Appeared in Proc. of **KR'94**

A Unified Framework for Class-Based Representation Formalisms

Diego Calvanese, Maurizio Lenzerini, Daniele Nardi

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
e-mail: {calvanese,lenzerini,nardi}@assi.dis.uniroma1.it

Abstract

The notion of class is ubiquitous in Computer Science and is central in many knowledge representation languages. In this paper we propose a representation formalism in the style of concept languages, with the aim of providing a unified framework for class-based formalisms. The language we consider is quite expressive and features a novel combination of constructs including number restrictions, inverse roles and inclusion assertions with no restrictions on cycles. We are able to show that such language is powerful enough to model frame systems, object-oriented database languages and semantic data models. As a consequence of the established correspondences, several significant extensions of each of the above formalisms become available. The high expressivity of the language and the need for capturing the reasoning in different contexts forces us to distinguish between unrestricted and finite model reasoning. A notable feature of our proposal is that reasoning in both cases is decidable. For the unrestricted case we exploit a correspondence with propositional dynamic logic and extend it to the treatment of number restrictions. For the finite model case we develop a new method based on the use of linear programming techniques. We argue that, by virtue of the high expressive power and of the associated reasoning techniques on both unrestricted and finite models, our language provides a unified framework for class-based representation formalisms.

1 INTRODUCTION

In many fields of Computer Science we find formalisms for the representation of objects and classes [MM92]. Generally speaking a *class* denotes a subset of the domain of discourse, and a class-based representation formalism allows one to express several kinds of relationships and constraints (e.g. subclass constraints) holding among classes. Moreover, class-based formalisms aim at taking advantage of the class structure in order to provide various information, such as whether an element belongs to a class, whether a class is a subclass of another class, and more generally, whether a given constraint holds between two classes.

Three main families of class-based formalisms can be identified. The first one comes from knowledge representation and in particular from the work on semantic networks and frames (see for example [Leh92, Sow91]). The second one originates in the field of databases and in particular from the work on semantic data models (see for example [HK87]). The third one arises from the work on types in programming languages and object-oriented systems (see for example [KL89]).

In the past there have been several attempts to establish relationships among class-based formalisms. In [BHR90] and [LNS91] a comparative analysis and an attempt to provide a unified view of class-based languages are carried out. The analysis makes it clear that several difficulties arise in identifying a common framework for the formalisms developed in different areas. Some recent papers address this problem. For example, an analysis of the relationships between concept languages and types in programming languages has been carried out in [Bor92], while in [BS92, PSS92] concept languages are used to enrich the deductive capabilities of semantic and object-oriented data models.

The proposed solutions are not fully general and a formalism capturing both the modeling constructs and the reasoning techniques for all the above families is still missing. In this paper we provide a solution to this problem by proposing a class-based representation formalism, called $\mathcal{ALUNI}$, which main characteristics are:

1. it is quite expressive and features a novel combination of constructs including number restrictions, inverse roles and inclusion assertions with no re-

strictions on cycles;

2. it is equipped with suitable techniques for both unrestricted and finite model reasoning, since it is designed for capturing the reasoning in different contexts;
3. sound and complete reasoning in both unrestricted and finite models can be done in worst-case exponential time.

The first characteristic allows us to show that $\mathcal{ALUNI}$ is powerful enough to provide a unified framework for frame systems, object-oriented languages and semantic data models. We show this by establishing a precise correspondence with the Entity Relationship model [Che76] model and with an object-oriented language in the style of [AK89]. Moreover, we demonstrate that the formalism proposed in this paper provides important features that are currently missing in each family, although their relevance has often been stressed. In this sense, the work reported here may also contribute to significant developments for the languages belonging to all the three families.

With regard to the second point, the two cases of reasoning in unrestricted and finite models are solved by means of different techniques. For unrestricted satisfiability, we exploit the correspondence with dynamic logic [Sch91], by extending it to the treatment of number restrictions, which have no direct counterpart in dynamic logics. For finite satisfiability we develop a new method based on linear programming techniques by extending the approach proposed in [CL94]. It is worth noting that the problem of finite reasoning, which arises mainly in the field of databases, has never been considered in knowledge representation languages, although it seems quite relevant for practical applications.

As for the third point, the expressive power of $\mathcal{ALUNI}$ makes reasoning hard, but nonetheless decidable. We consider this feature very important, because it makes this language an actual knowledge representation language and not simply a formal framework for comparing apparently different approaches. Obviously, there are a number of sublanguages of $\mathcal{ALUNI}$, where, by giving up some of the expressivity, one gains on the computational complexity. However, this issue is outside the scope of the present paper.

Summarizing, our framework provides an adequate expressive power to account for the most significant features of the major families of class-based formalisms. Moreover, it is equipped with suitable techniques for reasoning in both finite and unrestricted models. Therefore, $\mathcal{ALUNI}$ and the associated reasoning capabilities represent the essential core of the class-based representation formalisms belonging to all three families mentioned above.

The paper is organized as follows. In the next section we present our formalism and discuss its relationships with frame languages, semantic data models and object-oriented languages. Section 3 describes the technique for unrestricted model satisfiability and Section 4 the technique for finite model satisfiability. The final section contains some concluding remarks.

2 A UNIFYING CLASS-BASED REPRESENTATION LANGUAGE

In this section, we present $\mathcal{ALUNI}$, a class-based formalism in the style of concept languages, and show that it can be used to formalize knowledge represented with formalisms developed in different fields.

The basic elements of concept languages are *concepts* and *roles*, which denote classes and binary relations, respectively. In $\mathcal{ALUNI}$, concepts and roles are formed by means of the following syntax (A denotes an atomic concept, P an atomic role, C and D arbitrary concepts, R an arbitrary role and m and n positive integers):

$$\begin{array}{rcl} C, D & \longrightarrow & \top \mid \bot \mid A \mid \neg A \mid C \sqcap D \mid C \sqcup D \mid \\ & & \forall R.C \mid (\geq m\, R) \mid (\leq n\, R)^1 \\ R & \longrightarrow & P \mid P^{-1} \end{array}$$

Concepts are interpreted as subsets of a domain and roles as binary relations over that domain. More precisely, an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ (the *domain* of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ (the *interpretation function* of $\mathcal{I}$) that maps every concept to a subset of $\Delta^{\mathcal{I}}$ and every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that the following equations are satisfied: ($\sharp\{\}$ denotes the cardinality of a set)

$$\begin{array}{rcl} \bot^{\mathcal{I}} & = & \emptyset \\ \top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\ (\neg A)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \\ (\forall R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid \forall b.\, (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\ (\geq m\, R)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid \sharp\{b \mid (a,b) \in R^{\mathcal{I}}\} \geq m\} \\ (\leq n\, R)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid \sharp\{b \mid (a,b) \in R^{\mathcal{I}}\} \leq n\} \\ (R^{-1})^{\mathcal{I}} & = & \{(a,b) \in (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \mid (b,a) \in R^{\mathcal{I}}\} \end{array}$$

In an $\mathcal{ALUNI}$ *knowledge base*, the knowledge about classes and relations is expressed through the use of the so called *inclusion assertions* which have the form

$$A \sqsubseteq C$$

where A is an atomic concept and C an arbitrary concept. An interpretation $\mathcal{I}$ *satisfies* the inclusion assertion $A \sqsubseteq C$ if $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a

[1] We use the shorthand $(=\, n\, R)$ in place of $(\leq n\, R) \sqcap (\geq n\, R)$

model of a knowledge base $\mathcal{K}$ if it satisfies all inclusion assertions in $\mathcal{K}$. A *finite model* is a model with finite domain. Number restrictions, inverse roles and inclusion assertions may interact in such a way that a knowledge base is satisfiable only in infinite models. Therefore, it is meaningful to distinguish between unrestricted and finite satisfiability (implication): $\mathcal{K}$ is said to be *(finitely) satisfiable* if it admits a (finite) model, and it *(finitely) implies* an inclusion assertion $A \sqsubseteq C$ if the inclusion is satisfied in all (finite) models of $\mathcal{K}$.

Below we discuss three families of class-based formalisms, namely, frame languages, semantic data models, and object-oriented data models, and we show that their basic features are captured by knowledge bases in $\mathcal{ALUNI}$.

