A Scheme for Integrating Concrete Domains into Concept Languages

Franz Baader* and Philipp Hanschke*
German Research Center for AI (DFKI)
Postfach 2080
W-6750 Kaiserslautern, Germany

Abstract

A drawback which concept languages based on KL-ONE have is that all the terminological knowledge has to be defined on an abstract logical level. In many applications, one would like to be able to refer to concrete domains and predicates on these domains when defining concepts. Examples for such concrete domains are the integers, the real numbers, or also non-arithmetic domains, and predicates could be equality, inequality, or more complex predicates. In the present paper we shall propose a scheme for integrating such concrete domains into concept languages rather than describing a particular extension by some specific concrete domain. We shall define a terminological and an assertional language, and consider the important inference problems such as subsumption, instantiation, and consistency. The formal semantics as well as the reasoning algorithms can be given on the scheme level. In contrast to existing KL-ONE based systems, these algorithms are not only sound but also complete. They generate subtasks which have to be solved by a special purpose reasoner of the concrete domain.

1 Introduction

Concept languages based on KL-ONE [Brachman and Schmolze, 1985] are used to represent the taxonomical and conceptual knowledge of a particular problem domain on an abstract logical level. To describe this kind of knowledge, one starts with atomic concepts and roles, and defines new concepts using the operations provided by the language. Examples for atomic concepts may be Human and Female, and for roles child. If the logical connective conjunction is present as language construct, one may describe the concept Woman as "humans who are female", and represent it by the expression Human ⊓ Female. Many languages provide quantification over role fillers which allows for example to describe the concept Mother by the expression Woman D 3child.Human.

KL-ONE was first developed for the purpose of natural language processing [Brachman *et al,* 1979], and some of the existing systems are still mostly used in this context (see e.g., SB-ONE [Kobsa, 1989])- However, its success in this area has also led to applications in other fields (see e.g,, MESON [Edelmann and Owsnicki, 1986] which is used for computer configuration tasks, CLASSIC [Borgida *et* al,, 1989] which is e.g, used in the area of CAD/CAM, or K-Rep [Mays *et* al., 1988] which is used in a financial marketing domain),

A drawback which pure KL-ONE languages have is that all the terminological knowledge has to be defined on the abstract logical level. In many applications, one would like to be able to refer to concrete domains and predicates on these domains when defining concepts. An example for such a concrete domain could be the set of nonnegative integers. In the above example, one might think that being human and female is not enough to make a woman. As an additional property one could require that she should be old enough, e.g., at least 21. Thus one would like to introduce a new role age, and define Woman by an expression of the form Human ⊓ Female ⊓ $\geq_{21}$(age)- Here $\geq_{21}$ stands for the unary predicate $\{n ; \geq 21\}$ of all nonnegative integers greater or equal 21. Stating such properties directly with reference to a given concrete domain seems to be easier and more natural than encoding them somehow into abstract concept expressions.[1] Though this drawback already appears in natural language processing, it becomes even more important if one has other applications in mind. For example, in a technical application the adequate representation of geometrical concepts requires to relate points in a coordinate system. For that purpose one would e.g. like to have access to real arithmetic. Similar motivations have already led to extensions of KL-ONE in the above mentioned systems MESON, CLASSIC, and K-Rep. The MESON system provides "a separate hierarchy for de-

'Supported by the BMFT research project AKA-W1NO
Supported by the BMFT research project ARC-TEC

[1] See e.g. [Brachman and Schmolze, 1985], Section 9.2, where so-called Structural Descriptions are used to encode the concrete predicate "less than one hour". From a computational point of view, Structural Descriptions are as bad as Role Value Maps which cause undecidability of subsumption [Schmidt-SchauB, 1989].

scribing non-concepts (e.g., integer ranges and strings)" ([Patel-Schneider *et* al., 1990), p. 8) which are given as user-defined or machine-defined predicates. Similar features are provided by the "test" construct in CLASSIC. In K-Rep "the roles of concepts may in turn be other (complex) concepts, as well as numbers, strings and .., arbitrary Lisp objects" ([Mays *et al,* 1988], p. 62).

For similar reasons, Logic Programming has been extended to Constraint Logic Programming (CLP). The constraints in CLP languages "state properties directly in the domain of discourse as opposed to having these properties encoded into Prolog terms" ([Lassez, 1987], p.

