A Terminological Knowledge Representation System with Complete Inference Algorithms

Franz Baader and Bernhard Hollunder
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Projektgruppe WINO
Postfach 2080, D-6750 Kaiserslautern, West Germany
E-mail: {baader, hollunde}@dfki.uni-kl.de

Abstract

The knowledge representation system KL-ONE first appeared in 1977. Since then many systems based on the idea of KL-ONE have been built. The formal model-theoretic semantics which has been introduced for KL-ONE languages [BL84] provides means for investigating soundness and completeness of inference algorithms. It turned out that almost all implemented KL-ONE systems such as BACK, KL-TWO, LOOM, NIKL, SB-ONE use sound but incomplete algorithms.

Until recently, sound *and* complete algorithms for the basic reasoning facilities in these systems such as consistency checking, subsumption checking (classification) and realization were only known for rather trivial languages. However, in the last two years concept languages (term subsumption languages) have been thoroughly investigated (see for example [SS88, Neb90, HNS90, DLNN91]). As a result of these investigations it is now possible to provide sound and complete algorithms for relatively large concept languages.

In this paper we describe $\mathcal{KRIS}$ which is an implemented prototype of a KL-ONE system where all reasoning facilities are realized by sound and complete algorithms. This system can be used to investigate the behaviour of sound and complete algorithms in practical applications. Hopefully, this may shed a new light on the usefulness of complete algorithms for practical applications, even if their worst case complexity is NP or worse.

$\mathcal{KRIS}$ provides a very expressive concept language, an assertional language, and sound and complete algorithms for reasoning. We have chosen the concept language such that it contains most of the constructs used in KL-ONE systems with the obvious restriction that the interesting inferences such as consistency checking, subsumption checking, and realization are decidable. The assertional language is similar to languages normally used in such systems. The reasoning component of $\mathcal{KRIS}$ depends on sound and complete algorithms for reasoning facilities such as consistency checking, subsumption checking, retrieval, and querying.

1 Introduction and Motivation

In the last decade many knowledge representation systems in the tradition of KL-ONE [BS85] have been built, for example BACK [NvL88, Neb90], CLASSIC [BBMR89], KANDOR [Pat84], KL-TWO [Vil85], KRYPTON [BPGL85], LOOM [MB87], NIKL [KBR86], SB-ONE [Kob89]. A common feature of these systems is the separation of the knowledge into a terminological part and an assertional part. Knowledge about classes of individuals and relationships between these classes is stored in the *TBox*, and knowledge concerning particular individuals can be described in the *ABox*.

The TBox formalism provides a *concept language* (or term subsumption language) for the definition of concepts and roles, where concepts are interpreted as sets of individuals and roles as binary relations between individuals. Starting with primitive concepts and roles the language formalism is used to build up more complex concepts and roles.

For example, assume that `person`, `female`, and `shy` are primitive concepts, and `child` and `female_relative` are primitive roles. Taking the connectives concept conjunction (and), disjunction (or), and negation (not) one can express "persons who are female or not shy" by

(and `person` (or `female` (not `shy`))).

Since concepts are interpreted as sets, concept conjunction can be interpreted as set intersection, concept disjunction as set union, and negation of concepts as set complement. In addition to these operations on sets one can also employ roles for the definition of new concepts. *Value restrictions* can be used for instance to describe "individuals for whom all children are female" by the expression (all `child` `female`). *Number restrictions* allow for instance to describe "individuals having at most three children" by the expression (atmost 3 `child`). Beside the above mentioned constructs there are other well-known concept-forming constructs which are available in $\mathcal{KRIS}$ (see Section 2). An example for a role-forming construct is the conjunction of roles. We can define the role (and `child` `female_relative`), which intuitively yields the role daughter. The concept language presented in the next section also provides functional roles, so-called *attributes*. These attributes are interpreted as partial functions and not as arbitrary binary relations. Natural examples for attributes may be `father` or `first_name`. An *agreement* between two attribute chains for example allows to describe "individuals whose father and grandfather have the same first name" by the expression

(equal (compose `father` `first_name`) (compose `father` `father` `first_name`)).

Interestingly, agreements between attribute chains do not make reasoning in the language undecidable [HN90], whereas agreements between arbitrary role chains cause undecidability [Sch89].

The basic reasoning facilities concerning the TBox are the determination whether a concept denotes nothing, i.e., whether a concept denotes the empty set in every interpretation, and the computation of the subsumption hierarchy. A concept C subsumes (is more general than) a concept D iff in every interpretation the set denoted by C is a superset of the set denoted by D.

The ABox formalism consists of an assertional language which allows the introduction of individuals to express facts about a concrete world. One can state that individuals are instances of concepts, and that pairs of individuals are instances of roles or attributes.

The reasoning facilities concerning both the TBox and the ABox are classified as follows. We need algorithms for inferences such as

- checking the consistency of the represented knowledge,
- given an individual of the ABox, compute the most specific concepts in the TBox this individual is instance of,
- computing all individuals of the ABox that are instances of a given concept.

The formal model-theoretic semantics which has been introduced for KL-ONE languages [BL84] provides means for investigating soundness and completeness of inference algorithms. It turned out that the above mentioned systems use sound but incomplete algorithms. If a sound but incomplete subsumption algorithm detects a subsumption relation, this relation really exists; but if it fails to recognize that a concept subsumes another one, then we do not know anything. A subsumption relation may or may not exist. Thus, the results of the algorithms only partially coincides with what the formal semantics expresses.[1]

Until recently, sound *and* complete algorithms for the above mentioned inferences and for the subsumption problem were only known for rather trivial languages which explains the use of incomplete algorithms in existing KL-ONE systems. Another argument in favour of incomplete algorithms was that for many languages the subsumption problem is at least NP-hard [LB87, Neb88]. Consequently, complete algorithms have to be intractable, whereas incomplete algorithms may still be polynomial. However, one should keep in mind that these complexity results are worst case results. It is not at all clear how complete algorithms may behave for typical knowledge bases.

In [SS88, HNS90, Hol90] it is shown how to devise sound and complete algorithms for the above mentioned inferences in various concept languages. Thus it has become possible to implement a KL-ONE system ($\mathcal{KRIS}$) which provides

- a very expressive concept language,
- powerful reasoning facilities, and
- sound and complete algorithms for these facilities.

The purpose of this paper is as follows. Firstly, we will enumerate the language constructs which are available in $\mathcal{KRIS}$, and will give a formal semantics for their meaning. We have chosen the concept language such that it contains most of the constructs used in KL-ONE systems with the obvious restriction that the interesting inferences such as consistency checking, subsumption checking, and realization are decidable. Of course, taking

[1] But see Patel-Schneider [Pat89] who uses a four-valued semantics to formally describe the behaviour of an algorithm which is incomplete w.r.t. two-valued semantics.

such a large language means that the complexity of the inference algorithms is relatively high. But $\mathcal{KRIS}$ also provides faster algorithms for certain sublanguages.[2] Secondly, we will describe the inference mechanisms provided by $\mathcal{KRIS}$. Then we will explain the principles underlying the reasoning algorithms implemented in $\mathcal{KRIS}$. Finally, we will give an overview of the implemented $\mathcal{KRIS}$ system.

2 Formalisms for Representing Knowledge

In this section we will introduce the formalisms for representing knowledge in $\mathcal{KRIS}$. In Subsection 2.1 the syntax and semantics of the concept language and the terminological axioms are presented. In Subsection 2.2 the assertional language and its semantics are introduced.

2.1 The Concept Language Underlying $\mathcal{KRIS}$

Assume that we have three disjoint alphabets of symbols, called *concept names*, *role names*, and *attribute names*. The special concept name $*\mathsf{top}*$ is called *top concept*.