2.1 FRAME LANGUAGES

Frame languages are based on the idea of expressing knowledge by means of *frames*, which are structures representing classes of objects in terms of the properties that their instances must satisfy. Such properties are defined by the frame *slots*, that constitute the items of a frame definition. In Figure 1 we present an example of a knowledge base defined by frame languages. The notation is basically the one adopted in [FK85], which is used in the KEE[2] system. The corresponding formalization in $\mathcal{ALUNI}$ is given by:

$$
\begin{array}{rcl}
\texttt{Course} & \sqsubseteq & \forall \texttt{ENROLLS.Student} \sqcap \\
 & & (\geq 2\,\texttt{ENROLLS}) \sqcap (\leq 30\,\texttt{ENROLLS}) \sqcap \\
 & & \forall \texttt{TAUGHTBY.}(\texttt{Professor} \sqcup \texttt{Grad}) \sqcap \\
 & & (= 1\,\texttt{TAUGHTBY}) \\
\texttt{AdvCourse} & \sqsubseteq & \texttt{Course} \sqcap (\leq 20\,\texttt{ENROLLS}) \sqcap \\
 & & \forall \texttt{ENROLLS.}(\texttt{Grad} \sqcap \neg \texttt{Undergrad}) \\
\texttt{BasCourse} & \sqsubseteq & \texttt{Course} \sqcap \forall \texttt{TAUGHTBY.}(\texttt{Professor} \sqcap \neg \texttt{Grad}) \\
\texttt{Grad} & \sqsubseteq & \texttt{Student} \sqcap \forall \texttt{DEGREE.String} \sqcap \\
 & & (= 1\,\texttt{DEGREE}) \\
\texttt{Undergrad} & \sqsubseteq & \texttt{Student}
\end{array}
$$

We observe that inverse roles are not used in the formalization. Indeed, the possibility of referring to the inverse of a slot has been rarely considered in frame knowledge representation systems. However, as recent works show (see [DLNN91]), this is a strong limitation in expressivity. For instance, without inverse roles we cannot specify, in our example, that every student is enrolled in at least 4 courses. In fact, KEE, as well as many practical frame systems, embeds other features, such as attachments and overriding inheritance. Such features cannot be captured in our framework, which is intended to deal with the structural and monotonic aspects of these systems.

[2]KEE is a trademark of Intellicorp.

```
Frame: Course in KB University
  Subclasses: AdvCourse, BasCourse
  Memberslot: ENROLLS
    ValueClass: Student
    Cardinality.Min: 2
    Cardinality.Max: 30
  Memberslot: TAUGHTBY
    ValueClass: (UNION Grad Professor)
    Cardinality.Min: 1
    Cardinality.Max: 1

Frame: AdvCourse in KB University
  Superclasses: Course
  Memberslot: ENROLLS
    ValueClass: (INTERSECTION
                 Grad (NOT Undergrad))
    Cardinality.Max: 20

Frame: BasCourse in KB University
  Superclasses: Course
  Memberslot: TAUGHTBY
    ValueClass: (INTERSECTION
                 Professor (NOT Grad))

Frame: Professor in KB University

Frame: Student in KB University
  Subclasses: Grad, Undergrad

Frame: Grad in KB University
  Superclasses: Student
  Memberslot: DEGREE
    ValueClass: String
    Cardinality.Min: 1
    Cardinality.Max: 1

Frame: Undergrad in KB University
  Superclasses: Student
```

Figure 1: A KEE Knowledge Base

In [FK85], several reasoning services associated with frames are mentioned, such as: inheritance, cardinality reasoning and consistency checking. For example, one could ask the system whether the knowledge base implies that the filler of a given slot belongs to a certain class. Due to the absence of inverse roles, it is possible to show that if a frame knowledge base is satisfiable, then it admits a finite model. Therefore, the distinction between reasoning in finite and infinite models is not necessary, and all the above mentioned forms of reasoning are captured by unrestricted satisfiability and implication in $\mathcal{ALUNI}$.

In the last decade, the research on frame languages concentrated on the definition of concept languages, which are subsets of first-order logics, introduced for the formalization of KL-ONE languages (see [WS92]). The only limitation of $\mathcal{ALUNI}$-knowledge bases compared with some of the concept languages appeared in

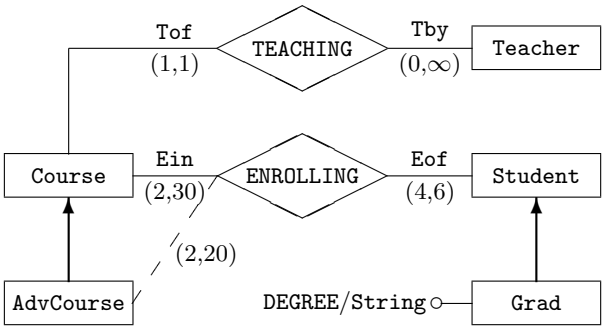

Figure 2: An ER-schema

the literature is that inclusion assertions require the left hand side to be an atomic concept. On the other hand, we do not rule out cyclic references in the inclusion assertions (see [Neb91]), as opposed to most of the approaches to concept languages. Moreover, $\mathcal{ALUNI}$ includes inverse roles, number restrictions and inclusion assertions, which combination has never been addressed in the literature, and whose decidability (both in unrestricted and finite models) was an open problem.

2.2 SEMANTIC DATA MODELS

Semantic data models were introduced primarily as formalisms for database schema design. They provide a means to model databases in a much richer way than traditional data models supported by Database Management Systems, and are becoming more and more important because they are adopted in most of the recent Computer Aided Software Engineering tools.

The most common semantic data model is the Entity-Relationship (ER) model introduced in [Che76]. Figure 2 shows the ER-schema for the same state of affairs represented by the KEE knowledge base in Figure 1. In the ER notation, classes are called *entities* and are represented as boxes, whereas relationships between entities are represented as diamonds. Arrows between entities, called ISA relationships, represent inclusion assertions. The links between entities and relationships represent the ER-*roles*, to which number restrictions are associated. Dashed links are used whenever such restrictions are refined for more specific entities. Finally, elementary properties of entities are modeled by *attributes* (`DEGREE` in Figure 2).

The ER model does not provide constructs for expressing negation and disjunction, although several recent papers stress their importance in database specification [CL93, CHS91]. Referring to our example, the absence of negation and disjunction makes it impossible to specify that courses are taught by either professors or graduate students. For this purpose, the new entity `Teacher` has been introduced as an abstraction of professor and graduate student.

An $\mathcal{ALUNI}$ knowledge base that captures exactly the semantics of the schema of Figure 2 is given by the following set of inclusion assertions:

$$
\begin{aligned}
\texttt{TEACHING} &\sqsubseteq \forall \texttt{Tof.Course} \sqcap (= 1\,\texttt{Tof}) \sqcap \forall \texttt{Tby.Teacher} \sqcap (= 1\,\texttt{Tby})\\
\texttt{ENROLLING} &\sqsubseteq \forall \texttt{Ein.Course} \sqcap (= 1\,\texttt{Ein}) \sqcap \forall \texttt{Eof.Student} \sqcap (= 1\,\texttt{Eof})\\
\texttt{Course} &\sqsubseteq \forall \texttt{Tof}^{-1}\texttt{.TEACHING} \sqcap (= 1\,\texttt{Tof}^{-1}) \sqcap \forall \texttt{Ein}^{-1}\texttt{.ENROLLING} \sqcap (\geq 2\,\texttt{Ein}^{-1}) \sqcap (\leq 30\,\texttt{Ein}^{-1})\\
\texttt{AdvCourse} &\sqsubseteq \texttt{Course} \sqcap (\leq 20\,\texttt{Ein}^{-1})\\
\texttt{Teacher} &\sqsubseteq \forall \texttt{Tby}^{-1}\texttt{.TEACHING}\\
\texttt{Student} &\sqsubseteq \forall \texttt{Eof}^{-1}\texttt{.ENROLLING} \sqcap (\geq 4\,\texttt{Eof}^{-1}) \sqcap (\leq 6\,\texttt{Eof}^{-1})\\
\texttt{Grad} &\sqsubseteq \texttt{Student} \sqcap \forall \texttt{DEGREE.String} \sqcap (= 1\,\texttt{DEGREE})
\end{aligned}
$$

In order to prove that in general $\mathcal{ALUNI}$ is powerful enough to capture all properties of ER-schemata, we first need formal definitions of their syntax and semantics. In the following, for ease of presentation, we do not consider attributes any more. We point out, however, that their inclusion in the specification is straightforward, and that even attributes with a predefined domain of a fixed cardinality do not pose special problems with respect to reasoning on the schema.

The definitions make use of the notion of *labeled tuple* over a generic set $\mathcal{D}$, which is a function from a subset of a set $\mathcal{U}$ of ER-roles to $\mathcal{D}$. The labeled tuple T that maps $U_i \in \mathcal{U}$ to $d_i \in \mathcal{D}$, for $i \in 1..k$, is denoted with $\langle U_1\!:d_1, \ldots, U_k\!:d_k\rangle$. We also write $T[U_i]$ to denote d_i.