Before describing our approach for extending a concept language by concrete domains we shall state *some* of the properties which such an extension should satisfy:

- The extension should still have a formal declarative semantics which is as close as possible to the usual semantics employed for concept languages.
- It should be possible to combine existing inference algorithms for concept languages with well-known reasoning algorithms in the concrete domain in order to get the appropriate algorithms for the extension.
- One should provide a scheme for extending concept languages by various concrete domains rather than constructing a single ad hoc extension for a specific concrete domain. The formal semantics as well as the combination of the algorithms should already be treated on this scheme level.

In order to satisfy these properties it is important to choose an appropriate interface between the concept language and the concrete domain. The interface which we shall use in the present paper was inspired by a construct which is e.g. present in the CLASSIC system, namely coreference constraints (also called agreements) between chains of single-valued roles (also called features).[2] With such a coreference constraint one can for example express the concept of all women whose father and husband are of the same age by the expression Woman ↤ (father age) *[* (husband age). But one cannot express that the husband is even older than the father. This becomes possible if we take the set of nonnegative integers as concrete domain. Then we can simply write Woman ⊓ ≥(husband age, father age) where > stands for the binary predicate {(n,m);n ≥ m} on nonnegative integers. More general, our extension will allow to state that feature chains satisfy a (nonnecessarily binary) predicate which is provided by the concrete domain in question.

The next section will contain a formal definition of what we mean by the notion "concrete domain". Section 3 describes our scheme for extending a concept language by an arbitrary concrete domain. As a starting point for this extension we use the language *ACC* of [Schmidt-Schaufi and Smolka, 199l]. The reason for choosing this language was that it is large enough to exhibit most of the problems connected with such an extension. Taking a larger language (e.g., including number restrictions) would only mean more work without bringing new insights. Section 4 describes how an assertional component for such an extended concept language can be defined. For both the terminological and the assertional part of our formalism we shall introduce the important inference problems. Because of the space limitations it is not possible to present the algorithm which can be used to decide all of these problems. A complete presentation of the algorithm, and a proof of its correctness can be found in [Baader and Hanschke, 1991], It is important to note that this algorithm is not only sound but also complete.[3]

2 Concrete Domains

The following definition formalizes the notion "concrete domain" which has until now only been used in an intuitive sense,

Definition 2.1 *A* concrete domain $\mathcal{D}$ *consists of a set* $dom(\mathcal{D})$, *the domain of* $\mathcal{D}$, *and a set* $pred(\mathcal{D})$, *the predicate names of* $\mathcal{D}$. *Each predicate name* P *is associated with an arity* n, *and an* n*-ary predicate* $P^{\mathcal{D}} \subseteq dom(\mathcal{D})^n$.

We shall now give some examples of concrete domains.

- In the examples of the introduction we have considered the concrete domain $\mathcal{N}$ which has the set of nonnegative integers as its domain. We have also used the binary predicate name $\geq$, and one of the unary predicate names $\geq_n$.
- The concrete domain $\mathcal{R}$ is defined as follows. The domain of $\mathcal{R}$ is the set of all real numbers, and the predicates of $\mathcal{R}$ are given by formulae which are built by first order means (i.e., by using logical connectives and quantifiers) from equalities and inequalities between integer polynomials in several indeterminates.[4] For example, $x + z^2 = y$ is an equality between the polynomials $p(x,z) = x + z^2$ and $q(y) = y$; and $x > y$ is an inequality between very simple polynomials. From these equalities and inequalities one can e.g. build the formulae $\exists z(x + z^2 = y)$ and $\exists z(x + z^2 = y) \vee (x > y)$. The first formula yields a predicate name of arity 2 (since it has two free variables), and it is easy to see that the associated predicate is $\{(r,s);\ r \text{ and } s \text{ are real numbers and } r \leq s\}$. Consequently, the predicate associated to the second formula is $\{(r,s);\ r \text{ and } s \text{ are real numbers}\} = dom(\mathcal{R}) \times dom(\mathcal{R})$.
- The runcrete domain *Z* is defined as R with the only difference that *dom(Z)* is the set of all integers instead of all real numbers.
- Our next example leaves the realm of numbers and arithmetic. Assume that D*B* is an arbitrary relational database equipped with an appropriate query

[2]Agreements on feature chains are just the restriction of Role Value Maps to single-valued (i,e., functional) roles; but unlike Role Value Maps they usually do not cause undecidability of subsumption [Hollunder and Nutt, 1990].