The sets of *concept terms*, *role terms*, and *attribute terms* are inductively defined as follows. Every concept name is a concept term, every role name is a role term, and every attribute name is an attribute term. Now let C, $C_1, \ldots, C_k$ be concept terms, R, $R_1, \ldots, R_l$ be role terms, f, g, $f_1, \ldots, f_m$ be attribute terms already defined, and let n be a nonnegative integer. Then

(and $C_1 \ldots C_k$),	(conjunction)
(or $C_1 \ldots C_k$),	(disjunction)
(not C),	(negation)
(all R C), (all f C),	(value restriction)
(some R C), (some f C),	(exists restriction)
(atleast n R) (atmost n R)	(number restrictions)
(equal f g),	(agreement)
(not-equal f g)	(disagreement)

are concept terms,

(and $R_1 \ldots R_l$),	(role conjunction)

is a role term, and

[2] That coincides with what Ramesh Patil proposed at the Workshop on Term Subsumption Languages in Knowledge Representation: "He therefore strongly opposed any attempt to further restrict the expressiveness of TSL (term subsumption language) systems. Instead, he proposed that such systems be configured on a "pay as you go" basis—if the application uses only a small portion of the expressive power of the TSL, then everything will be fast; if more expressive power is used, then the system may slow down, but still be able to represent and reason with the knowledge given to it." (see [PSOK+90]).

(and $f_1 \ldots f_m$),	(attribute conjunction)
(compose $f_1 \ldots f_m$)	(composition)

are attribute terms.

So-called *terminological axioms* are used to introduce names for concept, role, and attribute terms. A finite set of such axioms satisfying certain restrictions is called a terminology (TBox). There are three different ways of introducing new concepts (respectively roles or attributes) into a terminology.

Let A (P, f) be a concept (role, attribute) name, and let C (R, g) be a concept (role, attribute) term. By the terminological axioms

$$(\mathsf{defprimconcept}\ A), \quad (\mathsf{defprimrole}\ P), \quad (\mathsf{defprimattribute}\ f)$$

new concept, role, and attribute names are introduced without restricting their interpretation. The terminological axioms

$$(\mathsf{defprimconcept}\ A\ C), \quad (\mathsf{defprimrole}\ P\ R), \quad (\mathsf{defprimattribute}\ f\ g)$$

impose necessary conditions on the interpretation of the introduced concept, role, and attribute names. Finally, one can impose necessary and sufficient conditions by the terminological axioms

$$(\mathsf{defconcept}\ A\ C), \quad (\mathsf{defrole}\ P\ R), \quad (\mathsf{defattribute}\ f\ g).$$

A *terminology* (*TBox*) $\mathcal{T}$ is a finite set of terminological axioms with the additional restriction that (*i*) every concept, role, and attribute name may appear at most once as a first argument of a terminological axiom in $\mathcal{T}$ (unique definition), and (*ii*) $\mathcal{T}$ must not contain cyclic definitions[3] (acyclicity).

A terminology which describes knowledge about persons and relationships between persons is shown in Figure 1. At first, the attribute `sex` and the concept `male` is introduced. The axioms which define the concepts `female` and `person` can be read as follows: "no individual is both male and female"[4], and "a person has sex male or female." These axioms impose necessary conditions on the interpretation of the introduced concepts. The definition of the concept `parent` impose necessary and sufficient conditions: "an individual is a parent if and only if it is a person and has some child who is a person." The other concepts are also defined according to their intuitive meaning.

We will now give a formal model-theoretic semantics for the concept language and the terminological axioms. An *interpretation* $\mathcal{I}$ consists of a set $\Delta^{\mathcal{I}}$ (the *domain* of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ (the *interpretation function* of $\mathcal{I}$). The interpretation function maps every concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, every role name P to a subset $P^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and every attribute name f to a partial function $f^{\mathcal{I}}$ from $\Delta^{\mathcal{I}}$ to $\Delta^{\mathcal{I}}$. With $dom\, f^{\mathcal{I}}$ we

[3] For a discussion of terminological cycles see [Neb88, Baa90a].

[4] It might seem to be more convenient to allow explicit disjointness axioms for expressing such facts. In fact, we could easily provide such axioms at the user interface because they can be simulated by the constructs available in our language [Neb90, Baa90b].

```
(defprimattribute sex)
(defprimconcept male)
(defprimconcept female (not male))
(defprimconcept person (some sex (or male female)))
(defprimrole child)
(defconcept parent (and person (some child person)))
(defconcept mother (and parent (some sex female)))
(defconcept father (and parent (not mother)))
(defconcept grandparent (and parent (some child parent)))
(defconcept parent_with_two_children (and parent (atleast 2 child)))
(defconcept parent_with_sons_only (and parent (all child (some sex male))))
```

Figure 1: A terminology (TBox).

denote the domain of the partial function $f^{\mathcal{I}}$ (i.e., the set of elements of $\Delta^{\mathcal{I}}$ for which $f^{\mathcal{I}}$ is defined).

The interpretation function—which gives an interpretation for concept, role, and attribute names—can be extended to concept, role, and attribute terms as follows. Let C, $C_1, \ldots, C_k$ be concept terms, R, $R_1, \ldots, R_l$ role terms, f, g, $f_1, \ldots, f_m$ attribute terms, and let n be a nonnegative integer. Assume that $C^{\mathcal{I}}$, $C_1^{\mathcal{I}}, \ldots, C_k^{\mathcal{I}}$, $R^{\mathcal{I}}$, $R_1^{\mathcal{I}}, \ldots, R_l^{\mathcal{I}}$, $f^{\mathcal{I}}$, $g^{\mathcal{I}}$, $f_1^{\mathcal{I}}, \ldots, f_m^{\mathcal{I}}$ are already defined. Then

$$
\begin{array}{rcl}
(*\mathsf{top}*)^{\mathcal{I}} & := & \Delta^{\mathcal{I}} \\
(\mathsf{and}\ C_1 \ldots C_k)^{\mathcal{I}} & := & C_1^{\mathcal{I}} \cap \ldots \cap C_k^{\mathcal{I}} \\
(\mathsf{or}\ C_1 \ldots C_k)^{\mathcal{I}} & := & C_1^{\mathcal{I}} \cup \ldots \cup C_k^{\mathcal{I}} \\
(\mathsf{not}\ C)^{\mathcal{I}} & := & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\mathsf{all}\ R\ C)^{\mathcal{I}} & := & \{\ a \in \Delta^{\mathcal{I}} \mid \forall b : (a,b) \in R^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}}\} \\
(\mathsf{all}\ f\ C)^{\mathcal{I}} & := & \{\ a \in \Delta^{\mathcal{I}} \mid a \in dom\, f^{\mathcal{I}} \Rightarrow f^{\mathcal{I}}(a) \in C^{\mathcal{I}}\} \\
(\mathsf{some}\ R\ C)^{\mathcal{I}} & := & \{\ a \in \Delta^{\mathcal{I}} \mid \exists b : (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \\
(\mathsf{some}\ f\ C)^{\mathcal{I}} & := & \{\ a \in dom\, f^{\mathcal{I}} \mid f^{\mathcal{I}}(a) \in C^{\mathcal{I}}\} \\
(\mathsf{atleast}\ n\ R)^{\mathcal{I}} & := & \{a \in \Delta^{\mathcal{I}} \mid |\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\}| \geq n\} \\
(\mathsf{atmost}\ n\ R)^{\mathcal{I}} & := & \{a \in \Delta^{\mathcal{I}} \mid |\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\}| \leq n\} \\
(\mathsf{equal}\ f\ g)^{\mathcal{I}} & := & \{\ a \in dom\, f^{\mathcal{I}} \cap dom\, g^{\mathcal{I}} \mid f^{\mathcal{I}}(a) = g^{\mathcal{I}}(a)\} \\
(\mathsf{not\text{-}equal}\ f\ g)^{\mathcal{I}} & := & \{\ a \in dom\, f^{\mathcal{I}} \cap dom\, g^{\mathcal{I}} \mid f^{\mathcal{I}}(a) \neq g^{\mathcal{I}}(a)\} \\
(\mathsf{and}\ R_1 \ldots R_l)^{\mathcal{I}} & := & R_1^{\mathcal{I}} \cap \ldots \cap R_l^{\mathcal{I}} \\
(\mathsf{and}\ f_1 \ldots f_m)^{\mathcal{I}} & := & f_1^{\mathcal{I}} \cap \ldots \cap f_m^{\mathcal{I}} \\
(\mathsf{compose}\ f_1 \ldots f_m)^{\mathcal{I}} & := & f_1^{\mathcal{I}} \circ \ldots \circ f_m^{\mathcal{I}},
\end{array}
$$

where $|X|$ denotes the cardinality of the set X and $\circ$ denotes the composition of functions. The composition should be read from left to right, i.e., $f_1^{\mathcal{I}} \circ \ldots \circ f_m^{\mathcal{I}}$ means that $f_1^{\mathcal{I}}$ is applied first, then $f_2^{\mathcal{I}}$, and so on. Note, that if $f_1^{\mathcal{I}}, \ldots, f_m^{\mathcal{I}}$ are partial functions, then $f_1^{\mathcal{I}} \cap \ldots \cap f_m^{\mathcal{I}}$ and $f_1^{\mathcal{I}} \circ \ldots \circ f_m^{\mathcal{I}}$ are also partial functions.