Definition 2.1 *An* ER-schema $\mathcal{S}$ *is constituted by:*

- *a set* $\mathcal{E}_\mathcal{S}$ *of* entity *symbols, a set* $\mathcal{R}_\mathcal{S}$ *of* relationship *symbols and a set* $\mathcal{U}_\mathcal{S}$ *of* role *symbols;*
- *a set* $\mathcal{S}_{isa}$ *of statements of the form* $E_1 \preceq E_2$*, where* E_1 *and* E_2 *are entities; the reflexive transitive closure of* $\preceq$ *is denoted with* $\preceq^*$ *;*
- *for each relationship symbol* $R \in \mathcal{R}_\mathcal{S}$*, a labeled tuple over the set of entities*[3]*;*
- *for each relationship* $R\langle U_1\!:E_1, \ldots, U_k\!:E_k\rangle$ *in* S*, for* $i \in 1..k$ *and for each entity* $E \in \mathcal{E}_\mathcal{S}$ *such that* $E \preceq^* E_i$*, a non negative integer,* $minc(E, R, U_i)$*, and a non negative integer or* ∞*,* $maxc(E, R, U_i)$*. If not stated otherwise,* $minc(E, R, U_i)$ *is assumed to be* 0 *and* $maxc(E, R, U_i)$ *is assumed to be* ∞*.*

The semantics of an ER-schema can be given by specifying which database states conform to the informa-

[3] In the following we write $R\langle U_1\!:E_1, \ldots, U_k\!:E_k\rangle$ to denote the relationship R and to specify at the same time that $\langle U_1\!:E_1, \ldots, U_k\!:E_k\rangle$ is the labeled tuple associated to it.

tion structure represented by the schema. Formally, a database state $\mathcal{B}$ is constituted by a nonempty finite set $\Delta^{\mathcal{B}}$ and a function $\cdot^{\mathcal{B}}$ that maps

- every entity $E \in \mathcal{E}_{\mathcal{S}}$ to a subset $E^{\mathcal{B}}$ of $\Delta^{\mathcal{B}}$ and
- every relationship $R \in \mathcal{R}$ to a set $R^{\mathcal{B}}$ of labeled tuples over $\Delta^{\mathcal{B}}$.

The elements of $E^{\mathcal{B}}$ and $R^{\mathcal{B}}$ are called *instances* of E and R respectively.

A database state is considered acceptable if it satisfies all integrity constraints that are part of the schema. This is captured by the definition of legal database state.

Definition 2.2 *A database state $\mathcal{B}$ is said to be* legal with respect to *an ER-schema $\mathcal{S}$, if it satisfies the following conditions:*

- *for each statement $E_1 \preceq E_2 \in \mathcal{S}_{isa}$ it holds that $E_1^{\mathcal{B}} \subseteq E_2^{\mathcal{B}}$;*
- *for each relationship $R\langle U_1\colon E_1, \ldots, U_k\colon E_k\rangle$ in $\mathcal{S}$, all instances of R are of the form $\langle U_1\colon \tilde{e}_1, \ldots, U_k\colon \tilde{e}_k\rangle$, where $\tilde{e}_i \in E_i^{\mathcal{B}}$ for $i \in 1..k$;*
- *for each relationship $R\langle U_1\colon E_1, \ldots, U_k\colon E_k\rangle$ in $\mathcal{S}$, for $i \in 1..k$, for each entity $E \in \mathcal{E}_{\mathcal{S}}$ such that $E \preceq^* E_i$ and for each instance $\tilde{e}$ of E in $\mathcal{I}$, it holds that*

$$minc(E, R, U_i) \leq \sharp\left\{\tilde{r} \in R^{\mathcal{I}} \;\middle|\; \tilde{r}[U_i] = \tilde{e}\right\} \leq$$

$$\leq maxc(E, R, U_i).$$

Notice that the definition of database state reflects the usual assumption in the whole database area that database states are finite structures (see also [CKV90]).

Reasoning in the ER-model includes entity satisfiability and inheritance. Entity satisfiability amounts to checking if a given entity can be populated in some legal database state (see [AP86, LN90]), and corresponds to the notion of concept satisfiability in concept languages. We show that all these forms of reasoning are captured by finite satisfiability and finite implication in $\mathcal{ALUNI}$ knowledge bases. This is done by first defining a mapping Φ from ER-schemata to $\mathcal{ALUNI}$ knowledge bases, and then proving that there is a correspondence between legal database states and finite models of the derived knowledge base.

Definition 2.3 *Let $\mathcal{S}$ be an ER-schema. The $\mathcal{ALUNI}$ knowledge base $\mathcal{K} = \Phi(\mathcal{S})$ is defined as follows:*

- *for each entity $E \in \mathcal{E}_{\mathcal{S}}$, $\mathcal{K}$ contains an atomic concept $\Phi(E)$;*
- *for each statement $E_1 \preceq E_2 \in \mathcal{S}_{isa}$, $\mathcal{K}$ contains an inclusion assertion $\Phi(E_1) \sqsubseteq \Phi(E_2)$;*
- *for each relationship $R\langle U_1\colon E_1, \ldots, U_k\colon E_k\rangle$ in $\mathcal{S}$, $\mathcal{K}$ contains an atomic concept $\Phi(R)$, k primitive roles $U_{R,1}, \ldots, U_{R,k}$ and the following inclusion assertions:*

$$\begin{aligned}\Phi(R) &\sqsubseteq \forall U_{R,1}.\Phi(E_1) \sqcap \cdots \sqcap \forall U_{R,K}.\Phi(E_k) \sqcap \\ &\quad (=1\,U_{R,1}) \sqcap \cdots \sqcap (=1\,U_{R,k}) \\ \Phi(E_i) &\sqsubseteq \forall U_{R,i}^{-1}.\Phi(R), \quad \textit{for } i \in 1..k;\end{aligned}$$

- *for each relationship $R\langle U_1\colon E_1, \ldots, U_k\colon E_k\rangle$ in $\mathcal{S}$, for $i \in 1..k$ and for each entity $E \in \mathcal{E}_{\mathcal{S}}$ such that $E \preceq^* E_i$, if $m = minc(E, R, U_i) \neq 0$, then $\mathcal{K}$ contains the assertion $\Phi(E) \sqsubseteq (\geq m\,U_{R,i}^{-1})$, and if $n = maxc(E, R, U_i) \neq \infty$, then $\mathcal{K}$ contains the assertion $\Phi(E) \sqsubseteq (\leq n\,U_{R,i}^{-1})$;*
- *for each pair of relations R_1 and R_2 in $\mathcal{S}$, $\mathcal{K}$ contains the assertion $\Phi(R_1) \sqsubseteq \neg\Phi(R_2)$, and for each relation R and each entity E it contains the assertion $\Phi(R) \sqsubseteq \neg\Phi(E)$.*

The mapping demonstrates that both inverse roles and number restrictions are necessary in order to capture the semantics of ER-schemata. We observe that binary relations could be treated in a simpler way by mapping them directly to $\mathcal{ALUNI}$-roles. Notice also that the assumption of acyclicity of inclusion assertions is unrealistic when representing ER-schemata. The following theorem ensures that reasoning in the ER-model can be reduced to finite satisfiability and finite implication in $\mathcal{ALUNI}$ knowledge bases.

Theorem 2.4 *An entity $E \in \mathcal{E}_{\mathcal{S}}$ is satisfiable in an ER-schema $\mathcal{S}$ if and only if $\Phi(\mathcal{S})$ admits a finite model $\mathcal{I}$ in which $E^{\mathcal{I}} \neq \emptyset$.*

2.3 OBJECT-ORIENTED DATA MODELS

Object-Oriented (OO) data models have been proposed with the goal of devising database formalisms that could be integrated with OO-programming systems (see [Kim90]). They are the subject of an active area of research in the database field, and are based on the following features: (a) in contrast to traditional data models which are value-oriented, they rely on the notion of object identifiers at the extensional level, and on the notion of class at the intensional level; (b) the structure of the classes is specified by means of typing and inheritance.