[3]All the above mentioned systems employ sound but incomplete algorithms.

[4]For the sake of simplicity we assume here that the formula itself is the predicate name.

language. Then $\mathcal{DB}$ can be seen as a concrete domain where $dom(\mathcal{DB})$ is the set of atomic values in the database. The predicates of $\mathcal{DB}$ are the relations which can be defined over $\mathcal{DB}$ with the help of the query language.

As mentioned in the introduction, we want to combine inference algorithms for the given concept language with reasoning algorithms for the concrete domain in order to get inference algorithms for the extended concept language. This is only possible if the concrete domain satisfies some additional properties.

For technical reasons we shall have to push negation into concept terms. To make this possible we have to require that the set of predicate names of the concrete domain is *closed under negation*, i.e., if P is an n-ary predicate name in $pred(\mathcal{D})$ then there has to exist a predicate name Q in $pred(\mathcal{D})$ such that $Q^{\mathcal{D}} = dom(\mathcal{D})^n \setminus P^{\mathcal{D}}$. In addition, we need a unary predicate name which denotes the predicate $dom(\mathcal{D})$. The domain $\mathcal{N}$ from above does not satisfy these properties. We should have to add the predicate names $<, <_n$. The domains $\mathcal{R}$ and $\mathcal{Z}$ satisfy the properties. Whether a domain of the form $\mathcal{DB}$ satisfies these properties depends on the query language.

The property which will be formulated now clarifies what kind of reasoning mechanisms are required in the concrete domain. Let P_1, ..., P_k be k (not necessarily different) predicate names in $pred(\mathcal{D})$ of arities n_1, ..., n_k. We consider the conjunction

$$\bigwedge_{i=1}^{k} P_i(\underline{x}^{(i)}).$$

Here $\underline{x}^{(i)}$ stands for an n_i-tuple $(x_1^{(i)}, ..., x_{n_i}^{(i)})$ of variables. It is important to note that neither all variables in one tuple nor those in different tuples are assumed to be distinct. Such a conjunction is said to be *satisfiable* iff there exists an assignment of elements of $dom(\mathcal{D})$ to the variables such that the conjunction becomes true in $\mathcal{D}$.

For example, let $P_1(x, y)$ be the predicate $\exists z(x + z^2 = y)$ in pred($\mathcal{R}$), and let $P_2(x, y)$ be the predicate $x > y$ in $pred(\mathcal{R})$. Obviously, neither the conjunction $P_1(x, y) \wedge P_2(x, y)$ nor $P_2(x, x)$ is satisfiable.

Definition 2.2 *A concrete domain $\mathcal{D}$ is called* admissible *iff* (i) *the set of its predicate names is closed under negation and contains a name for $dom(\mathcal{D})$, and* (ii) *the satisfiability problem for finite conjunctions of the above mentioned form is decidable.*

The concrete domain $\mathcal{R}$ is admissible. This is a consequence of Tarski's decidability result for real arithmetic [Tarski, 1951]. The concrete domain $\mathcal{Z}$ is not admissible since Hilbert's Tenth Problem—one of the most prominent undecidable problems [Matijacevič, 1970]—is a special case of its satisfiability problem.

3 The Concept Language

We shall now present our scheme for integrating an arbitrary concrete domain $\mathcal{D}$ into the concept language $\mathcal{ALC}$. The result of this integration will be called $\mathcal{ALC}(\mathcal{D})$.

3.1 Syntax and Semantics

In addition to the usual language constructs of $\mathcal{ALC}$, the language $\mathcal{ALC}(\mathcal{D})$ allows features (i.e., functional roles) in value restrictions, and predicate names of $\mathcal{D}$ applied to feature chains. For a set F of feature names, a feature chain is just a nonempty word over F.

