Reconciling Different Semantics for Concept Definition (Extended Abstract)

Giuseppe De Giacomo

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italia

e-mail: degiacom@assi.ing.uniroma1.it

1 Introduction

Most of the modern formalisms used in Databases and Artificial Intelligence for describing an application domain allow for using the notions of concept (or class) and relationship among concepts. There are basically two ways of using and describing classes. In the first one (prescriptive approach) the description formalism allows for expressing properties of the classes to be represented, thus prescribing the properties that instances of the classes must possess. 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 for assigning a meaning to a definition of a class, especially if such a definition contains some sort of recursion ([Baa90, Neb91, BB92, Bee90]).

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 powerful formalism allowing different semantics to coexist. Specifically, we present a concept language (an extension of $\mathcal{ALC}$) with the above characteristics, discuss its properties, and describe a method for reasoning effectively with knowledge bases (T-Boxes) expressed in the language. Such a method is based on a correspondence with a particular modal logic of programs called modal mu-calculus ([Koz83, ES88, SA89]), which has been recently investigated for expressing temporal properties of reactive and parallel processes ([Sti92, Lar90]).

2 Concept definitions as equations

It is widely recognized that the notion of knowledge base can be made more powerful if we allow some sort of concept definitions to be expressed. Let us assume that the form of a *definitin statent* (or simply definition) is

$$A =_{def} C$$

where A is an atomic concept and C is a concept expression in $\mathcal{ALC}$. Intuitively, the above definition provides an account for of A in terms of C (i.e. it defines A as the set of objects satisfying C). For example, the definition statement

$$parent =_{def} \exists child.\top$$

provides the *definition* of the concept *parent*, in the following sense: given an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$[1], $parent^{\mathcal{I}}$ denotes a *single* subset of $\Delta^{\mathcal{I}}$, exactly the one denoted by $(\exists child.\top)^{\mathcal{I}}$, i.e., $\{s \mid \exists t.(s,t) \in child^{\mathcal{I}}\}$.

The situation remains clear when we define new concepts using already defined ones. Things change dramatically if we allow an atomic concept to appear both in the left-hand side and the right-hand side of a definition statement. For example, in

$$A =_{def} \exists child.A$$

we cannot appeal any more to the idea of defining new concepts on already defined ones. Actually the word "defining" itself seems misleading in this case. What the above definition statement specifies is that, given any interpretation $\mathcal{I}$, it must be the case that

$$A^{\mathcal{I}} = \{s \in \Delta^{\mathcal{I}} \mid \exists t.(s,t) \in child^{\mathcal{I}} \text{ and } t \in A^{\mathcal{I}}\}.$$

We call the definition statements of the above kind *cyclic definition statements*[2] (or simply cyclic definition), and we use

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

for denoting the generic cyclic definition, where $F(A)$ stands for a concept that contains A as a subconcept[3].

¿From a semantical point of view, a cyclic definition statement $A =_{def} F(A)$ is a sort of equation specifying that, for any interpretation $\mathcal{I}$, the subsets of $\Delta^{\mathcal{I}}$ which can be tied to the concept A must satisfy the equation $A^{\mathcal{I}} = (F(A))^{\mathcal{I}}$, i.e. must be its *solutions*. Notice that, in general, given any interpretation $\mathcal{I}$, either none, one, or several subsets of $\Delta^{\mathcal{I}}$ may exist which are solutions of the above equation. Notice also that, given an interpretation $\mathcal{I}$, 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 $A =_{def} \exists child.A$ we can associate the operator: $\lambda S.\{s \in \Delta^{\mathcal{I}} \mid \exists t.(s,t) \in child^{\mathcal{I}} \text{ and } t \in S\}$ for any interpretation $\mathcal{I}$.

Typically, in the literature on concept languages, there are three ways of interpreting cyclic definitions, according to one of the following semantics:

- Descriptive Semantics,
- Least Fixpoint Semantics,
- Greatest Fixpoint Semantics.