The semantics of terminological axioms is now defined as follows. An interpretation

$\mathcal{I}$ *satisfies* the terminological axiom

$$\begin{array}{lll}
(\mathsf{defprimconcept}\ A\ C) & \text{iff} & A^{\mathcal{I}} \subseteq C^{\mathcal{I}}, \\
(\mathsf{defconcept}\ A\ C) & \text{iff} & A^{\mathcal{I}} = C^{\mathcal{I}}, \\
(\mathsf{defprimrole}\ P\ R) & \text{iff} & P^{\mathcal{I}} \subseteq R^{\mathcal{I}}, \\
(\mathsf{defrole}\ P\ R) & \text{iff} & P^{\mathcal{I}} = R^{\mathcal{I}}, \\
(\mathsf{defprimattribute}\ f\ g) & \text{iff} & f^{\mathcal{I}} \subseteq g^{\mathcal{I}}, \\
(\mathsf{defattribute}\ f\ g) & \text{iff} & f^{\mathcal{I}} = g^{\mathcal{I}},
\end{array}$$

where A (P, f) is a concept (role, attribute) name, and C (R, g) is a concept (role, attribute) term. Note that the terminological axioms (defprimconcept A), (defprimrole P), and (defprimattribute f) are satisfied in every interpretation by the definition of interpretation. An interpretation $\mathcal{I}$ is a *model* for a TBox $\mathcal{T}$ iff $\mathcal{I}$ satisfies all terminological axioms in $\mathcal{T}$.

2.2 Assertions

The assertional formalism allows to introduce individuals (objects). We can describe a concrete world by stating that individuals are instances of concepts, and that pairs of individuals are instances of roles or attributes.

Assume that we have a further alphabet of symbols, called *individual names*. Names for individuals are introduced by *assertional axioms* which have the form

$$(\mathsf{assert\text{-}ind}\ a\ C), \quad (\mathsf{assert\text{-}ind}\ a\ b\ R), \quad (\mathsf{assert\text{-}ind}\ a\ b\ g),$$

where a, b are individual names, and C (R, g) is a concept (role, attribute) term. A *world description* (*ABox*) is a finite set of assertional axioms.

Figure 2 shows an example of an ABox. This ABox describes a world in which `Tom` is

```
(assert-ind Tom father)
(assert-ind Tom Peter child)                    (assert-ind Tom Harry child)
(assert-ind Mary parent_with_sons_only)
(assert-ind Mary Tom child)                     (assert-ind Mary Chris child)
```

Figure 2: A world description (ABox).

father of `Peter` and `Harry`. Furthermore, `Mary` has only sons; two of them are `Tom` and `Chris`.

Note that an ABox can be considered as a relational database where the arity of each tuple is either one or two. However, in contrast to the closed world semantics which is usually employed in databases, we assume an *open world semantics*, since we want to allow for incomplete knowledge. Thus, we cannot conclude in the above example that `Tom` has exactly two children, since there may exist a world in which `Tom` has some additional children.

The semantics of individual names and assertional axioms is defined as follows. The interpretation function $\cdot^{\mathcal{I}}$ of a TBox interpretation $\mathcal{I}$ can be extended to individual names

by mapping them to elements of the domain such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$. This restriction on the interpretation function ensures that individuals with different names denote different individuals in the world. It is called *unique name assumption*, which is usually also assumed in the database world.

Let a, b be individual names, and C (R, g) be a concept (role, attribute) term. An interpretation $\mathcal{I}$ *satisfies* the assertional axiom

(assert-ind a C)	iff	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
(assert-ind a b R)	iff	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
(assert-ind a b f)	iff	$f^{\mathcal{I}}(a^{\mathcal{I}}) = b^{\mathcal{I}}$.

The semantics of an ABox together with a TBox is defined as follows. We say that an interpretation $\mathcal{I}$ is a *model* for an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ if $\mathcal{I}$ satisfies all assertional axioms in $\mathcal{A}$ and all terminological axioms in $\mathcal{T}$.

3 Reasoning

In this section we describe the inference mechanisms provided by $\mathcal{KRIS}$. The reasoning component of $\mathcal{KRIS}$ allows one to make knowledge explicit which is only implicitly represented in an ABox and a TBox. For example, from the TBox and Abox given in the previous section one can conclude that `Mary` is a grandparent, though this knowledge is not explicitly stored in the ABox.

An obvious requirement on the represented knowledge is that it should be consistent since everything would be deducible from inconsistent knowledge (from a logical point of view). If, for example, an ABox contains the axioms (assert-ind `Chris mother`) and (assert-ind `Chris father`), then the system should detect this inconsistency.[5] The underlying model-theoretic semantics allows a clear and intuitive definition of consistency. We say that an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ is *consistent* if it has a model. Thus, we have the

Consistency problem of an ABox $\mathcal{A}$ w.r.t. a Tbox $\mathcal{T}$: Does there exist a model for $\mathcal{A}$ w.r.t. $\mathcal{T}$?

In order to devise an algorithm which decides consistency of an ABox w.r.t. a TBox, it is appropriate to reduce this problem to a consistency problem of an ABox w.r.t. the empty TBox, i.e., a TBox that does not contain any terminological axiom. The idea behind the reduction is to enlarge the ABox by the facts expressed in the TBox. More precisely, we apply the following *expansion procedure*.

1. *Elimination of partial definitions in $\mathcal{T}$:* Any partial definition (i.e., a terminological axiom with keyword defprimconcept, defprimrole, or defprimattribute followed by two arguments) occurring in $\mathcal{T}$ is replaced by a complete definition (i.e., a terminological axiom with keyword defconcept, defrole, or defattribute). For example, the partial concept definition

(defprimconcept `female` (not `male`))

[5] However, in general it is not always as easy as in this example to check whether the represented knowledge is consistent.

is replaced by

(defconcept `female` (and (not `male`) `female`$*$))

where the newly introduced concept name `female`$*$ stands for the absent part of the definition of `female`. In a similar way partial role and attribute definitions are replaced by complete definitions. Let $\mathcal{T}'$ be the TBox which is obtained from $\mathcal{T}$ by replacing all partial definitions by complete definitions.
2. *Expansion of $\mathcal{T}'$:* Every defined concept, role, and attribute name (i.e., the first argument of a complete definition) which occurs in the defining term of a concept, role, or attribute definition (i.e., in the second argument of a complete definition) is substituted by its defining term. This process is iterated until there remain only undefined concept, role, and attribute names in the second arguments of definitions. This yields a TBox $\mathcal{T}''$.
3. *Expansion of $\mathcal{A}$:* Every concept, role, and attribute name occurring in $\mathcal{A}$ which is defined in $\mathcal{T}''$ is substituted by its defining term in $\mathcal{T}''$.

This transformation has the nice property that it is consistency preserving. That means that an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ is consistent if and only if the ABox which is obtained from $\mathcal{A}$ and $\mathcal{T}$ by applying the expansion procedure is consistent. Thus the above defined consistency problem can be reduced to the

Consistency problem of an ABox $\mathcal{A}$: Does there exist a model for $\mathcal{A}$?

Beside an algorithm for checking the consistency of an ABox $\mathcal{KRIS}$ provides algorithms for the basic reasoning facilities such as subsumption and instantiation. Let A, B be defined concepts in a TBox $\mathcal{T}$. We say that A *subsumes* B *in* $\mathcal{T}$ iff for every model $\mathcal{I}$ of $\mathcal{T}$ we have $A^{\mathcal{I}} \supseteq B^{\mathcal{I}}$. Thus, given a TBox $\mathcal{T}$ and two defined concepts A, B we have the

Subsumption problem w.r.t. a TBox $\mathcal{T}$: Does A subsume B in $\mathcal{T}$?