Definition 3.1 *Let* C, R, *and* F *be disjoint sets of concept, role, and feature names. The set of* concept terms *of $\mathcal{ALC}(\mathcal{D})$ is inductively defined. As a starting point of the induction, any element of* C *is a concept term* (atomic terms). *Now let C and D be concept terms, let R be a role name or feature name, $P \in pred(\mathcal{D})$ be an n-ary predicate name, and u_1, ..., u_n be feature chains. Then the following expressions are also concept terms:*

1. $C \sqcup D$ (disjunction), $C \sqcap D$ (conjunction), *and* $\neg C$ (negation),
2. $\exists R.C$ (exists-in restriction) *and* $\forall R.C$ (value restriction),
3. $P(u_1, ..., u_n)$ (predicate restriction).

Let A be a concept name and let D be a concept term. Then $A = D$ is a terminological axiom. *A* terminology (T-box) *is a finite set $\mathcal{T}$ of terminological axioms with the additional restrictions that* (i) *no concept name appears more than once as a left hand side of a definition, and* (ii) *$\mathcal{T}$ contains no cyclic definitions.*[5]

Please note that the exists-in and the value restrictions are not only defined for roles but also for features.

The following is an example of a T-box in $\mathcal{ALC}(\mathcal{N})$. Let Human, Female, Mother, Woman be concept names, let child be a role name, and let age be a feature name. The T-box—which proposes yet another definition of the concept woman—consists of the following axioms:

$$\begin{aligned} \mathsf{Mother} &= \mathsf{Human} \sqcap \mathsf{Female} \sqcap \exists \mathsf{child}.\mathsf{Human} \\ \mathsf{Woman} &= \mathsf{Human} \sqcap \mathsf{Female} \sqcap (\mathsf{Mother} \sqcup \geq_{21}(\mathsf{age})) \end{aligned}$$

The reason for choosing child as role and age as feature is that an individual can have more than one child, but (s)he has only one age. The next definition gives a model-theoretic semantics for the languages introduced in Definition 3.1.

Definition 3.2 *An* interpretation *$\mathcal{I}$ for $\mathcal{ALC}(\mathcal{D})$ consists of a set $dom(\mathcal{I})$, the abstract domain of the interpretation, and an interpretation function. The abstract domain and the given concrete domain have to be disjoint, i.e., $dom(\mathcal{D}) \cap dom(\mathcal{I}) = \emptyset$. The interpretation function associates with each concept name A a subset $A^{\mathcal{I}}$ of $dom(\mathcal{I})$, with each role name R a binary relation $R^{\mathcal{I}}$ on $dom(\mathcal{I})$, i.e., a subset of $dom(\mathcal{I}) \times dom(\mathcal{I})$, and with each feature name f a partial function $f^{\mathcal{I}}$ from $dom(\mathcal{I})$ into $dom(\mathcal{I}) \cup dom(\mathcal{D})$.*
For such a partial function $f^{\mathcal{I}}$ the expression $f^{\mathcal{I}}(x) = y$ is sometimes written as $(x, y) \in f^{\mathcal{I}}$. If $u = f_1 ... f_n$ is a feature chain, then $u^{\mathcal{I}}$ denotes the composition $f_1^{\mathcal{I}} \circ ... \circ f_n^{\mathcal{I}}$ of the partial functions $f_1^{\mathcal{I}}, ..., f_n^{\mathcal{I}}$.[6]

[5] See [Nebel, 1989; Baader, 1990] for a treatment of cyclic definitions in concept languages.

[6] The composition should be read from left to right, i.e., $f_1^{\mathcal{I}} \circ ... \circ f_n^{\mathcal{I}}$ means apply first $f_1^{\mathcal{I}}$, then $f_2^{\mathcal{I}}$, and so on.

The interpretation function—which gives an interpretation for atomic terms—can be extended to arbitrary concept terms as follows: Let C and D be concept terms, let R be a role name or feature name, $P \in pred(\mathcal{D})$ be an n-ary predicate name, and $u_1, ..., u_n$ be feature chains. Assume that $C^{\mathcal{I}}$ and $D^{\mathcal{I}}$ are already defined. Then

1. $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, *and* $(\neg C)^{\mathcal{I}} = dom(\mathcal{I}) \setminus C^{\mathcal{I}}$,
2. $(\forall R.C)^{\mathcal{I}} = \{x \in dom(\mathcal{I});\ \forall y\colon (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$, $(\exists R.C)^{\mathcal{I}} = \{x \in dom(\mathcal{I});\ \exists y\colon (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$,
3. $P(u_1, ..., u_n)^{\mathcal{I}} = \{x \in \mathrm{dom}(\mathcal{I}); \exists r_1, ..., r_n \in \mathrm{dom}(\mathcal{D})\colon u_1^{\mathcal{I}}(x) = r_1, ..., u_n^{\mathcal{I}}(x) = r_n \ \wedge\ (r_1, ..., r_n) \in P^{\mathcal{D}}\}$.