Figure 3 shows the OO-schema corresponding to a fragment of the KEE knowledge base of Figure 1. The formalization in $\mathcal{ALUNI}$ is given by:

$$
\begin{array}{rcl}
\texttt{Course} & \sqsubseteq & \texttt{AbstractClass} \sqcap (= 1\,\texttt{VALUE}) \sqcap \\
& & \forall \texttt{VALUE}.(\texttt{RecType} \sqcap \forall \texttt{ENROLLS.SetStud} \sqcap \\
& & \qquad (= 1\,\texttt{ENROLLS}) \sqcap \\
& & \qquad \forall \texttt{TAUGHTBY.Teacher} \sqcap \\
& & \qquad (= 1\,\texttt{TAUGHTBY})) \\
\texttt{SetStud} & \sqsubseteq & \texttt{SetType} \sqcap \forall \texttt{MEMBER.Student} \\
\texttt{Teacher} & \sqsubseteq & \texttt{AbstractClass} \sqcap (\texttt{Grad} \sqcup \texttt{Professor}) \\
\texttt{Grad} & \sqsubseteq & \texttt{AbstractClass} \sqcap \texttt{Student} \sqcap (= 1\,\texttt{VALUE}) \sqcap \\
& & \forall \texttt{VALUE}.(\texttt{RecType} \sqcap \forall \texttt{DEGREE.String} \sqcap \\
& & \qquad (= 1\,\texttt{DEGREE})) \\
\texttt{SetType} & \sqsubseteq & \neg\texttt{AbstractClass} \sqcap \neg\texttt{RecType} \\
\texttt{RecType} & \sqsubseteq & \neg\texttt{AbstractClass}
\end{array}
$$

The example shows that both classes and type structures of the OO-schema are translated into $\mathcal{ALUNI}$ concepts. We analyze now this correspondence more in detail by providing both the formal definition of the language used for specifying OO-schemata, and the mapping from OO-schemata to $\mathcal{ALUNI}$ knowledge bases. The OO-language is in the style of most popular models featuring complex objects and object identity. In particular, we follow [AK89], although with a slightly different syntax.

An OO-schema $\mathcal{S}$ is constituted by a set of class names, a set of attribute names, and a set of class declarations. Class declarations make use of type expressions over $\mathcal{S}$, which are built according to the following syntax (where C denotes a class name, A_i an attribute name, and T, T_i type expressions):

$$
\begin{array}{rcl}
T, T_1, \ldots, T_k & \longrightarrow & C \mid \\
& & \underline{\text{Union}}\ T_1, \ldots, T_k \mid \\
& & \underline{\text{Set-of}}\ T \mid \\
& & \underline{\text{Record}}\ A_1\!:\!T_1; \ldots; A_k\!:\!T_k\ \underline{\text{End}}
\end{array}
$$

The meaning of an OO-schema is given by specifying the characteristics of an instance of the schema. The definition of instance makes use of the notions of object identifiers and values. Given an OO-schema $\mathcal{S}$ and a finite set $\mathcal{O}$ of *object identifiers* denoting real world objects, the set $\mathcal{V}$ of *values* over $\mathcal{S}$ and $\mathcal{O}$ is inductively defined as follows:

- $\mathcal{O} \subseteq \mathcal{V}$;
- if $v_1, \ldots, v_k \in \mathcal{V}$ then $\{v_1, \ldots, v_k\} \in \mathcal{V}$;
- if $v_1, \ldots, v_k \in \mathcal{V}$ then $[A_1\!:\!v_1, \ldots, A_k\!:\!v_k] \in \mathcal{V}$;
- nothing else is in $\mathcal{V}$.

A database instance $\mathcal{I}$ of a schema $\mathcal{S}$ is constituted by a finite set $\mathcal{O}$ of object identifiers, a mapping π that assigns to each class name a subset of $\mathcal{O}$, and a mapping ρ assigning a value in $\mathcal{V}$ to each object in $\mathcal{O}$. The interpretation of type expressions in $\mathcal{I}$ is defined through an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns to each type expression a subset of $\mathcal{V}$ as follows:

$$
\begin{array}{rcl}
C^{\mathcal{I}} & = & \pi(C) \\
(\underline{\text{Union}}\ T_1, \ldots, T_k)^{\mathcal{I}} & = & T_1^{\mathcal{I}} \cup \cdots \cup T_k^{\mathcal{I}} \\
(\underline{\text{Set-of}}\ T)^{\mathcal{I}} & = & \{\{v_1, \ldots, v_k\} \mid k \geq 0, \\
& & \qquad v_i \in T^{\mathcal{I}}, i \in 1..k\ \} \\
(\underline{\text{Record}}\ A_1\!:\!T_1; \ldots; A_k\!:\!T_k\ \underline{\text{End}})^{\mathcal{I}} = & & \\
\quad \{[A_1\!:\!v_1, \ldots, A_h\!:\!v_h] \mid h \geq k, v_i \in T_i^{\mathcal{I}}, i \in 1..k, & & \\
\qquad v_j \in \mathcal{V}, j \in k+1..h\ \} & &
\end{array}
$$

The set of class declarations of an OO-schema is used to specify the structure of the objects in an instance of the database. Each declaration has the form

$$\underline{\text{Class}}\ C\ \underline{\text{Is-a}}\ C_1, \ldots, C_n\ \underline{\text{Type-is}}\ T.$$

The Is-a part of such a declaration allows to specify inclusion between the sets of instances of the involved classes, while the Type-is part specifies the structure allowed for the values assigned to the objects that are instances of the class. This justifies the following definition:

Definition 2.5 *Let $\mathcal{S}$ be an OO-schema. A database instance $\mathcal{I}$ is said to be* legal *with respect to $\mathcal{S}$ if for each declaration*

$$\underline{\textit{Class}}\ C\ \underline{\textit{Is-a}}\ C_1, \ldots, C_n\ \underline{\textit{Type-is}}\ T$$

in $\mathcal{S}$, it holds that $C^{\mathcal{I}} \subseteq C_i^{\mathcal{I}}$ for each $i \in 1..n$, and $\rho(C^{\mathcal{I}}) \subseteq T^{\mathcal{I}}$.

The relationship between $\mathcal{ALUNI}$ and the OO-language presented above is provided by means of a mapping from OO-schemata into $\mathcal{ALUNI}$ knowledge bases. Since the interpretation domain for $\mathcal{ALUNI}$ knowledge bases consists of objects without structures whereas the instances of OO-schemata refer to a structured universe (see the definition of $\mathcal{V}$), we need to explicitly represent some of the notions that underlie the OO-language. In particular, while there is a correspondence between concepts and classes, one must explicitly account for the type structure of each class.

```
Class Course Type-is
    Record
            ENROLLS: Set-of Student;
            TAUGHTBY: Teacher
    End

Class Teacher Type-is
    Union Professor, Grad
    End

Class Grad Is-a Student Type-is
    Record
            DEGREE: String
    End
```

Figure 3: An Object-Oriented data schema

This can be accomplished by introducing the atomic concepts `AbstractClass` to represent the classes in the OO-schema, and `RecType` and `SetType` to represent the corresponding types. The associations between classes and types induced by the class declarations, as well as the basic characteristics of types, are modeled by means of atomic roles: the (functional) role `VALUE` models the association between classes and types, and the role `MEMBER` is used for specifying the type of the elements of a set. Moreover, the concepts representing types are assumed to be mutually disjoint, and disjoint from the concepts representing classes. These constraints are expressed by the following assertions that will be part of the $\mathcal{ALUNI}$ knowledge base $\mathcal{K}$ derived from the schema

$$\begin{aligned} \texttt{SetType} &\sqsubseteq \neg\texttt{AbstractClass} \sqcap \neg\texttt{RecType} \\ \texttt{RecType} &\sqsubseteq \neg\texttt{AbstractClass} \end{aligned}$$

We now define the function Ψ that maps each type expression into an $\mathcal{ALUNI}$ concept expression as follows:

- every class C is mapped into an atomic concept $\Psi(C)$;
- every type expression Union $T_1, \ldots, T_k$ is mapped into $\Psi(T_1) \sqcup \cdots \sqcup \Psi(T_k)$;
- every type expression Set-of T is mapped into the concept $\texttt{SetType} \sqcap \forall \texttt{MEMBER}.\Psi(T)$.
- every attribute A is mapped into an atomic role $\Psi(A)$, and every type expression Record $A_1\!:T_1; \ldots; A_k\!:T_k$ End is mapped into the concept

$$\begin{aligned} \texttt{RecType} \sqcap \forall\Psi(A_1).\Psi(T_1) \sqcap (=1\,\Psi(A_1)) \sqcap \cdots \sqcap \\ \forall\Psi(A_k).\Psi(T_k) \sqcap (=1\,\Psi(T_k)); \end{aligned}$$

Definition 2.6 *The $\mathcal{ALUNI}$ knowledge base $\Psi(\mathcal{S})$ corresponding to an OO-schema $\mathcal{S}$ is constituted by*

- *the inclusion assertions that express mutual disjointness of* `AbstractClass`*,* `RecType`*, and* `SetType`*;*
- *an inclusion assertion*

$$\begin{aligned} \Psi(C) \sqsubseteq\ & \texttt{AbstractClass} \sqcap \\ & \Psi(C_1) \sqcap \cdots \sqcap \Psi(C_n) \sqcap \\ & \forall\texttt{VALUE}.\Psi(T) \sqcap (=1\,\texttt{VALUE}) \end{aligned}$$

for each class declaration

$$\underline{\textit{Class}}\ C\ \underline{\textit{Is-a}}\ C_1, \ldots, C_n\ \underline{\textit{Type-is}}\ T$$

in $\mathcal{S}$.