The subsumption problem w.r.t. a TBox $\mathcal{T}$ can be reduced to the subsumption problem of concept terms. For two concept terms C, D we say that C *subsumes* D if and only if $C^{\mathcal{I}} \supseteq D^{\mathcal{I}}$ in every interpretation $\mathcal{I}$. Let $\mathcal{T}''$ be the TBox which is obtained from $\mathcal{T}$ by applying the first two steps of the expansion procedure. Assume that C and D are the definitions of the defined concepts A and B in $\mathcal{T}''$. Then A subsumes B in $\mathcal{T}$ if and only if the concept term C subsumes the concept term D. Thus the subsumption problem w.r.t. a TBox $\mathcal{T}$ can be reduced to the

Subsumption problem: Does a concept term C subsume a concept term D ?

The subsumption problem in concept languages has been thoroughly investigated in [SS88, HNS90, DLNN91]. In these papers, subsumption algorithms for various concept languages and sublanguages are given and their computational complexity is discussed. In fact, the papers do not directly describe subsumption algorithms but algorithms for a closely related problem—the so-called *satisfiability problem* of concepts. These algorithms check whether a given concept term C is satisfiable, i.e., whether there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$. Since C subsumes D if and only if (and D (not C)) is not satisfiable, satisfiability algorithms can also be used to decide subsumption.

An algorithm for instantiation decides whether an assertional axiom is deducible from the represented knowledge. More formally, let α be an assertional axiom. We say that an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ *implies* α iff all models of $\mathcal{A}$ w.r.t. $\mathcal{T}$ satisfy α, written $\mathcal{A}, \mathcal{T} \models \alpha$. Thus we define the

Instantiation problem: Is α implied by $\mathcal{A}$ and $\mathcal{T}$?

If α is of the form (assert-ind a b R) or (assert-ind a b f), then it is relatively easy to solve the instantiation problem since the concept language contains only few constructs for building complex role or attribute terms. If α is of the form (assert-ind a C), the instantiation problem can be reduced to the consistency problem as follows:

$$\mathcal{A}, \mathcal{T} \models (\text{assert-ind } a\ C) \quad \text{iff} \quad \mathcal{A} \cup \{(\text{assert-ind } a\ (\text{not } C))\} \text{ is not consistent w.r.t. } \mathcal{T}.$$

In [Hol90], a sound and complete algorithm for the consistency and instantiation problem for a sublanguage of the language defined in Section 2 is described.

KRIS also provides the user with algorithms which find out certain relationships between the defined concepts, roles, attributes, and individuals. These algorithms are based on the algorithms for subsumption and instantiation. Assume that $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox.

The *subsumption hierarchy* is the preordering of the concept names in $\mathcal{T}$ w.r.t. the subsumption relation. The so-called *classifier* has to solve the

Classification problem: Compute the subsumption hierarchy.

Given an individual in $\mathcal{A}$, one wants to know the set of concept names in $\mathcal{T}$ which describe it most accurately. To be more formal, let a be an individual occurring in $\mathcal{A}$. The set of *most specialized concepts* for a is a set $\{A_1, \ldots, A_n\}$ of concept names occurring in $\mathcal{T}$ such that

1. $\mathcal{A}, \mathcal{T} \models (\text{assert-ind } a\ A_i)$ for every i, $1 \leq i \leq n$,
2. for every i, $1 \leq i \leq n$, there does not exist a concept name A in $\mathcal{T}$ such that $\mathcal{A}, \mathcal{T} \models (\text{assert-ind } a\ A)$, A_i subsumes A, and A and A_i are different names, and
3. for every concept name A in $\mathcal{T}$ such that $\mathcal{A}, \mathcal{T} \models (\text{assert-ind } a\ A)$, there exists an A_i such that A subsumes A_i.

The first condition means that each A_i is in fact a description of a. The second condition guarantees that the set contains only the minimal descriptions w.r.t. the subsumption relation, and the third condition means that we do not omit any nonredundant description. Thus, to describe an individual most accurately we need an algorithm for the

Realization problem: Compute for an individual in $\mathcal{A}$ the set of most specialized concepts in $\mathcal{T}$.

Conversely, one may want to know the individuals of $\mathcal{A}$ which are instances of a given concept term. Let C be a concept term. The set $INST(C)$ contains all the individuals $a_1, \ldots, a_n$ of $\mathcal{A}$ such that $\mathcal{A}, \mathcal{T} \models (\mathsf{assert\text{-}ind}\ a_i\ C)$ holds. Thus we also have the

Retrieval problem: Compute for a given concept term C the set $INST(C)$.

4 The Basic Reasoning Algorithms

In this section we will explain the principles underlying the reasoning algorithms implemented in $\mathcal{KRIS}$. To this purpose, we restrict our attention to the concept language $\mathcal{ALC}$ of Schmidt-Schauß and Smolka [SS88] which allows one to use concept names, role names, and the concept forming constructs conjunction, disjunction, negation, value restriction and exists restriction. The language $\mathcal{ALC}$ is only a sublanguage of the actual concept language available in our system, but it is large enough to demonstrate the principle problems one has to overcome when devising sound and complete algorithms for terminological KR-systems. Algorithms for various other concept languages can e.g. be found in [DLNN91, HB91, HNS90].

In the previous section we have shown that it is enough to have algorithms which test for satisfiability of concept terms and for consistency of ABoxes since all the other introduced reasoning problems can be reduced to these two problems. We will first illustrate by an example how satisfiability can be checked for concept terms of $\mathcal{ALC}$. We will then describe an algorithm which is more appropriate for an implementation than the original one given in [SS88]. Finally, it will be explained how the ideas underlying the satisfiability algorithm can be generalized to consistency checking for ABoxes.

4.1 An Example for the Satisfiability Test for $\mathcal{ALC}$

Assume that C is a concept term of $\mathcal{ALC}$ which has to be checked for satisfiability. In a first step we can push all negations as far as possible into the term using the fact that the terms $(\mathsf{not}\ (\mathsf{not}\ D))$ and D, $(\mathsf{not}\ (\mathsf{and}\ D\ E)$ and $(\mathsf{or}\ (\mathsf{not}\ D)\ (\mathsf{not}\ E))$, $(\mathsf{not}\ (\mathsf{or}\ D\ E)$ and $(\mathsf{and}\ (\mathsf{not}\ D)\ (\mathsf{not}\ E))$, $(\mathsf{not}\ (\mathsf{all}\ R\ D))$ and $(\mathsf{some}\ R\ (\mathsf{not}\ D))$, as well as $(\mathsf{not}\ (\mathsf{some}\ R\ D))$ and $(\mathsf{all}\ R\ (\mathsf{not}\ D))$ are equivalent, that is, they denote the same set in every interpretation. We end up with a term C' in negation normal form where negation is only applied to concept names.

Example 4.1 Let A, B be concept names, and let R be a role name. Assume that we want to know whether $(\mathsf{and}\ (\mathsf{some}\ R\ A)\ (\mathsf{some}\ R\ B))$ is subsumed by $(\mathsf{some}\ R\ (\mathsf{and}\ A\ B))$. That means that we have to check whether the term

$$C := (\mathsf{and}\ (\mathsf{some}\ R\ A)\ (\mathsf{some}\ R\ B)\ (\mathsf{not}\ (\mathsf{some}\ R\ (\mathsf{and}\ A\ B))))$$

is not satisfiable. The negation normal form of C is the term

$$C' := (\mathsf{and}\ (\mathsf{some}\ R\ A)\ (\mathsf{some}\ R\ B)\ (\mathsf{all}\ R\ (\mathsf{or}\ (\mathsf{not}\ A)\ (\mathsf{not}\ B)))).$$

In a second step we try to construct a finite interpretation $\mathcal{I}$ such that $C'^{\mathcal{I}} \neq \emptyset$. That means that there has to exist an individual in $\Delta^{\mathcal{I}}$ which is an element of $C'^{\mathcal{I}}$. Thus the algorithm generates such an individual b, and imposes the constraint $b \in C'^{\mathcal{I}}$ on it. In the example, this means that b has to satisfy the following constraints: $b \in (\mathsf{some}\ R\ A)^{\mathcal{I}}$, $b \in (\mathsf{some}\ R\ B)^{\mathcal{I}}$, and $b \in (\mathsf{all}\ R\ (\mathsf{or}\ (\mathsf{not}\ A)\ (\mathsf{not}\ B)))^{\mathcal{I}}$.