An interpretation $\mathcal{I}$ is a model *of the T-box $\mathcal{T}$ iff it satisfies $A^{\mathcal{I}} = D^{\mathcal{I}}$ for all terminological axioms $A = D$ in $\mathcal{T}$.*

The philosophy underlying this definition is that we assume that the concrete domain *7)* is sufficiently structured by the predicates in *pred(D).* That means that we do not want to define new classes of elements of *dorn(V)* or new relations between elements of *dom(D)* with the help of our concept language. Consequently, concept terms are always interpreted as subsets of the abstract domain, i.e., an individual of the concrete domain cannot be element of a concept.

3.2 Terminological Reasoning

An important service terminological representation systems provide is computing the subsumption hierarchy, i.e., computing the subconcept-superconcept relationships between the concepts of a T-box. Let $\mathcal{T}$ be a T-box and let A, B be concept names. Then B *subsumes* A with respect to $\mathcal{T}$ (symbolically $A \sqsubseteq_{\mathcal{T}} B$) iff $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{T}$.

In our example, it is very easy to see that Woman subsumes Mother. However, in general it is not at all trivial to determine such relationships, Until recently, sound and complete subsumption algorithms were only known for rather trivial concept languages (see [Levesque and Brachman, 1987]). Consequently, all the existing KL-ONE systems use only sound, but incomplete algorithms. If such an algorithm gives a positive answer, a subsumption relationship really exists; but if its answer is negative, then we do not know anything. A subsumption relationship may or may not exist. In [Schmidt-SchauB and Smolka, 1991] a sound and complete subsumption algorithm for *ACC* is described. The underlying method of constraint propagation was used in [Hollander e*t* al,, 1990] to derive algorithms for various other concept languages. This method can—with appropriate modifications—also be applied to the languages of the form *ACC(D).* As a subtask, such an algorithm for *ACC(V)* will have *to* decide satisfiability of conjunctions of the form Ai=i Pi($x^{(i)}$) in the concrete domain. Thus we shall have to require that *V* is admissible.

In [Baadcr and Hanschke, 1991] we do not directly give a subsumption algorithm for *ACC(V),* Instead we reduce the subsumption problem to a problem which will be introduced in the next section: the consistency problem for A-boxes.

4 The Assertional Language

The terminological formalism introduced in the previous section allows to describe knowledge about classes of objects (the concepts) and relationships between these classes (e.g., subsumption relationships which are consequences of the descriptions). Many applications, however, require that one can also say something about objects in the world. For this reason, most KL-ONE systems provide additional assertional capabilities. This assertional part of the system uses the concept terms for making statements about parts of a given world. The expressiveness of this component varies between the rather weak formalism employed in the original KL-ONE system [Brachman and Schmolze, 1985] to full first order predicate logic as used in KRYPTON [Brachman *et al.,* 1985], We shall now show how to integrate a concrete domain into an assertional language which is similar to the ones used in KANDOR [Patel-Schneider, 1984], MESON [Edelmann and Owsnicki, 1986], CLASSIC [Borgida *et al.,* 1989], or BACK [Nebel and von Luck, 1988].

4.1 Syntax and Semantics

Let *V* be an arbitrary concrete domain, We have seen in Section 3 that we have to deal with two different kinds of objects: the individuals of the concrete domain and the individuals in the abstract domain (see Definition 3.2). The names for objects of the concrete domain will come from a set OC of object names, and the names for objects of the abstract domain from a set OA.