From the above correspondence, we can observe that inverse roles are not necessary for the formalization of OO-data models. Indeed, the possibility of referring to the inverse of an attribute is generally ruled out in such models. However, recent papers (see for example [AGO91]) point out that this strongly limits the expressive power of the data model. Note also that the use of number restrictions is limited to the value 1, which corresponds to existence constraints and functionality, whereas union is used in a more general form than in the KEE system.

The effectiveness of the mapping Ψ is sanctioned by the following theorem.

Theorem 2.7 *For every OO-schema $\mathcal{S}$, there exist two correspondences α, β between instances of a schema $\mathcal{S}$ and interpretations of its translation $\Psi(\mathcal{S})$ such that, for each legal instance $\mathcal{I}$ of $\mathcal{S}$, $\alpha(\mathcal{I})$ is a model of $\Psi(\mathcal{S})$, and, on the converse, for each model $\mathcal{M}$ of $\Psi(\mathcal{S})$, $\beta(\mathcal{M})$ is a legal instance of $\mathcal{S}$.*

The basic reasoning services considered in OO-databases are subtyping (check whether a type denotes a subset of another type in every legal instance) and type checking (check whether an instance is legal). Based on theorem 2.7, it is possible to show that these forms of reasoning are fully captured by finite satisfiability and implication in $\mathcal{ALUNI}$ knowledge bases.

2.4 DISCUSSION

The above subsections should clarify that the language $\mathcal{ALUNI}$ and the associated reasoning capabilities represent the essential core of the class-based representation formalisms belonging to all three families mentioned above. On the other hand, we have shown that the formalism proposed in this paper provides important features that are currently missing in each family, although their relevance has often been stressed. In this sense, the work reported here not only provides a common powerful representation formalism, but may also contribute to significant developments for the languages belonging to all the three families. For this purpose it is essential to develop adequate techniques for reasoning in all of the above contexts. This implies that we have to deal with both unrestricted and finite satisfiability and implication. In the following two sections we present the reasoning methods for the two cases. Due to space limitations, in this paper we concentrate on satisfiability only; a direct extension of the methods provides decision procedures for logical implication too.

3 REASONING IN UNRESTRICTED MODELS

In order to show that the problem of checking unrestricted satisfiability of an $\mathcal{ALUNI}$ knowledge base is decidable, we make use of a correspondence between $\mathcal{ALUNI}$ and a sublanguage of deterministic converse propositional dynamic logic ($\mathcal{CPDL}$). Although this correspondence is similar to the one established in

[Sch91], due to the presence of number restrictions, we cannot directly make use of the known results.

The basic idea of our method is to show that standard reasoning techniques for $\mathcal{CPDL}$ can still be exploited if we perform a preliminary transformation of the knowledge base that allows us to weaken the constraints imposed by number restrictions. We call the knowledge base $\mathcal{K}_{rel}$ resulting from the application of the transformation to a knowledge base $\mathcal{K}$, the *relaxation of* $\mathcal{K}$, defined as follows:

- all number restrictions $(\geq m\, R)$, with $m \neq 1$, and $(\leq n\, R)$ in $\mathcal{K}$ are treated as new symbols for atomic concepts in $\mathcal{K}_{rel}$;
- for each pair of number restrictions $(\geq m\, R)$ and $(\leq n\, R)$ present in $\mathcal{K}$, such that $m > n$, the assertion $(\geq m\, R) \sqsubseteq \neg(\leq n\, R)$ is added to $\mathcal{K}_{rel}$.
- for each number restriction $(\geq m\, R)$, with $m \neq 1$, present in $\mathcal{K}$, the assertion $(\geq m\, R) \sqsubseteq (\geq 1\, R)$ is added to $\mathcal{K}_{rel}$.

The following lemma gives a necessary condition for the satisfiability of an $\mathcal{ALUNI}$ knowledge base.

Lemma 3.1 *If an $\mathcal{ALUNI}$ knowledge base is satisfiable, then its relaxation is also satisfiable.*

In the rest of the section we show that the converse of Lemma 3.1 also holds. This is done by exploiting the model preserving transformation of $\mathcal{K}_{rel}$ into a formula ϕ_{rel} of $\mathcal{CPDL}$. Notice that since in $\mathcal{K}_{rel}$ all number restrictions are treated as atomic concepts, the transformation is defined in the same way as in [Sch91]. The resulting formula belongs to a sublanguage of $\mathcal{CPDL}$ which we call $\mathcal{CPDL}^-$. In $\mathcal{CPDL}^-$, programs, denoted with p, q, and formulae, denoted with ϕ, ψ, are built from atomic programs P and atomic formulae A by the following syntax rules:

$$\begin{array}{rcl} p, q & \longrightarrow & P \mid P^- \mid p^* \mid p \cup q \mid p; q \\ \phi, \psi & \longrightarrow & \top \mid A \mid \neg A \mid \phi \vee \psi \mid \phi \wedge \psi \mid \\ & & \langle P \rangle \top \mid \langle P^- \rangle \top \mid [p]\phi. \end{array}$$

We will use the term *basic program* to denote an atomic program or the inverse of an atomic program. The semantics of $\mathcal{CPDL}^-$ is derived from the semantics of $\mathcal{CPDL}$ in a straightforward way (see for example [KT90]).

For an example of a transformation of a knowledge base into a $\mathcal{CPDL}^-$ formula, see Figure 4, showing a knowledge base $\mathcal{K}$, describing the properties of trees in which each node has at least two outgoing edges, the relaxation $\mathcal{K}_{rel}$, and the $\mathcal{CPDL}^-$ formula corresponding to $\mathcal{K}_{rel}$. Note that $(\geq 1\,\mathtt{ARC})$ is transformed into $\langle \mathtt{ARC} \rangle \top$.

In the following, let ϕ_{rel} be the $\mathcal{CPDL}^-$ formula obtained from the relaxation $\mathcal{K}_{rel}$ of $\mathcal{K}$ and let $\mathcal{M}$

$\mathcal{K}$:

$$\begin{array}{lcl} \mathtt{Root} & \sqsubseteq & \forall \mathtt{ARC}^{-1}.\bot \sqcap \forall \mathtt{ARC}.\mathtt{Node} \sqcap (\geq 2\,\mathtt{ARC}) \sqcap \neg \mathtt{Node} \\ \mathtt{Node} & \sqsubseteq & \forall \mathtt{ARC}^{-1}.(\mathtt{Root} \sqcup \mathtt{Node}) \sqcap \forall \mathtt{ARC}.\mathtt{Node} \sqcap \\ & & (\geq 2\,\mathtt{ARC}) \sqcap (= 1\,\mathtt{ARC}^{-1}) \end{array}$$

$\mathcal{K}_{rel}$:

$$\begin{array}{lcl} \mathtt{Root} & \sqsubseteq & \forall \mathtt{ARC}^{-1}.\bot \sqcap \forall \mathtt{ARC}.\mathtt{Node} \sqcap (\geq 2\,\mathtt{ARC}) \sqcap \\ & & \neg \mathtt{Node} \\ \mathtt{Node} & \sqsubseteq & \forall \mathtt{ARC}^{-1}.(\mathtt{Root} \sqcup \mathtt{Node}) \sqcap \forall \mathtt{ARC}.\mathtt{Node} \sqcap \\ & & (\geq 2\,\mathtt{ARC}) \sqcap (= 1\,\mathtt{ARC}^{-1}) \\ (\geq 2\,\mathtt{ARC}) & \sqsubseteq & (\geq 1\,\mathtt{ARC}) \end{array}$$

$$\begin{array}{l} \phi_{rel} = [(\mathtt{ARC} \cup \mathtt{ARC}^-)^*] \\ \quad ((\neg \mathtt{Root} \vee ([\mathtt{ARC}^-]\bot \wedge [\mathtt{ARC}]\mathtt{Node} \wedge \\ \qquad A_{(\geq 2\,\mathtt{ARC})} \wedge \neg \mathtt{Node})) \wedge \\ \quad (\neg \mathtt{Node} \vee ([\mathtt{ARC}^-](\mathtt{Root} \vee \mathtt{Node}) \wedge [\mathtt{ARC}]\mathtt{Node} \wedge \\ \qquad A_{(\geq 2\,\mathtt{ARC})} \wedge \\ \qquad A_{(\geq 1\,\mathtt{ARC}^-)} \wedge A_{(\leq 1\,\mathtt{ARC}^-)})) \wedge \\ \quad (\neg A_{(\geq 2\,\mathtt{ARC})} \vee \langle \mathtt{ARC} \rangle \top)) \end{array}$$