From $b \in (\mathsf{some}\ R\ A)^{\mathcal{I}}$ we can deduce that there has to exist an individual c such that $(b,c) \in R^{\mathcal{I}}$ and $c \in A^{\mathcal{I}}$. Analogously, $b \in (\mathsf{some}\ R\ B)^{\mathcal{I}}$ implies the existence of an individual d with $(b,d) \in R^{\mathcal{I}}$ and $d \in B^{\mathcal{I}}$. We should not assume that $c = d$ since this would possibly impose too many constraints on the individuals newly introduced to satisfy the exists restrictions on b. Thus the algorithm introduces for any exists restriction a new individual as role-successor, and this individual has to satisfy the constraints expressed by the restriction.

Since b also has to satisfy the value restriction (all R (or (not A) (not B))), and c, d were introduced as $R^{\mathcal{I}}$-successors of b, we also get the constraints $c \in (\mathsf{or}\ (\mathsf{not}\ A)\ (\mathsf{not}\ B))^{\mathcal{I}}$, and $d \in (\mathsf{or}\ (\mathsf{not}\ A)\ (\mathsf{not}\ B))^{\mathcal{I}}$. Now c has to satisfy the constraints $c \in A^{\mathcal{I}}$ and $c \in (\mathsf{or}\ (\mathsf{not}\ A)\ (\mathsf{not}\ B))^{\mathcal{I}}$, whereas d has to satisfy the constraints $d \in B^{\mathcal{I}}$ and $d \in (\mathsf{or}\ (\mathsf{not}\ A)\ (\mathsf{not}\ B))^{\mathcal{I}}$. Thus the algorithm uses value restrictions in interaction with already defined role relationships to impose new constraints on individuals.

Now $c \in (\mathsf{or}\ (\mathsf{not}\ A)\ (\mathsf{not}\ B))^{\mathcal{I}}$ means that $c \in (\mathsf{not}\ A)^{\mathcal{I}}$ or $c \in (\mathsf{not}\ B)^{\mathcal{I}}$, and we have to choose one of these possibilities. If we assume $c \in (\mathsf{not}\ A)^{\mathcal{I}}$, this clashes with the other constraint $c \in A^{\mathcal{I}}$. Thus we have to choose $c \in (\mathsf{not}\ B)^{\mathcal{I}}$. Analogously, we have to choose $d \in (\mathsf{not}\ A)^{\mathcal{I}}$ in order to satisfy the constraint $d \in (\mathsf{or}\ (\mathsf{not}\ A)\ (\mathsf{not}\ B))^{\mathcal{I}}$ without creating a contradiction to $d \in B^{\mathcal{I}}$. Thus, for disjunctive constraints, the algorithm tries both possibilities in successive attempts. It has to backtrack if it reaches a contradiction, i.e., if the same individual has to satisfy conflicting constraints.

In the example, we have now satisfied all the constraints without getting a contradiction. This shows that C' is satisfiable, and thus $(\mathsf{and}\ (\mathsf{some}\ R\ A)\ (\mathsf{some}\ R\ B))$ is not subsumed by $(\mathsf{some}\ R\ (\mathsf{and}\ A\ B))$. We have generated an interpretation $\mathcal{I}$ as witness for this fact: $\Delta^{\mathcal{I}} = \{b,c,d\}$; $R^{\mathcal{I}} := \{(b,c),(b,d)\}$; $A^{\mathcal{I}} := \{c\}$ and $B^{\mathcal{I}} := \{d\}$. For this interpretation, $b \in C'^{\mathcal{I}}$. That means that $b \in (\mathsf{and}\ (\mathsf{some}\ R\ A)\ (\mathsf{some}\ R\ B))^{\mathcal{I}}$, but $b \notin (\mathsf{some}\ R\ (\mathsf{and}\ A\ B))^{\mathcal{I}}$.

Termination of the algorithm is ensured by the fact that the newly introduced constraints are always smaller than the constraints which enforced their introduction.

4.2 A Rule-Based and a Functional Algorithm for Satisfiability

In this subsection, we will first give a more formal description of the algorithm sketched in the previous section. We will then show how this rule-based algorithm can be modified to a functional algorithm which is more appropriate for implementation purposes.

Let C_0 be a concept term of $\mathcal{ALC}$. Without loss of generality we assume that C_0 is in negation normal form. In principle, the algorithm starts with the set $S_0 := \{b_0 \in C_0^{\mathcal{I}}\}$ of constraints, and transform it with the help of certain rules until one of the following two situations occurs: *(i)* the obtained set of constraints is "obviously contradictory",

or *(ii)* the obtained set of constraints is "complete", i.e., one can apply no more rules. In the second case, the complete set of constraints describes an interpretation $\mathcal{I}$ with $C_0^{\mathcal{I}} \neq \emptyset$. For the language $\mathcal{ALC}$, a set of constraints is obviously contradictory iff it contains conflicting constraints of the form $c \in A^{\mathcal{I}}$, $c \in (\mathsf{not}\ A)^{\mathcal{I}}$ for some individual c and concept name A. Please note that such contadictions can only occur between two constraints imposed on the same individual c.

Because of the presence of disjunction in our language, a given set of constraints must sometimes be transformed into two different new sets. For that reason, we will work with sets $\mathcal{M}$ of sets of constraints rather than with a single set of constraints. If we want to test C_0 for satisfiability, we start with the singleton set $\mathcal{M}_0 := \{\{b_0 \in C_0^{\mathcal{I}}\}\}$.

Let $\mathcal{M}$ be a finite set of sets of constraints, and let S be an element of $\mathcal{M}$. The following rules will replace S by a set S' or by two sets S' and S'':

1. *The conjunction rule.* Assume that $c \in (\mathsf{and}\ C_1\ C_2)^{\mathcal{I}}$ is in S, and $c \in C_1^{\mathcal{I}}$ or $c \in C_2^{\mathcal{I}}$ is not in S. The set of constaints S' is obtained from S by adding $c \in C_1^{\mathcal{I}}$ and $c \in C_2^{\mathcal{I}}$ to S.

2. *The disjunction rule.* Assume that $c \in (\mathsf{or}\ C_1\ C_2)^{\mathcal{I}}$ is in S, and neither $c \in C_1^{\mathcal{I}}$ nor $c \in C_2^{\mathcal{I}}$ is in S. The set of constraints S' is obtained from S by adding $c \in C_1^{\mathcal{I}}$ to S, and the set of constraints S'' is obtained from S by adding $c \in C_2^{\mathcal{I}}$to S.

3. *The exists restriction rule.* Assume that $c \in (\mathsf{some}\ R\ D)^{\mathcal{I}}$ is in S, and there is no individual e such that $(c, e) \in R^{\mathcal{I}}, e \in D^{\mathcal{I}}$ are in S. Then we create a new individual d, and add the constraints $(c, d) \in R^{\mathcal{I}}, d \in D^{\mathcal{I}}$ to S.

4. *The value restriction rule.* Assume that $c \in (\mathsf{all}\ R\ D)^{\mathcal{I}}$ and $(c, d) \in R^{\mathcal{I}}$ are in S, and that $d \in D^{\mathcal{I}}$ is not in S. Then the set of constraints S' is obtained from S by adding $d \in D^{\mathcal{I}}$.

It can be shown that there cannot be an infinite chain of sets $\mathcal{M}_0$, $\mathcal{M}_1$, $\mathcal{M}_2$, ... where each $\mathcal{M}_{i+1}$ is obtained from $\mathcal{M}_i$ by application of one of the above defined rules. Thus if we start with a set $\mathcal{M}_0 = \{\{b_0 \in C_0^{\mathcal{I}}\}\}$, and apply rules as long as possible, we finally end up with a complete set M_r, i.e., a set to which no more rules are applicable. Now C_0 is satisfiable iff there exists a set of constraints in $\mathcal{M}_r$ which is not obviously contradictory.