Let us describe the basic ideas underlying the above semantics. According to the *Descriptive Semantics*, a definition statement of the form $A =_{def} F(A)$ is a *constraint* stating that A has to be *some* solution of the corresponding equation. Hence, in our example, $A =_{def} \exists child.A$ states that the individuals in the class A have a child in the class A, and the individuals that have a child in the class A are themselves in the class A, where A is no better specified. Thus, interpreting the definition statement $A =_{def} F(A)$ in terms of the descriptive semantics means that we consider it just as an *equivalence statement*, $A \doteq F(A)$ $(\{A \dot{\leq} F(A),\ F(A) \dot{\leq} A\})$, and therefore it does not represent a real definition.

According to the *Least Fixpoint Semantics*, a definition statement of the form $A =_{def} F(A)$ specifies that A is to be interpreted as the smallest solution (if it exists) of the corresponding equation. Notice that, if the operator associated with the definition statement is monotonic, then for any interpretation $\mathcal{I}$ satisfying the definition statement, the corresponding equation singles

[1] We recall that an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a *domain o nterpretation* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$, mapping concepts to subsets of $\Delta^{\mathcal{I}}$, in particular $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = emptyset$.

[2] Terminological cycles in [Neb91].

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

out a *unique* subset of $\Delta^{\mathcal{I}}$, hence it *defines* the concept C. It is easy to verify that in the example $A =_{def} \exists child.A$, the least fixpoint semantics leads to identify A with $\bot$. Indeed the empty set is a solution of the equation corresponding to the statement, and it is obviously the smallest solution.

By the *Greatest Fixpoint Semantics*, a definition statement of the form $A =_{def} F(A)$ specifies that A is to be interpreted as the greatest solution (if it exists) of the corresponding equation. Again, if the operator associated with the definition statement is monotonic, then for any interpretation $\mathcal{I}$ satisfying the definition statement, the corresponding equation singles out a *unique* subset of $\Delta^{\mathcal{I}}$. Therefore, as for the least fixpoint semantics, the definition statement *defines* the concept A. In the example $A =_{def} \exists child.A$, the greatest fixpoint semantics leads to interpret A as the class of *all* the individuals having a child in A, i.e. the greatest solution of the equation corresponding to the above definition statement.

Which of the three semantics is the best is a long standing matter of debate. It is easy to find examples in which one is adequate and the others are not. Actually, they capture different interesting and important intuitions. In particular, the descriptive semantics is effective when we want to specify constraints on concepts, the least fixpoint semantics is appropriate when we want to define a structure inductively, and, finally, the greatest fixpoint semantics is the one to go for when defining cyclic structures. Therefore, we may need them all in the same knowledge base in order to model the various properties of the different concepts.

3 The language $\mu\mathcal{ALC}$

Our proposal in this work is exactly in the direction of reconciling the various semantics in the same knowledge base. This idea is pursued by means of a language, called $\mu\mathcal{ALC}$, that incorporates special constructors denoting the least fixpoint and the greatest fixpoint of definitions, respectively.

The idea underlying the language $\mu\mathcal{ALC}$ is to add to $\mathcal{ALC}$ the two constructors (the symbols $X, Y, \ldots$ stand for concept variables)

$$\mu X.F(X)$$

$$\nu X.F(X)$$

denoting respectively the smallest solution and the greatest solution of the equation corresponding to the definition $X =_{def} F(X)$. We enforce the restriction that every occurrence of any variable X in $F(X)$ must be in the scope of an even number of $\neg$ (this guarantees that both the smallest and the greatest solutions exist).

No definition statement will actually appear in $\mu\mathcal{ALC}$ knowledge base. Instead, a knowledge base will be simply a set of inclusion statements or equivalence statements over $\mu\mathcal{ALC}$ concepts (that are interpreted according to the descriptive semantics). Let us show some examples of $\mu\mathcal{ALC}$ concepts. First, consider the following inductive definition of a single source directed acyclic graph (DAG):

- The EMPTY-DAG is a DAG (base step).
- A NODE that has connections and all connections are DAGs, is a DAG (inductive step).
- Nothing else is a DAG.