Figure 4: An $\mathcal{ALUNI}$ knowledge base, its relaxation, and the corresponding $\mathcal{CPDL}^-$ formula

be a model of ϕ_{rel}. ϕ_{rel} will contain atomic formulae $A_{(\geq m\, R)}$ and $A_{(\leq n\, R)}$ for each number restriction $(\geq m\, R)$ and $(\leq n\, R)$ present in $\mathcal{K}$. We say that a state s of $\mathcal{M}$ *numerically satisfies* $A_{(\geq m\, R)}$ if $\sharp \left\{ t \mid (s,t) \in R^{\mathcal{M}} \right\} \geq m$. Similarly, s numerically satisfies $A_{(\leq n\, R)}$, if $\sharp \left\{ t \mid (s,t) \in R^{\mathcal{M}} \right\} \leq n$. According to these definitions, any model $\mathcal{M}$ of ϕ_{rel} is also a model of $\mathcal{K}$ if all states of $\mathcal{M}$ numerically satisfy all atomic formulae corresponding to number restrictions. We show that if ϕ_{rel} is satisfiable then we can construct a model in which this is indeed the case.

[Str82] shows that the *tree model property* holds for $\mathcal{CPDL}$ (see [Str82] for a formal definition of *tree model*). This result carries over immediately to $\mathcal{CPDL}^-$, and therefore every satisfiable $\mathcal{CPDL}^-$ formula admits a model which is a tree, if we view each state as a node and each transition between states as an arc labeled with the corresponding program. However, we can show that for $\mathcal{CPDL}^-$ an even stronger result holds, which is based on the following definition.

Definition 3.2 *A* deterministic direct-inverse interpretation *is an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that for each state $s \in \Delta^{\mathcal{I}}$ and for each atomic program P, there is at most one state $t \in \Delta^{\mathcal{I}}$ such that $(s,t) \in P^{\mathcal{I}}$ and at most one state $r \in \Delta^{\mathcal{I}}$ such that $(r,s) \in P^{\mathcal{I}}$.*

Lemma 3.3 *Every satisfiable $\mathcal{CPDL}^-$ formula ϕ admits a deterministic direct-inverse tree model.*

Proof (sketch). Since ϕ is satisfiable, by the tree model property it admits a tree model $\mathcal{T}$. Starting from $\mathcal{T}$ we can construct a deterministic direct-inverse tree model

$\mathcal{D}$, proceeding by induction on the depth of $\mathcal{T}$ and removing for each state all but one of the arcs incident to that state and labeled with the same basic program. Since the only subformulae of ϕ involving $\langle\cdot\rangle$, are of the form $\langle R\rangle\top$, where R is a basic program, we can prove by induction on the structure of ϕ that the root of $\mathcal{D}$ still satisfies ϕ. □

Notice that in any deterministic direct-inverse tree model $\mathcal{D}$ all atomic formulae $A_{(\leq n\,R)}$ are already numerically satisfied in all states of $\mathcal{D}$ and no formula $A_{(\geq m\,R)}$ with $m > 1$ is numerically satisfied in any state of $\mathcal{D}$. The following lemma guarantees that we can transform any such model in one in which all atomic formulae are numerically satisfied in all states.

Lemma 3.4 *Let $\mathcal{D}$ be a deterministic direct-inverse tree model of the $\mathcal{CPDL}^-$ formula ϕ_{rel}. Then $\mathcal{D}$ can be transformed into a tree model $\mathcal{T}$ such that all atomic formulae in ϕ_{rel} that correspond to number restrictions are numerically satisfied in all states of $\mathcal{T}$.*

Proof (sketch). We can construct $\mathcal{T}$ in the following way: Initially we set $\mathcal{T}$ equal to $\mathcal{D}$ and then we proceed by induction on the depth of the tree we are constructing. For each state s of $\mathcal{T}$ that we are considering and for each basic program R appearing in ϕ, we consider the atomic formula $A_{(\geq m\,R)}$ with maximum m that is satisfied in s but is not numerically satisfied. The assertions added to $\mathcal{K}_{rel}$ ensure that there is at least one state t connected to s through R. We can take the whole tree structure starting in t and connected to s through R, duplicate it $m-1$ times and connect these $m-1$ trees to s via R. In this way we ensure that $A_{(\geq m\,R)}$ is numerically satisfied in s. Furthermore the assertions added to $\mathcal{K}_{rel}$ guarantee that by proceeding in this way all atomic formulae $A_{(\leq n\,R)}$ are still numerically satisfied in s. □

By combining the results of the previous lemmas we can conclude that the relaxation of a knowledge base in fact captures all relevant properties of the knowledge base itself. This is stated in the following theorem.

Theorem 3.5 *An $\mathcal{ALUNI}$ knowledge base $\mathcal{K}$ is satisfiable if and only if its relaxation $\mathcal{K}_{rel}$ is satisfiable.*

In [VW84] it has been shown that deciding if a $\mathcal{CPDL}$ formula is satisfiable can be done in deterministic exponential time, which also gives the upper bound for satisfiability in $\mathcal{CPDL}^-$. We have seen that $\mathcal{K}_{rel}$ can be transformed in a straightforward way into a formula ϕ_{rel} of $\mathcal{CPDL}^-$ whose size is polynomial in the size of $\mathcal{K}_{rel}$ and which is satisfiable if and only if it is $\mathcal{K}_{rel}$. Therefore we get immediately the following corollary of the previous theorem.

Corollary 3.6 *Unrestricted satisfiability and implication for an $\mathcal{ALUNI}$ knowledge base can be decided in deterministic exponential time.*

4 REASONING IN FINITE MODELS

In this section we sketch a method for verifying finite satisfiability of an $\mathcal{ALUNI}$ knowledge base $\mathcal{K}$. This task requires a quite different approach form the one used for the unrestricted case, since the actual numbers that appear in the number restrictions of $\mathcal{K}$ play a crucial role in the existence of finite models. For this reason we model number restrictions by means of an associated system $\Psi_{\mathcal{K}}$ of linear disequations, defined in such a way that the existence of a finite model for $\mathcal{K}$ is reflected into the existence of particular solutions of $\Psi_{\mathcal{K}}$.

The unknowns introduced in $\Psi_{\mathcal{K}}$ are intended to represent the number of instances of each concept and each role in a possible finite model of $\mathcal{K}$, while the disequations take into account the constraints on the number of instances deriving from number restrictions in $\mathcal{K}$. Because of atomic concepts that may have instances in common, it is not possible to adopt the most natural approach which would be to use one unknown for each atomic concept and role (see [LN90]). We will overcome this problem by introducing the notion of expansion of a knowledge base.

In the sequel we will use the term *literal* for an atomic or negated atomic concept. A concept will be called *simple* if it is of the form: $L \mid L_1 \sqcup L_2 \mid \forall R.L \mid (\geq m\,R) \mid (\leq n\,R)$, where L, L_1 and L_2 are literals. A knowledge base whose inclusion assertions have a simple concept on the right hand side is said to be *simple*. Since a generic knowledge base $\mathcal{K}$ can be transformed in linear time into a simple knowledge base $\mathcal{K}'$ that is finitely satisfiable if and only if it is $\mathcal{K}$ we can restrict our attention to simple knowledge bases.

Therefore in each assertion of a knowledge base $\mathcal{K}$ at most one operator appears on the right hand side. We will denote with $\mathcal{K}_X$, where $X \in \{\sqcup, \forall, \geq, \leq\}$, the subset of assertions involving operator X, and with $\mathcal{K}_{isa}$ those involving only literals.