Definition 4.1 *Let* OC *and* OA *be two disjoint sets of object names. The set of all* assertional axioms *is defined as follows. Let C be a concept term of ACC(V), R be a role name, f be a feature name, P be an n-ary predicate name of D, and let a,* 6 *be elements* of OA *and y, y_1, ..., y_n be elements of OC. Then the following are assertional axioms:*

$$a : C,\quad (a,b) : R,\quad (a,b) : f,\quad (a,y) : f,\quad (y_1, ..., y_n) : P.$$

An A-box *is a finite set of such assertional axioms.*

The assertional language can for example be used to express the facts that the woman Lolita, the daughter of Humbert, is married to Vladimir, a man older than Humbert, by the assertional axioms

$$\mathsf{LOLITA} : \mathsf{Woman},\ (\mathsf{LOLITA}, \mathsf{HUMBERT}) : \mathsf{father},\ (\mathsf{LOLITA}, \mathsf{VLADIMIR}) : \mathsf{husband},\ (\mathsf{HUMBERT}, \mathsf{A1}) : \mathsf{age},\ (\mathsf{VLADIMIR}, \mathsf{A2}) : \mathsf{age},\ (\mathsf{A2}, \mathsf{A1}) : >.$$

Here LOLITA, HUMBERT, and VLADIMIR are elements of OA, and A1 and A2 are elements of OC.

It may seem to be a drawback of the above defined assertional language that it disallows the use of specific elements of $dom(\mathcal{D})$ in the assertions. For example, we are not allowed to write the axiom $(\mathsf{LOLITA}, 12) : \mathsf{age}$. However, if we have a predicate name for the singleton set $\{12\}$, say $=_{12}$, then we can express the same fact by the two axioms $(\mathsf{LOLITA}, \mathsf{A3}) : \mathsf{age}$ and $=_{12}(\mathsf{A3})$. In A-boxes of $\mathcal{ALC}(\mathcal{R})$ one can use algebraic numbers such as $\sqrt{2}$ because the corresponding singleton set $\{\sqrt{2}\}$ corresponds to a predicate name in $\mathcal{R}$, namely $(x^2 = 2) \wedge x \geq 0$.

Definition 4.2 *An* interpretation *for the assertional language is simply an interpretation for $\mathcal{ALC}(\mathcal{D})$ which, in addition, assigns an object $a^{\mathcal{I}} \in dom(\mathcal{I})$ to each object name $a \in \mathsf{OA}$, and an object $x^{\mathcal{I}} \in dom(\mathcal{D})$ to each object name $x \in \mathsf{OC}$. Such an interpretation satisfies an assertional axiom*

$$a : C \text{ iff } a^{\mathcal{I}} \in C^{\mathcal{I}}, \qquad (a,b) : R \text{ iff } (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}},$$
$$(a,b) : f \text{ iff } f^{\mathcal{I}}(a^{\mathcal{I}}) = b^{\mathcal{I}}, \quad (a,y) : f \text{ iff } f^{\mathcal{I}}(a^{\mathcal{I}}) = y^{\mathcal{I}},$$
$$(y_1, ..., y_n) : P \text{ iff } (y_1^{\mathcal{I}}, ..., y_n^{\mathcal{I}}) \in P^{\mathcal{D}}.$$

An interpretation is a model of an A-box $\mathcal{A}$ *iff it satisfies all the assertional axioms of $\mathcal{A}$, and it is a* model of an A-box $\mathcal{A}$ together with a T-box $\mathcal{T}$ *iff it is a model of $\mathcal{T}$ and a model of $\mathcal{A}$.*

The definition shows that we do not require unique names for the objects occurring in an A-box.[7] For example, assume that we have the abstract names VLADIMIR and LOLITA'S_FATHER in our A-box. As our knowledge about the world increases, we may learn that Vladimir is in fact Lolita's father. Similarly, if we introduce concrete names A1, A2 for the ages of two persons PERSON1, PERSON2 into the A-box, we do not want to assume automatically that these two numbers are different.

4.2 Assertional Reasoning

In the following, $\mathcal{A}$ will always denote an A-box, $\mathcal{T}$ a T-box, C, D concept terms, $a, b \in \mathsf{OA}$ names of abstract objects, and $x, y \in \mathsf{OC}$ names of concrete objects.

An obvious requirement on the represented knowledge is that it should not be contradictory. Otherwise, it would be useless to deduce other facts from this knowledge since logically, everything follows from an inconsistent set of assumptions. However, a given A-box together with a given T-box need not have a model. For example, an A-box containing the axioms $a : C$ and $a : \neg C$, or the axioms $(a,b) : f$, $(a,y) : f$ for a feature name f is contradictory, and thus cannot have a model.

