number restrictions and fixpoints, that makes it one of the most powerful concept languages proposed in the literature. We already noticed that this combination allows for representing not only abstract classes, but also several data structures extensively used in application programs. We believe that this characteristic is an important step towards a satisfactory integration of concept languages with both traditional and declarative programming systems.

A more detailed analysis of $\mu\mathcal{ALCN}$ reveals that the language indeed provides powerful mechanisms for data structure modeling. Consider a $\mu\mathcal{ALCN}$ knowledge base $\mathcal{T}$ containing the two equivalence statements $list\_of\_student \doteq \mu X\,.\, nil \sqcup (student \sqcap (\leq 1\, succ.\top) \sqcap \exists succ.X)$, and $list\_of\_person \doteq \mu X\,.\, nil \sqcup (person \sqcap (\leq 1\, succ.\top) \sqcap \exists succ.X)$ defining the concepts list of student and list of persons. It is possible to show that, if $\mathcal{T} \models student \sqsubseteq person$ then $\mathcal{T} \models list\_of\_student \sqsubseteq list\_of\_person$. This property can be the base to formulate a notion of *parametric concept* [10] i.e. concepts denoted by expressions containing free variables, by which both generic and polymorphic data structures can be represented. For instance, the expression (named $list\_of[Z]$) $\mu X\,.\, nil \sqcup (Z \sqcap (\leq 1\, succ.\top) \sqcap \exists succ.X)$ where Z is a formal parameter, denotes the class of LISTs whose elements are left unspecified. This class can be used in several ways in the knowledge base. For example, it can be instantiated by binding the formal parameter to actual parameters, thus getting, say, $list\_of[student], list\_of[person], \ldots$, which are concepts inheriting the properties of $list\_of[Z]$.

Our proposal of allowing for fixpoint constructors explicitly in the formalism is shared by a recent work independently carried out by Schild [13]. The main goal of that work is to study both the expressive power and the computational complexity of subsumption and satisfiability for terminological knowledge bases expressed in $\mathcal{ALC}$ (no number restrictions), whose definitions can be *mutually* recursive. To this end, a concept language is defined that corresponds to a variant of the modal mu-calculus ([19]) in which *mutual fixpoints* are allowed but some restrictions on embedding fixpoints are enforced. It is well known that mutual fixpoints can be re-expressed by means of embedded ones (see, for example, [6, 13]). As a consequence of this observation it follows that our language is strictly more expressive than the one analyzed in [13].

[10] Note that, parametric concepts can be introduced also in simpler languages not containing constructs for fixpoints.

We conclude noticing that although the proposed language is very powerful, it lacks a construct for *inverse roles* which is needed for example to correctly capture the notions of TREE, BINARY-TREE, etc. Indeed, to define the concept of TREE (an EMPTY-TREE is a TREE; a NODE that has at most one parent, some child, and all children are TREEs, is a TREE; nothing else is a TREE) we can write $tree \doteq \mu X\,.\, empty\_tree \sqcup (node \sqcap (\leq 1\, child^{-1}.\top) \sqcap \exists child.\top \sqcap \forall child.X)$. Notice that the introduction of inverse roles does not pose any difficulty from the semantical point of view; however, its impact on the reasoning method needs to be investigated.

REFERENCES

[1] F. Baader, 'Terminological cycles in KL-ONE-based KR-languages', in *Proc. of the 8th Nat. Conf. on Artificial Intelligence*, pp. 621–626, (1990).

[2] C. Beeri, 'A formal approach to object-oriented database', in *Data and Knowledge Engineering*, pp. 353–382, (1990).

[3] D. Beneventano and S. Bergamaschi, 'Subsumption for complex object data models', in *Proc. of the 4th Int. Conf. on Database Theory*, LNCS 646, pp. 357–375. Springer-Verlag, (1992).

[4] S. Bergamaschi and C. Sartori, 'On taxonomic reasoning in conceptual design', *ACM Transaction on Database Systems*, **17**(3), 385–422, (1992).

[5] A. Borgida, 'From type systems to knowledge representation: Natural semantics specifications for description logics', *International Journal of Intelligent and Cooperative Information Systems*, **1**(1), 93–126, (1992).

[6] J. de Bakker, *Mathematical Theory of Program Correctness*, Prentice-Hall, 1980.

[7] G. De Giacomo and M. Lenzerini, 'Boosting the correspondence between description logics and propositional dynamic logics', in *Proc. of the 12th Nat. Conf. on Artificial Intelligence*, (1994).

[8] E. A. Emerson and C. S. Jutla, 'The complexity of tree automata and logics of programs', in *Proc. of the 20th Ann. Symp. on the Foundations of Computer Science*, pp. 328–337, (1988).

[9] D. Kozen, 'Results on the propositional mu-calculus', *Theoretical Computer Science*, **27**, 333–355, (1983).

[10] B. Nebel, *Reasoning and Revision in Hybrid Representation Systems*, Lecture Notes In Artificial Intelligence, Springer-Verlag, 1990.

[11] R. Parikh, 'Propositional dynamic logic of programs: A survey', in *Proc. of the 1st Work. on Logic of Programs*, LNCS 125, pp. 102–144. Springer-Verlag, (1981).

[12] S. Safra, 'On the complexity of ω-automata', in *Proc. of the 20th Ann. Symp. on the Foundations of Computer Science*, pp. 319–327, (1988).

[13] K. Schild, 'Terminological cycles and the propositional mu-calculus', in *Proc. of the 4th Int. Conf. on Principles of Knowledge Representation and Reasoning*, (1994).

[14] K. Schild, 'A correspondence theory for terminological logics: Preliminary report', in *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence*, pp. 466–471, (1991).

[15] C. Stirling, 'Modal and temporal logic', in *Handbook of Logic in Computer Science*, 477–563, Clarendon Press, Oxford, (1992).

[16] R. S. Streett and E. A. Emerson, 'The propositional mu-calculus is elementary', in *Proc. of the 6th Int. Col. on Automata, Languages and Programming*, LNCS 172, pp. 465–472. Springer-Verlag, (1984).

[17] R. S. Streett and E. A. Emerson, 'An automata theoretic decision procedure for the propositional mu-calculus', *Information and Control*, **81**, 249–264, (1989).

[18] A. Tarski, 'A lattice-theoretical fixpoint theorem and its applications', *Pacific Journal of Mathematics*, **5**, 285–309, (1955).

[19] M. Y. Vardi and P. Wolper, 'Automata theoretic techniques for modal logics of programs', in *Proc. of the 16th Ann. Symp. on the Foundations of Computer Science*, pp. 446–456, (1984).