Let $\mathcal{C}$ be the set of all atomic concepts present in $\mathcal{K}$, together with the symbol $\top$. A *compound concept* is defined as a subset of $\mathcal{C}$ containing $\top$. Intuitively a compound concept $\bar{C}$ represents exactly those elements of the domain that are instances of all atomic concepts in $\bar{C}$ and are not instances of all atomic concepts not in $\bar{C}$. More formally, the extension $\bar{C}^{\mathcal{I}}$ of $\bar{C}$ is defined as:

$$\bar{C}^{\mathcal{I}} = \bigcap \{A^{\mathcal{I}} \mid A \in \bar{C}\} \setminus \bigcup \{A^{\mathcal{I}} \mid A \in \mathcal{C} \setminus \bar{C}\}.$$

Let $\mathcal{R}$ be the set of all atomic roles present in $\mathcal{K}$. We represent explicitly for each such role the association with all possible pairs of compound concepts. This can

be accomplished by defining a *compound role* as an indexed pair $\langle \bar{C}_1, \bar{C}_2 \rangle_P$, where $\bar{C}_1$ and $\bar{C}_2$ are compound concepts and P is an atomic role appearing in $\mathcal{K}$. It is interpreted as the restriction of $P^{\mathcal{I}}$ to pairs whose first and second element belong to $\bar{C}_1^{\mathcal{I}}$ and $\bar{C}_2^{\mathcal{I}}$ respectively.

Notice that the way we interpret compound concepts and roles forces them to be disjoint in all interpretations. This property is crucial in order to construct a model from a solution of the system of disequations. The price we have to pay for it is the exponential number of different compound concepts and roles. We are now ready to give the following definition.

Definition 4.1 *The* expansion $\bar{\mathcal{K}}$ *of an* $\mathcal{ALUNI}$ *knowledge base* $\mathcal{K}$ *is constituted by*

- *the set* $\bar{\mathcal{C}}$ *of all compound concepts of* $\mathcal{K}$ *and the set* $\bar{\mathcal{R}}$ *of all compound roles of* $\mathcal{K}$*;*
- *all assertions of* $\mathcal{K}_{isa} \cup \mathcal{K}_{\sqcup} \cup \mathcal{K}_{\forall}$*;*
- *a set* $\mathcal{K}_{num}$ *of assertions involving a compound concept on the left hand side and a number restriction on the right hand side, obtained in the following way: for each compound concept* $\bar{C}$ *and for each role* R*:*

  – if for some $A \in \bar{C}$ *and some positive integer* m*,* $A \sqsubseteq (\geq m\, R)$ *is in* $\mathcal{K}_{\geq}$*, then* $\bar{C} \sqsubseteq (\geq m_{max}\, R)$ *is in* $\mathcal{K}_{num}$*, where*

  $$m_{max} = \max\left\{m \mid A \sqsubseteq (\geq m\, R) \in \mathcal{K}_{\geq} \wedge A \in \bar{C}\right\}$$

  – if for some $A \in \bar{C}$ *and some positive integer* n*,* $A \sqsubseteq (\leq n\, R)$ *is in* $\mathcal{K}_{\leq}$*, then* $\bar{C} \sqsubseteq (\leq n_{min}\, R)$ *is in* $\mathcal{K}_{num}$*, where*

  $$n_{min} = \min\left\{n \mid A \sqsubseteq (\leq n\, R) \in \mathcal{K}_{\leq} \ \wedge \ A \in \bar{C}\right\}.$$

From the expansion $\bar{\mathcal{K}}$ we can derive a system $\Psi_{\mathcal{K}}$ of linear disequations, with one unknown $\mathrm{Var}(\bar{C})$ for each compound concept $\bar{C}$, and one unknown $\mathrm{Var}(\bar{R})$ for each compound role $\bar{R}$. The disequations of $\Psi_{\mathcal{K}}$ are obtained in the following way:

- It is possible to check in polynomial time, with respect to the size of the expansion, whether a compound concept $\bar{C}$ is consistent with $\mathcal{K}_{isa} \cup \mathcal{K}_{\sqcup}$, i.e. whether there is a model $\mathcal{I}$ of $\mathcal{K}_{isa} \cup \mathcal{K}_{\sqcup}$ such that $\bar{C}^{\mathcal{I}}$ is nonempty. In a similar way, we can check whether a compound role is consistent with $\mathcal{K}_{\forall}$. We force to be equal to 0 all those unknowns corresponding to compound concepts and roles that are not consistent respectively with $\mathcal{K}_{isa} \cup \mathcal{K}_{\sqcup}$ and with $\mathcal{K}_{\forall}$, and force to be nonnegative all the others.
- We introduce disequations that reflect the number restrictions by relating the unknown corresponding to a compound concept $\bar{C}$ to the sum of the unknowns corresponding to compound roles in which $\bar{C}$ appears. As an example, if $\bar{C} \sqsubseteq (\geq m\, P) \in \mathcal{K}_{num}$, where P is an atomic role, we introduce $m \cdot \mathrm{Var}(\bar{C}) \leq \sum_{\bar{C}_2 \in \bar{\mathcal{C}}} \mathrm{Var}(\langle \bar{C}, \bar{C}_2 \rangle_P)$.

$\bar{\mathcal{C}} = \{\mathtt{R}, \mathtt{N}, \mathtt{O}, \mathtt{RN}, \mathtt{RO}, \mathtt{NO}, \mathtt{RNO}\}$, where
$\mathtt{R} = \{\mathtt{Root}\}$, $\mathtt{N} = \{\mathtt{Node}\}$, $\mathtt{O} = \{\mathtt{RootOrNode}\}$,
$\mathtt{RN} = \{\mathtt{Root}, \mathtt{Node}\}$, $\mathtt{RO} = \{\mathtt{Root}, \mathtt{RootOrNode}\}$,
$\mathtt{NO} = \{\mathtt{Node}, \mathtt{RootOrNode}\}$,
$\mathtt{RNO} = \{\mathtt{Root}, \mathtt{Node}, \mathtt{RootOrNode}\}$;
Subset of $\bar{\mathcal{C}}$ consistent with $\mathcal{K}_{isa} \cup \mathcal{K}_{\sqcup}$: $\{\mathtt{R}, \mathtt{N}, \mathtt{RO}, \mathtt{NO}\}$;

$\bar{\mathcal{R}} = \{\langle \bar{C}, \bar{C}' \rangle_{\mathtt{ARC}} \mid \bar{C}, \bar{C}' \in \bar{\mathcal{C}}\}$;
Consistent compound roles:
$\{\langle \mathtt{RO}, \mathtt{N} \rangle_{\mathtt{ARC}}, \langle \mathtt{RO}, \mathtt{NO} \rangle_{\mathtt{ARC}}, \langle \mathtt{NO}, \mathtt{N} \rangle_{\mathtt{ARC}}, \langle \mathtt{NO}, \mathtt{NO} \rangle_{\mathtt{ARC}}\}$;

$\mathcal{K}_{num}$	$\Psi_{\mathcal{K}}$
$\mathtt{R} \sqsubseteq (\geq 2\,\mathtt{ARC})$	$2\mathtt{r} \leq 0$
$\mathtt{RO} \sqsubseteq (\geq 2\,\mathtt{ARC})$	$2\mathtt{ro} \leq \mathtt{arc}_{\mathtt{ro},\mathtt{n}} + \mathtt{arc}_{\mathtt{ro},\mathtt{no}}$
$\mathtt{N} \sqsubseteq (\geq 2\,\mathtt{ARC})$	$2\mathtt{n} \leq 0$
$\mathtt{N} \sqsubseteq (\geq 1\,\mathtt{ARC}^{-1})$	$\mathtt{n} \leq \mathtt{arc}_{\mathtt{ro},\mathtt{n}} + \mathtt{arc}_{\mathtt{no},\mathtt{n}}$
$\mathtt{N} \sqsubseteq (\leq 1\,\mathtt{ARC}^{-1})$	$\mathtt{n} \geq \mathtt{arc}_{\mathtt{ro},\mathtt{n}} + \mathtt{arc}_{\mathtt{no},\mathtt{n}}$
$\mathtt{NO} \sqsubseteq (\geq 2\,\mathtt{ARC})$	$2\mathtt{no} \leq \mathtt{arc}_{\mathtt{no},\mathtt{n}} + \mathtt{arc}_{\mathtt{no},\mathtt{no}}$
$\mathtt{NO} \sqsubseteq (\geq 1\,\mathtt{ARC}^{-1})$	$\mathtt{no} \leq \mathtt{arc}_{\mathtt{ro},\mathtt{no}} + \mathtt{arc}_{\mathtt{no},\mathtt{no}}$
$\mathtt{NO} \sqsubseteq (\leq 1\,\mathtt{ARC}^{-1})$	$\mathtt{no} \geq \mathtt{arc}_{\mathtt{ro},\mathtt{no}} + \mathtt{arc}_{\mathtt{no},\mathtt{no}}$

Figure 5: The expansion of the knowledge base shown in Figure 4

Figure 5 shows the expansion of the simple knowledge base derived from the one of Figure 4 and the corresponding system of disequations. Each unknown is given the name of the corresponding compound concept or role, but in lower case (for brevity we have not included unknowns corresponding to inconsistent compound concepts and roles). The concept `RootOrNode` derives from the transformation into a simple knowledge base.