We say that an A-box together with a T-box is *consistent* iff it has a model. Otherwise, it is called *inconsistent*.

For the above mentioned reason it is important to have an algorithm which decides consistency of a given A-box together with a T-box. In addition, it will turn out that such an algorithm can also be used to solve the other important inference problems, namely subsumption between concepts and the so-called instantiation problem.

This last problem is defined as follows. The abstract object a is an *instance* of C with respect to $\mathcal{A}$ together with $\mathcal{T}$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models of $\mathcal{A}$ together with $\mathcal{T}$.

As an example, we consider the T-box defining the concepts Mother and Woman of Section 3, and an A-box containing the axioms (LOLITA, A3) : age, $=_{12}$(A3) and LOLITA : Woman. Then LOLITA is an instance of Mother with respect to the A-box together with the T-box.

The instantiation problem can be reduced to the consistency problem as follows: a is an instance of C with respect to $\mathcal{A}$ together with $\mathcal{T}$ iff the A-box $\mathcal{A} \cup \{a : \neg C\}$ together with $\mathcal{T}$ is inconsistent.

[7] Many KL-ONE based systems have a unique name assumption for their A-box individuals; but for example KL-TWO [Vilain, 1985] does not assume unique names.

Finally, the subsumption problem can also be reduced to the instantiation problem, and thus to the consistency problem for A~boxes, In fact, *C* is subsumed by *D* with respect to T iff the abstract object a is an instance of *D* with respect to the A-box {o ; *C}* together with *T.*

In [Baader and Hanschke, 1991] a sound and complete algorithm is described which decides the consistency problem for *ALC(V),* provided that the concrete domain *V* is admissible. Such an algorithm for *ACC* without concrete domain and features can be found in [Hollunder, 1990]. Since the other inference problems introduced above can be reduced to the consistency problem, we thus have

Theorem 4.3 *Let V be an admissible concrete domain. Then there exists a sound and complete algorithm which is able to decide the following problems for ACC(V): the subsumption problem w.r.t. a T-box, the instantiation problem w.r.t. an A-box together with a T-box, and the consistency problem for an A-box together with a T-box.*

5 Conclusion

We have proposed a KL-ONE based knowledge representation and reasoning system which is hybrid in two respects. On the one hand, it makes the usual distinction between two epistemological different kinds of knowledge, the terminological knowledge and the assertional knowledge. On the other hand, the terminological and assertional language, which usually describes the knowledge on an abstract logical level, is extended by allowing to refer to concrete domains and predicates on these domains.

The different parts of the system are integrated with the help of a unified model-theoretic semantics. Reasoning in the terminological and the assertional part can be done with the help of a single basic reasoning algorithm. This algorithm creates subtasks which have to be solved by the special purpose reasoner of the concrete domain. But there is no other interaction necessary between our basic reasoning algorithm and the reasoner on the concrete domain.

Our approach differs from other extensions of KL-ONE which were done for similar reasons in several respects. Firstly, we have proposed a scheme for such an extension, and not a particular extension by some specific concrete domains. The formal semantics and the algorithm are given on this scheme level. Secondly, the basic reasoning algorithm is not only sound but also complete with respect to this semantics. In addition, we can utilize special purpose reasoners which may already exist for the concrete domain in question- This shows another difference to e.g. the MESON system where the important relationships between the user-defined or machine-defined predicates have to be explicitely supplied by the user.

An advantage other systems have compared to our approach is that their incomplete algorithms are usually polynomial, whereas our complete algorithm is of complexity at least PSPACE (depending on the concrete domain). However, one should keep in mind that these complexity results are worst case results. The algorithm may behave much better for "typical" knowledge bases.

Our main motivation for developing the presented KL-oNE extension was to represent knowledge in a mechanical engineering domain. In particular, we wanted to describe both geometric and other attributes of lathe work pieces in a unified framework. For that purpose we intend to use the language *ACC(R)* where geometric properties can be described with the help of predicates over real numbers. Currently, we are experimenting with a prototypical implementation which uses a relatively small fragment of real arithmetics as its concrete domain.

References

[Baader and Hanschke, 1991] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. Research Report RR-91-10, DFKI, Kaiserslautern, 1991.