Please note that this fact is independent of the order in which the rules have been applied. By using appropriate strategies, one may get optimized versions of the algorithm. We shall now sketch how an algorithm can be derived which no longer depends on an explicit representation of individuals and role relationships between individuals. Until now, such an explicit representation is necessary for the following two reasons. First, we need the individual names to detect which constraints are obviously contradictory. Second, the explicit representation of role relationships is necessary to show for what other individuals d a constraint of the form $c \in (\mathsf{all}\ R\ D)^{\mathcal{I}}$ yields a new constraint $d \in D^{\mathcal{I}}$.

In order to explain the ideas underlying our optimized algorithm, we first analyse from which sources constraints for a given individual c may come. On the one hand, application of a conjunction or disjunction rule to a constraint on c itself may yield a new constraint

on c. On the other hand, a constraint on c may come from an other individual b when the exists or value restriction rule is applied to b. Please note that in this case c is a role successor of b for some role R, and that there can be at most one such individual b for a given c. There is one exception to this second case. The individual b_0 we start with does not have a role predecessor, but it has the original constraint $b_0 \in C_0^{\mathcal{I}}$.

Assume that we start with the original constraint $b_0 \in C_0^{\mathcal{I}}$. By applying the conjunction and disjunction rule to the constraints on b_0 as long as possible, we obtain all possible constraints on b_0. This means that we can now detect all possible obvious contradictions caused by constraints on b_0. Since all exists restrictions for b_0 are already present, we know how many new individuals we have to introduce as role successors of b_0, and since all the value restrictions on b_0 are already present, we also know exactly which constraints are propagated from b_0 to these successors. Obviously, if we have the exists restriction $b_0 \in (\mathsf{some}\ R\ D)^{\mathcal{I}}$, and $b_0 \in (\mathsf{all}\ R\ E_1)^{\mathcal{I}}, \ldots, b_0 \in (\mathsf{all}\ R\ E_k)^{\mathcal{I}}$ are all the value restrictions imposed on b_0 w.r.t. the role R, then the individual c which is created because of this exists restriction has to satisfy the constraints $c \in D^{\mathcal{I}}, c \in E_1^{\mathcal{I}}, \ldots, c \in E_k^{\mathcal{I}}$.

After imposing these constraints on c, all the constraints coming from its unique role predecessor b_0 are already present in the actual constraint system. In this case, one can forget the role relationship between b_0 and c because it no longer yields new constraints on c. Since there is no more interaction between constraints on c and constraints on other individuals, one can test the satisfiability of the constraints on c independently from all the other constraints in our system. This means that we may now continue with c in place of b_0, i.e., first the apply conjunction and disjunction rules to the constraints on c as long as possible, etc.

This has to be done independently for all the exists restrictions on b_0. Since we now consider only one individual at a time we need no longer explicitly introduce names for the individuals, and we have already pointed out that one can forget about the role relationships. It is now enough to memorize the concept constraints currently imposed on the actual individual by the corresponding set of concept terms. Obviously, if the conjunction rule (resp. disjunction rule) has been applied for a concept term $(\mathsf{and}\ C_1\ C_2)$ (resp. $(\mathsf{or}\ C_1\ C_2)$ of this set, thus adding the terms C_1 and C_2 (resp. C_1 or C_2) to the current set, we can remove the original term from the set.

A functional algorithm which is based on these ideas is presented in Figure 3. Please note that the algorithm, which is described in a Lisp-like notation, can very easily be implemented.

4.3 An Algorithm for Checking the Consistency of an ABox

In this subsection, an algorithm for solving the consistency problem of an ABox will be sketched with the help of an example. As for the satisfiability algorithm, the idea behind this consistency algorithm is that it tries to construct a model for a given ABox. One can view the consistency problem of an ABox as a generalization of the satisfiability problem of concept terms. In fact, suppose that the ABox $\mathcal{A}$ contains the axioms $(\mathsf{assert\text{-}ind}\ a\ C_1), \ldots, (\mathsf{assert\text{-}ind}\ a\ C_n)$. If $\mathcal{A}$ is consistent, then the concept term $(\mathsf{and}\ C_1 \ldots C_n)$ is obviously satisfiable. Thus, a simple-minded idea for a consistency

$sat(\mathcal{C}) =$
  if $A \in \mathcal{C}$ *and* (not A) $\in \mathcal{C}$ *for some concept name* A
    then false
  else if (and C_1 C_2) $\in \mathcal{C}$
    then $sat(\mathcal{C} \setminus \{(\text{and } C_1\ C_2)\} \cup \{C_1, C_2\})$
  else if (or C_1 C_2) $\in \mathcal{C}$
    then $sat((\mathcal{C} \setminus \{(\text{or } C_1\ C_2)\}) \cup \{C_1\})$ *or* $sat((\mathcal{C} \setminus \{(\text{or } C_1\ C_2)\}) \cup \{C_2\})$
  else if for all (some R C) $\in \mathcal{C}$
      $sat(\{C\} \cup \{D \mid (\text{all } R\ D) \in \mathcal{C}\})$
    then true
    else false

Figure 3: A functional algorithm deciding satisfiability of $\mathcal{ALC}$-concepts. A concept term C in negation normal form is satisfiable if and only if the call $sat(\{C\})$ returns *true*.

checking algorithm could be: Check for every individual a occurring in the ABox whether the conjunction of all concept terms C_i with (assert-ind a C_i) $\in \mathcal{A}$ is satisfiable. The following example, however, shows that this naive algorithm may fail to detect that an ABox is inconsistent.

Suppose the ABox

$$\mathcal{A} = \{(\text{assert-ind Tim Tom child}),\ (\text{assert-ind Tom Human})\}$$

is given, and we are interested in whether the fact (assert-ind Tim (some child Human)) is implied by $\mathcal{A}$. As mentioned in the previous section this instantiation problem can be reduced to the test whether

$$\mathcal{A}' = \mathcal{A} \cup \{(\text{assert-ind Tim (all child (not Human)))}\}$$

is inconsistent.[6] The naive consistency algorithm from above checks whether the concept terms Human (coming from the individual Tom) and (all child (not Human)) (coming from Tim) are satisfiable. Since both concept terms are satisfiable it concludes that $\mathcal{A}'$ is consistent. However, it is easy to see that $\mathcal{A}'$ is inconsistent.

The reason why this simple algorithm does not detect the inconsistency is that it ignores role relationships occurring in the ABox. The interaction of role relationships with value restrictions may enforce that individuals of the ABox are instances of additional concepts. Thus, to overcome this problem, we modify our simple algorithm as follows. In a *preprocessing step* we enlarge a given ABox by axioms implied by the interaction of role relationships with value restrictions. If an ABox contains the axioms (assert-ind a b R) and (assert-ind a (all R C)), then the axiom (assert-ind b C) has to be added. This is one of the rules applied in the preprocessing step. However, this rule alone is not sufficient. If (assert-ind a b R) and (assert-ind a (and $\ldots$ (all R C) $\ldots$) are in an ABox, we also have to enlarge the ABox by the axioms (assert-ind b C). Thus we also have to decompose conjunctive and, for similar reasons, disjunctive concept terms occurring in the ABox. This yields the two other rules for the preprocessing step. The preprocessing is finished

[6] Note that (all child (not Human)) is the negation normal form of (not (some child Human)).

if applications of the three rules do not add new axioms to the current ABox. As a consequence, role relationships in the ABox thus obtained can be ignored because they no longer carry any additional information. Now, in a second step we can use the simple consistency algorithm mentioned before. This yields a correct and complete algorithm for deciding consistency of an ABox of $\mathcal{ALC}$.

As an example, let us apply this consistency algorithm to the ABox $\mathcal{A}'$ from above. The preprocessing step returns the ABox

$$\mathcal{A}'' = \mathcal{A}' \cup \{(\texttt{assert-ind Tom (not Human)})\}.$$

In the second step we collect for each individual occurring in $\mathcal{A}''$ its concept constraints, and apply a satisfiability algorithm to their conjunction. Thus, to check whether $\mathcal{A}''$ (and hence $\mathcal{A}'$) is consistent we check whether the concept terms (and Human (not Human)) (coming from the individual Tom) and (all child (not Human)) (coming from Tim) are satisfiable. Since the first concept term is obviously not satisfiable, we now correctly conclude that $\mathcal{A}'$ is inconsistent.