We can easily write a definition statement reflecting the first two conditions:

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

To enforce the third condition we need to take the smallest solution of the corresponding equation (interpreting the above definition statement according to the least fixpoint semantics). In $\mu\mathcal{ALC}$ we can *define* such a concept by a non-cyclic equivalence statement:

$$dag \doteq \mu X \,.\, emptydag \sqcup (node \sqcap \exists arc.\top \sqcap \forall arc.X).$$

Second, suppose we want to denote the class FOB of individuals who are "blond" and generation after generation have some descendant who is "blond". We can write the following definition

$$X =_{def} blond \sqcap \exists child.X.$$

With this definition statement we want to denote the class of *all* individuals satisfying the equation corresponding to the above definition statement, that is, we want its greatest solution (greatest fixpoint semantics). In $\mu\mathcal{ALC}$ we can write:

$$fob \doteq \nu X.blond \sqcap \exists child.X.$$

Third, assume we want to express the fact that humans are mammals having parents that are humans, and, on the converse, that mammals having parents that are humans are humans themselves. We can write the equivalence statement:

$$human \doteq mammal \sqcap \exists parent.\top \sqcap \forall parent.human.$$

This is by no means a definition for *human*, indeed horses satisfy an analogous property:

$$horse \doteq mammal \sqcap \exists parent.\top \sqcap \forall parent.horse.$$

It is interesting to observe that the above two equivalence statements do not imply any mutual relationship between *human* and *horse*. This is not true if we use a fixpoint semantics for interpreting these two concepts.

The idea of adding suitable constructors for the least and the greatest solutions of equations not only allows different semantics to be used in the same knowledge base, but also increases the expressive power of concept definitions.

Consider the following example: 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 quite strange 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), who in turn does 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 rather intricate hereditary pattern (foo_hp) of the above disease can be defined as follows:

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

where *visible* denotes the visible form of the disease, while $\neg visible$ denotes the latent form. This example shows that embedding fixpoint constructors within each other goes beyond the simple equational format by which we motivated their introduction.

In the full paper we formally introduce the language $\mu\mathcal{ALC}$, providing a formal account of the meaning of the fixpoint constructors and we discuss several properties of the language including the complexity of reasoning, by showing a tight correspondence between our language and the modal mu-calculus. With respect to the complexity of reasoning we prove that subsumption between concepts in a $\mu\mathcal{ALC}$ knowledge base $\mathcal{T}$ (and hence satisfiability of concepts in $\mathcal{T}$) is decidable in deterministic exponential time (tight bound). Thus, in spite of the big increase of expressivity, reasoning in $\mu\mathcal{ALC}$ knowledge bases is no harder then reasoning in $\mathcal{ALC}$ knowledge bases.

References

[Baa90] F. Baader. Terminological cycles in KL-ONE-based KR-languages. In *Proc. of the 8th Nat. Conf. on Artificial Intelligence (AAAI-90)*, pages 621–626, 1990.

[BB92] D. Beneventano and S. Bergamaschi. Subsumption for complex object data models. In *Proc. of the 4th Int. Conf. on Database Theory*, LNCS 646, pages 357–375. Springer-Verlag, 1992.

[Bee90] C. Beeri. A formal approach to object-oriented database. In *Data and Knowledge Engeneering*, pages 353–382, 1990.

[ES88] E. A. Emerson and Jutla C. S. The complexity of tree automata and logics of programs. In *Proc. of the 20th Ann. Sym. on the Foundations of Computer Science (FOCS-88)*, pages 328–337, 1988.

[Koz83] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–355, 1983.

[Lar90] K. J. Larsen. Proof systems for satisfiability in Hennessy-Milner logic with recursion. *Theoretical Computer Science*, 72:265–288, 1990.

[Neb91] B. Nebel. Terminological cycles: Semantics and computational properties. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, 1991.

[SA89] R. S. Streett and Emerson E. A. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Control*, 81:249–264, 1989.

[Sti92] C. Stirling. Modal and temporal logic. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, pages 477–563. Clarendon Press, Oxford, 1992.