[Baader, 1990] F. Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proceedings of ike Eighth National Conference on Artificial Intelligence,* volume 2, pages 621-626. AAAI, 1990.

[Borgida *et al,* 1989] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A structural data model foT objects. In *International Conference on Management of Data.* ACM SIGMOD, 1989.

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

[Brachman et al., 1979] R. J. Brachman, R. J. Bobrow, P. R. Cohen, J. W. Klovstad, B. L. Webber, and W. A. Woods. Research in natural language understanding, annual report. Tech. Rep. No. 4274, Cambrige, MA, 1979. Bolt Beranek and Newman.

[Brachman *et al,* 1985] R, J, Brachman, V. Pigman Gilbert, and R. J* Levesque. An essential hybrid reasoning system: knowledge and symbol level accounts in KRYPTON. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 532-539, 1985.

[Edelmann and Owsnicki, 1986] J. Edelmann and B. Owsnicki. Data models in knowledge representation systems: a case study. In GWAI-86 *und 8. Ostereichische ArtificiaUIntelligence-Tagung,* volume 124 of *Informatik-Fachberichte,* pages 69-74. Springer, 1986.

[Hollunder and Nutt, 1990] B. Hollunder and W. Nutt. Subsumption algorithms for concept languages. Research Report RR-90-04, DFKI, Kaiserslautern, 1990,

[Hollunder *et* al., 1990] B. Hollunder, W. Nutt, and M. Schmidt-SchauB Subsumption algorithms for concept description languages. In *9th European Conference on Artificial Intelligence (ECAI'90),* pages 348-353, 1990.

[Hollunder, 1990] B- Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In GWAI-90; 14*th German Workshop on Artificial Intelligence,* volume 251 of *Informatik-Fachberichte,* pages 38-47. Springer, 1990.

[Kobsa, 1989] A. Kobsa, The SB-ONE knowledge representation workbench. In *Preprints of the Workshop on Formal Aspects of Semantic Networks,* 1989. Two Harbors, Cal.

[Lassez, 1987] C. Lassez. Constraint logic programming. In *Constraint Logic Programming: A Reader,* 1987. Fourth IEEE Symposium on Logic Programming, San Francisco.

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

[Matijacevic, 1970] Y. Matijacevic. Enumerable sets are diophantine. *Soviet Math. Doklady,* 11:354-357, 1970. English translation.

[Mays *et* al., 1988] E. Mays, C. Apte, J. Griesmer, and J. Kastner. Experience with K-Rep: an object centered knowledge representation language. In *Proceedings* of IEEE CAIA-88, pages 62-67, 1988.

[Nebel and von Luck, 1988] B. Nebel and K. von Luck. Hybrid reasoning in BACK. In Z. W. Ras and L. Saitta, editors, *Methodologies for Intelligent Systems,* volume 3, pages 260-269. North-Holland, 1988.

[Nebel, 1989] B. Nebel. Terminological cycles: Semantics and computational properties. In *Proceedings of the Workshop on Format Aspects of Semantic Networks,* 1989. Two Harbors, Cal.

[Patel-Schneider *et ah,* 1990] P. F. Patel-Schneider, B. Owsnicki-Klewe, A. Kobsa, N. Guarino, R. McGregor, W. S. Mark, D. McGuiness, B. Nebel, A. Schmiedel, and J, Yen. Report on the workshop on term subsumption languages in knowledge representation. AI *Magazine,* 11(2):16~23, 1990.

[Patel-Schneider, 1984] P. F. Patel-Schneider. Small can be beautiful in knowledge representation. In *Proceedings of the* IEEE *Workshop on Principles of Knowledge-Based Systems,* pages 11-16. Denver, Colo., 1984.

[Schmidt-SchauB and Smotka, 1991] M. Schmidt-Schaufl and G. Smolka. Attributive concept descriptions with complements. To appear in Journal of Artificial Intelligence, 47, 1991.

[Schmidt-SchauB, 1989] M. Schmidt-SchauB. Subsumption in KL-ONE is undecidable. In R. J. Brachman, editor, *First International Conference on Principles of Knowledge Representation and Reasoning,* pages 421-431, 1989.

[Tarski, 1951] A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* U. of California Press. Berkley, 1951.

[Vilain, 1985] M. Vilain. The restricted language architecture of a hybrid representation system. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence,* pages 547-551, 1985.