5 $\mathcal{KRIS}$: the Overall Structure

In this section we give a short description of $\mathcal{KRIS}$. The representation component offers the formalisms presented in Section 2: a very expressive concept language and an assertional language which is similar to the languages used in most KL-ONE systems. The reasoning component of $\mathcal{KRIS}$ provides sound and complete algorithms which solve the problems mentioned in the previous section.

$\mathcal{KRIS}$ is implemented in Common Lisp on a Symbolics Lisp machine. The main menu of $\mathcal{KRIS}$ is shown in Figure 4.

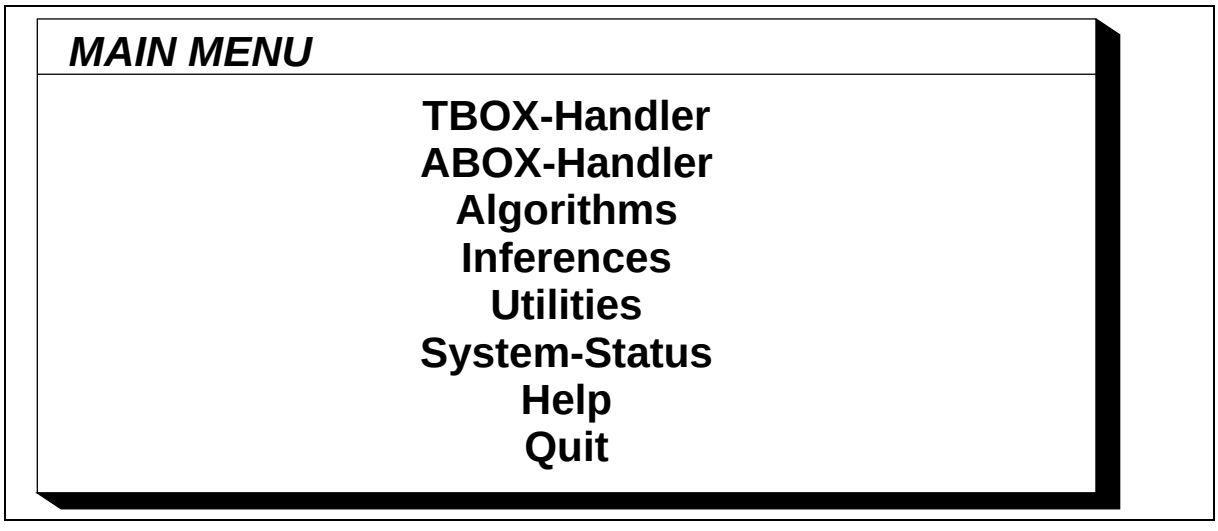

Figure 4: $\mathcal{KRIS}$ main menu.

Clicking one of the menu items causes $\mathcal{KRIS}$ to generate submenus. They allow the following operations.

- The *TBox-Handler* organizes the treatment of terminologies. That means, it can be used to create, load, edit, and delete TBoxes.
- Similarly, the *ABox-Handler* manages ABoxes.
- The item *Algorithms* allows to choose an appropriate algorithm. We have implemented several algorithms for the inferences which are based on different data-structures. Furthermore, for some sublanguages of the concept language presented in Section 2 we have implemented optimized algorithms.
- We can start a chosen algorithm using *Inferences*. $\mathcal{KRIS}$ provides algorithms which solve the consistency problem, the subsumption problem, the instantiation problem, the classification problem, the realization problem, and the retrieval problem.
- *Utilities* provides possibilities to measure the run-time of algorithms.
- *Help* and *System-Status* give more informations about the system.

$\mathcal{KRIS}$ can be used as follows. First of all, the user has to edit the terminological and assertional knowledge of the domain of interest using *TBox-Handler* and *ABox-Handler*. Assume that the TBox of Figure 1 and the ABox of Figure 2 have been edited, and hence are known to $\mathcal{KRIS}$. The consistency algorithm will find out that the represented knowledge is consistent. That means, there exists a model for the ABox w.r.t. the TBox. The classification algorithm computes the subsumption hierarchy as shown in Figure 5.

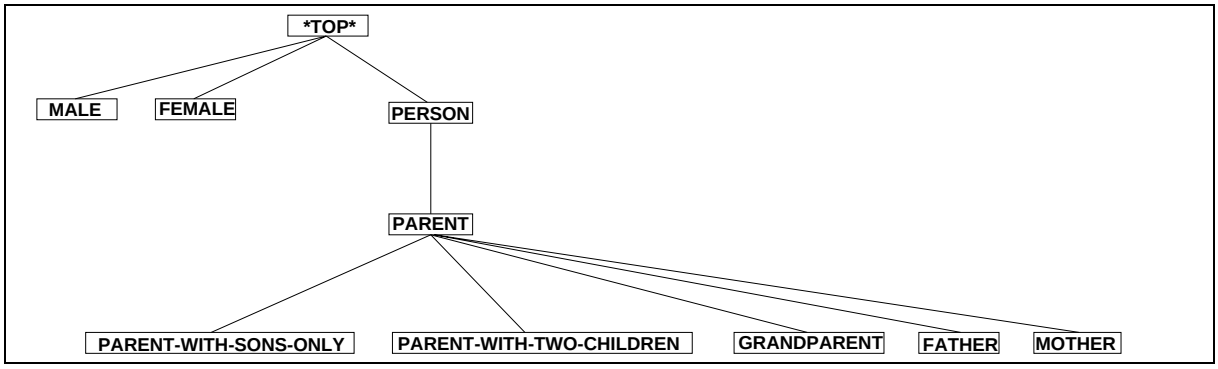

Figure 5: The subsumption hierarchy of the TBox given in Figure 1.

One can use the instantiation algorithm to get the most accurate information about an individual. For example, the algorithm will detect the following relationships:

individual	most specialized concepts
`Tom`	`father`, `parent_with_two_children`
`Mary`	`parent_with_two_children`, `grandparent`, `parent_with_sons_only`

The retrieval algorithm computes for a given concept term the individuals of the ABox which are instances of it:

concept term	individuals
grandparent	Mary
parent_with_two_children	Mary, Tom
(some sex male)	Tom, Chris

That means, for instance, (*i*) the fact that Tom and Chris have sex male is implied by the represented knowledge, and (*ii*) for the other individuals in $\mathcal{A}$ this property cannot be concluded.

The user may cause $\mathcal{KRIS}$ to compute for a given TBox and ABox (*i*) the subsumption hierarchy, (*ii*) for every individual in the ABox the most specialized concepts, and (*iii*) for every concept name in the TBox the individuals which are instances of it. After $\mathcal{KRIS}$ has once determined these structures, it is able to access this information efficiently.[7] Note that only a small amount of memory is needed to store this information. Consequently, the subsumption problem and the retrieval problem for concepts defined in the TBox, and the instantiation problem can afterwards be solved very fast by looking into the precomputed structures.

At any time the user may add terminological and assertional axioms to an already existing TBox and ABox. Assume that $\mathcal{KRIS}$ has computed the structures mentioned before. In this case $\mathcal{KRIS}$ gives the user the possibility to update these structures. If a terminological axiom is added, then, for instance, the subsumption hierarchy is enlarged by the inserting concept name defined by the axiom at the appropriate place.

6 Summary and Outlook

The $\mathcal{KRIS}$ system which has been presented in this paper distinguishes itself from all the other implemented KL-ONE based systems in that it employs complete inference algorithms. Nevertheless its concept language is relatively large. Of course, the price one has to pay is that the worst case complexity of the algorithms is worse than NP. But it is not clear whether the behaviour for "typical" knowledge bases is also that bad. An important reason for implementing the $\mathcal{KRIS}$ system was that it could be used to investigate this question.

Thus an important part of our future work will be to test the system with typical applications. In addition, we intent to further extend the system. On the one hand, we want to integrate the possibility to refer to concrete domains (such as integers, real numbers, strings, etc.) in the definition of concepts [BH90]. On the other hand, we will allow further concept forming operators such as qualifying number restrictions [HB91] and role forming operators such as transitive closure of roles [Baa90c] (at least for a sublanguage of the presented concept language); for additional constructs see [BBHH$^+$90].