The system of disequations we obtain from the expansion of the knowledge base is linear and homogeneous and admits only nonnegative solutions. The following theorem relates the existence of particular solutions of this system to the existence of finite models for the knowledge base from which the disequations are derived.

We call a solution of $\Psi_{\mathcal{K}}$ *acceptable* if it assigns a positive value to at least one unknown, and for all compound roles $\bar{R} = \langle \bar{C}_1, \bar{C}_2 \rangle_P$, the value assigned to $\mathrm{Var}(\bar{R})$ is 0 whenever the value assigned to either $\bar{C}_1$ or $\bar{C}_2$ is 0.

Theorem 4.2 $\mathcal{K}$ *is finitely satisfiable if and only if* $\Psi_{\mathcal{K}}$ *admits an acceptable integer solution.*

Proof (sketch). Given an acceptable integer solution $\mathcal{X}$ of $\Psi_{\mathcal{K}}$, it is possible to construct a model of $\mathcal{K}$ such that the number of instances of each compound concept and role is exactly the value assigned by $\mathcal{X}$ to the corre-

sponding unknown. Since $\mathcal{X}$ is nontrivial, the model constructed will be nonempty. Conversely, given a finite model $\mathcal{M}$ of $\mathcal{K}$ it is possible to show that we obtain a solution of $\mathcal{X}$ by assigning to each unknown the number of instances in $\mathcal{M}$ of the corresponding compound concept or role. These can be directly deduced from the interpretations of all concepts and roles. □

In order to make use of this result and show that we can reason with respect to finite models in $\mathcal{ALUNI}$ knowledge bases, we have to guarantee that verifying the existence of acceptable integer solutions for a system of disequations is decidable. This is indeed the case and, by using linear programming techniques it can be proved that it takes polynomial time in the size of the system. Therefore we can state the following theorem.

Theorem 4.3 *Finite satisfiability and implication for an $\mathcal{ALUNI}$ knowledge base can be decided in deterministic exponential time.*

Proof (sketch). The decidability follows immediately from theorem 4.2 and the previous observation. The exponential upper bound derives from the exponential size of the system of disequations and the polinomial time required for the search of an acceptable solution of the system. □

5 CONCLUSIONS

We have presented a unified framework for representing information about class structures and reasoning about them. We have pursued this goal by looking at various class-based formalisms proposed in different fields of computer science and trying to rephrase them in the framework of concept languages. The resulting language includes a combination of constructs that was not addressed before, although all of the constructs had previously been considered separately.

The major achievement of the paper is the demonstration that class-based formalisms can be given a precise characterization by means of a powerful first-order language where the basic reasoning problems remain decidable, in particular EXPTIME. This has several consequences.

First of all, any of the formalisms considered in the paper can be enriched with constructs originating from other formalisms and treated in the general framework. For example, the usage of inverse roles in concept languages greatly enhances the expressivity of roles, while the combination of ISA, number restrictions and union enriches the reasoning capabilities available in semantic data models.

Secondly, the comparison of class-based formalisms emphasizes the importance of distinguishing between unrestricted reasoning and reasoning in finite models. Although this aspect has seldom been considered in the case of knowledge representation formalisms, the assumption of finiteness seems to be appropriate in most applications, and must be addressed when the representation formalism becomes sufficiently powerful. We have developed a novel technique for finite model reasoning. Although we did not address the problems related to the practical behavior of the method, we point out that, on one hand the constraints imposed on the domain to be modeled make the worst case complexity rarely occur in practice, and on the other hand we can effectively exploit the technology of linear programming for the implementation of real systems.

Finally, it is worth mentioning that the results presented in this paper can be extended to deal both with more general inclusion assertions, and with the extensional level of the knowledge base, where assertions about the instance-of relation between individual objects and classes are specified.

Acknowledgements

This work was partly funded by the ESPRIT BRA Compulog II, and the Italian CNR under Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, LDR Ibridi.

References

[AGO91] A. Albano, G. Ghelli, and R. Orsini. A relationship mechanism for strongly typed Object-Oriented database programming languages. In *Proc. of the 17th Int. Conf. on Very Large Data Bases VLDB-91*, pages 565–575, Barcelona, 1991.

[AK89] S. Abiteboul and P. Kanellakis. Object identity as a query language primitive. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 159–173, 1989.

[AP86] P. Atzeni and D.S. Parker Jr. Formal properties of net-based knowledge representation schemes. In *Proc. of the 2nd IEEE Int. Conf. on Data Engineering*, pages 700–706, Los Angeles, 1986.

[BHR90] K.H. Bläsius, U. Hedstück, and C.-R. Rollinger, editors. *Sorts and Types in Artificial Intelligence.* Number 418 in Lecture Notes in Artificial Intelligence. Springer-Verlag, 1990.

[Bor92] Alexander Borgida. From type systems to knowledge representation: Natural semantics specifications for description logics. *Journal of Intelligent and Cooperative Inf. Syst.*, 1(1):93–126, 1992.

[BS92] Sonia Bergamaschi and Claudio Sartori. On taxonomic reasoning in conceptual design. *ACM Trans. on Database Syst.*, 17(3):385–422, 1992.

[Che76] P.P. Chen. The Entity-Relationship model: Toward a unified view of data. *ACM Trans. on Database Syst.*, 1(1):9–36, March 1976.

[CHS91] C. Collet, M.N. Huhns, and W. Shen. Resource integration using a large knowledge base in carnot. *IEEE Computer*, 24(12), 1991.

[CKV90] S.S. Cosmadakis, P.C. Kanellakis, and M. Vardi. Polynomial-time implication problems for unary inclusion dependencies. *Journal of the ACM*, 37(1):15–46, January 1990.

[CL93] Tiziana Catarci and Maurizio Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Inf. Syst.*, 1993. To appear.

[CL94] Diego Calvanese and Maurizio Lenzerini. On the interaction between ISA and cardinality constraints. In *Proc. of the 10th IEEE Int. Conf. on Data Engineering*, Houston, 1994. To appear.

[DLNN91] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. Tractable concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence IJCAI-91*, pages 458–463, Sydney, 1991.

[FK85] Richard Fikes and Tom Kehler. The role of frame-based representation in reasoning. *Communications of the ACM*, 28(9):904–920, 1985.

[HK87] R.B. Hull and R. King. Semantic database modelling: Survey, applications and research issues. *ACM Computing Surveys*, 19(3):201–260, September 1987.

[Kim90] Won Kim. *Introduction to Object-Oriented Databases*. The MIT Press, 1990.

[KL89] Won Kim and Frederick H. Lochovsky, editors. *Object-Oriented Concepts, Databases, and Applications*. ACM Press and Addison Wesley, New York, 1989.

[KT90] Dexter Kozen and Jerzy Tiuryn. Logics of programs. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science – Formal Models and Semantics*, pages 789–840. Elsevier Science Publishers (North-Holland), Amsterdam, 1990.

[Leh92] Fritz Lehmann, editor. *Semantic Networks in Artificial Intelligence*. Pergamon Press, Oxford, 1992.

[LN90] Maurizio Lenzerini and Paolo Nobili. On the satisfiability of dependency constraints in entity-relationship schemata. *Information Systems*, 15(4):453–461, 1990.

[LNS91] Maurizio Lenzerini, Daniele Nardi, and Maria Simi, editors. *Inheritance Hierarchies in Knowledge Representation and Programming Languages*. John Wiley & Sons, Chichester, 1991.

[MM92] R. Motschnig-Pitrik and J. Mylopoulous. Classes and instances. *Journal of Intelligent and Cooperative Inf. Syst.*, 1(1), 1992.

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

[PSS92] Barbara Piza, Klaus-Dieter Schewe, and Joachim W. Schmidt. Term subsumption with type constructors. In Y. Yesha, editor, *Proc. of the Int. Conf. on Information and Knowledge Management CIKM-92*, pages 449–456, Baltimore, 1992.

[Sch91] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence IJCAI-91*, pages 466–471, Sydney, 1991.

[Sow91] John F. Sowa, editor. *Principles of Semantic Networks*. Morgan Kaufmann, Los Altos, 1991.

[Str82] R.S. Street. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Control*, 54:121–141, 1982.

[VW84] M. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. In *Proc. of the 16th ACM SIGACT Symp. on Theory of Computing STOC-84*, pages 446–455, 1984.

[WS92] William A. Woods and James G. Schmolze. The KL-ONE family. In F.W. Lehmann, editor, *Semantic Networks in Artificial Intelligence*, pages 133–178. Pergamon Press, 1992. Published as a special issue of *Computers & Mathematics with Applications*, Volume 23, Number 2–9.