Another point is that until now the user has to specify which algorithm should be used. In an improved $\mathcal{KRIS}$ version, this system will itself choose the optimal algorithm

[7] The idea that some of the important inferences can be computed in advance was already used in the original KL-ONE system. Cf. [BS85] p. 178: "In KL-ONE the network (i.e. the subsumption hierarchy) is computed first from the forms of descriptions, and subsumption questions are always read off from the hierarchy."

by inspecting what combination of language constructs are used.

The main objective of our research group WINO—as a part of the larger project AKA (Autonomous Cooperating Agents)—is the investigation of logical foundations of knowledge representation formalisms which can be used for applications in cooperating agent scenarios [BM91]. Thus our long term goals also comprise further extensions of $\mathcal{KRIS}$ such as

- a constrained-based approach for integrating full first order predicate logics with concept languages [BBHNS90, Bür90] which can be used to represent non-taxonomical knowledge,
- modal-logical approaches for the integration of knowledge concerning time and space.

Acknowledgements. We are grateful to our colleague Werner Nutt for his remarks concerning the implementation. We would like to thank Erich Achilles, Armin Laux, Jörg Peter Mohren, and Gebhard Przyrembel for their implementational work. This research was supported by the German Bundesministerium für Forschung und Technologie under grant ITW 8903 0.

References

[Baa90a] F. Baader. "Terminological Cycles in KL-ONE-based Knowledge Representation Languages." In *Proceedings of the 8th National Conference of the AAAI*, pp. 621-626, Boston, Mas., 1990.

[Baa90b] F. Baader. "A Formal Definition for the Expressive Power of Knowledge Representation Languages." In *Proceedings of the 9th European Conference on Artificial Intelligence*, pp. 53–58, Stockholm, Sweden, 1990.

[Baa90c] F. Baader. "Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles." To appear in *Proceedings of IJCAI '91*

[BBHH$^+$90] F. Baader, H.-J. Bürckert, J. Heinsohn, B. Hollunder, J. Müller, B. Nebel, W. Nutt, H.-J. Profitlich. *Terminological Knowledge Representation: A Proposal for a Terminological Logic.* DFKI Technical Memo TM-90-04, DFKI, Postfach 2080, D-6750 Kaiserslautern, West Germany.

[BBHNS90] F. Baader, H.-J. Bürckert, B. Hollunder, W. Nutt, J. H. Siekmann. "Concept Logics" In *Proceedings of the Symposium on Computational Logics*, Brüssel, November 1990.

[BH90] F. Baader, P. Hanschke. "A Schema for Integrating Concrete Domains into Concept Languages." To appear in *Proceedings of IJCAI '91*

[BBMR89] A. Borgida, R. J. Brachman, D. L. McGuinness, L. A. Resnick. "CLASSIC: A Structural Data Model for Objects." In *Proceedings of the International Conference on Management of Data*, Portland, Oregon, 1989.

[BPGL85] R. J. Brachman, V. Pigman Gilbert, H. J. Levesque. "An essential hybrid reasoning system: knowledge and symbol level accounts in KRYPTON." In *Proceedings of the 9th IJCAI*, pp. 532–539, Los Angeles, Cal., 1985.

[BL84] R. J. Brachmann, H. J. Levesque. "The tractability of subsumption in frame based description languages." In *Proceedings of the 4th National Conference of the AAAI*, pp. 34–37, Austin, Tex., 1984.

[BS85] R. J. Brachman, J. G. Schmolze. "An Overview of the KL-ONE knowledge representation system." *Cognitive Science*, 9(2):171-216, April 1985.

[Bür90] H.-J. Bürckert. "A Resolution Principle for Clauses with Constraints" In *Proceedings of the 10th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence, LNAI 449, Springer Verlag, pp. 178-192, 1990.

[BM91] H.-J. Bürckert, J. Müller. "RATMAN: A Rational Agent Testbed for Multi Agent Networks", In *Proceedings of Modeling Autonomous Agents in Multi-Agent Worlds*, Elsevier Publishers, 1991.

[DLNN91] F. Donini, M. Lenzerini, D. Nardi, W. Nutt. "The Complexity of Concept Languages." In J. A. Allan, R. Fikes, E. Sandewall (editors), *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, Mas., 1991.

[Hol90] B. Hollunder. "Hybrid Inferences in KL-ONE-based Knowledge Representation Systems." In *Proceedings of the 14th German Workshop on Artificial Intelligence*, pp. 38–47, Eringerfeld, Germany, 1990.

[HB91] B. Hollunder, F. Baader. "Qualifying Number Restrictions in Concept Languages." In J. A. Allan, R. Fikes, E. Sandewall (editors), *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, Mas., 1991.

[HN90] B. Hollunder, W. Nutt. *Subsumption Algorithms for Concept Description Languages.* DFKI Research Report RR-90-04, DFKI, Postfach 2080, D-6750 Kaiserslautern, West Germany.

[HNS90] B. Hollunder, W. Nutt, M. Schmidt-Schauß. "Subsumption Algorithms for Concept Description Languages." In *Proceedings of the 9th European Conference on Artificial Intelligence*, pp. 348–353, Stockholm, Sweden, 1990.

[KBR86] T. S. Kaczmarek, R. Bates, G. Robins. "Recent developments in NIKL." In *Proceedings of the 5th National Conference of the AAAI*, pp. 578–587, Philadelphia, Pa., 1986.

[Kob89] A. Kobsa. "The SB-ONE knowledge representation workbanch" In *Preprints of the Workshop on Formal Aspects of Semantic Networks*, Two Harbors, Cal., February 1989.

[LB87] H. J. Levesque, R. J. Brachman. "Expressiveness and tractability in knowledge representation and reasoning." *Computational Intelligence*, 3:78–93, 1987.

[MB87] R. MacGregor, R. Bates. *The Loom Knowledge Representation Language.* Technical Report ISI/RS-87-188, University of Southern California, Information Science Institute, Marina del Rey, Cal., 1987.

[Neb90] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, Lecture Notes in Artificial Intelligence, LNAI 422, Springer Verlag, 1990.

[Neb89] B. Nebel. "Terminological Cycles: Semantics and Computational Properties." In *Proceedings of the Workshop on Formal Aspects of Semantic Networks*, Two Harbors, Cal., February 1989.

[Neb88] B. Nebel. "Computational complexity of terminological reasoning in BACK." *Artificial Intelligence*, 34(3):371–383, 1988.

[NvL88] B. Nebel, K. von Luck. "Hybrid Reasoning in BACK." In Z. W. Ras, L. Saitta (editors), *Methodologies for Intelligent Systems*, pp. 260–269, North Holland, Amsterdam, Netherlands, 1988.

[Pat84] P. Patel-Schneider. "Small can be beautiful in knowledge representation." In *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, pp. 11–16, Denver, Colo., 1984.

[Pat89] P. Patel-Schneider. "A four-valued Semantics for Terminological Logics." *Artificial Intelligence*, 39(2):263-272, 1989.

[PSOK+90] P. Patel-Schneider, B. Owsnicki-Klewe, A. Kobsa, N. Guarino, R. MacGregor, W. S. Mark, D. L. McGuinness, B. Nebel, A. Schmiedel, J. Yen. "Term Subsumption in Knowledge Representation." In *AI Magazine*, 11(2):16-23, 1990. pp. 11–16, Denver, Colo., 1984.

[Sch89] M. Schmidt-Schauß. "Subsumption in KL-ONE is undecidable." In R. J. Brachmann, H. J. Levesque, R. Reiter (editors), *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pp. 421–431, Toronto, Ont., 1989.

[SS88] M. Schmidt-Schauß, G. Smolka. "Attributive Concept Descriptions with Complements". *Artificial Intelligence*, 48:1–46, 1991.

[Vil85] M. B. Vilain. "The restricted language architecture of a hybrid representation system." In R. J. Bachmann, H. J. Levesque, R. Reiter (editors), *Proceedings of the 9th IJCAI*, pp. 547–551, Los Angeles, Cal., 1985.