Published in D. Karagianis (ed.), *Information Systems and Artificial Intelligence: Integration Aspects*, p. 181–212. Springer-Verlag, Berlin, Heidelberg, New York, 1991.

Terminological Reasoning and Information Management*

Bernhard Nebel
Deutsches Forschungszentrum
für Künstliche Intelligenz
D-6600 Saarbrücken

Christof Peltason
Technische Universität Berlin
Project KIT-BACK
D-1000 Berlin 10

Abstract

Reasoning with terminological logics is a subfield in the area of knowledge representation that evolved from the representation language KL-ONE. Its main purpose is to automatically determine the location of a a new concept description (or object description) in a partially ordered set of given concepts. It seems to be a promising approach to apply the techniques developed in this area to the development of new object-based database models. The main advantages are a uniform query and database definition language and the utilization of an indexing technique, which we call semantic indexing.

1 Introduction

The development of elaborate techniques for information description is an important task in building advanced information systems. The appropriate means of describing classes, objects, and complex dependencies of an application domain can help users to express their problems in a natural way and make an important contribution to the effort of turning an information management system into a system which finally might be called a system managing a "knowledge base". In addition, description techniques can also be exploited to guide the internal reasoning and retrieval processes of the information management system. The main goal of this paper is to show that a proper treatment of the domain terminology is a good starting point for dealing with both aspects, the usage aspect of the

*This work was supported by the German Ministry for Research and Technology BMFT under contract ITW 8901 8, and by the Commission of the European Communities within ESPRIT Project 311.

system's expressiveness and the implementation aspect of information management.

We will start in Section 2 with a short overview of the underlying *terminological logic* approach. This representation paradigm has evolved from the work in the context of the knowledge representation language KL-ONE [12], and has gained a wide audience during the last decade. We will show how to use a terminological description language for the modelling of domain entities, and how to build expressions for complex information retrieval tasks within this framework. This exposition will use the formalism employed in the BACK system [42], thereby introducing an essential subset of the BACK-formalism.

In Section 3 we will sketch some aspects of implementing an information system following this approach. We discuss how management and persistency procedures can take advantage of a knowledge base which is structured by a terminological scheme.

The interdependencies between the related work in knowledge representation and advanced database systems are sketched in Section 4. One interesting result of this survey is that database research can profit from research in knowledge representation. In particular, we will point out that the *refinement* algorithm for the object-oriented database system O_2 published in [27] is incomplete and argue that the problem itself is intractable – insights based on theoretical results achieved in the area of terminological logics.

While the approach we focus on in this paper stems from a tradition in Artificial Intelligence research it is interesting to see how it has gradually also become part of the converging tendencies between research on databases and AI (cf. [13]). Originally, the notion of a knowledge representation system was often used in its most ambitious variant, i.e. as an attempt to support the representation of all aspects of knowledge, such as dynamic processes, various kinds of natural-language phenomena, uncertain or vague information, beliefs, and many more. The paradigm of terminological reasoning, however, started from a very limited (but well-founded, set-theoretical) formalism. Although the limitation led to an increasing distance from the initial, purely AI-oriented goals, the results were acknowledged as contributions to the research area of database systems, where they now are attracting a growing interest. The reason is that – due to the formal rigidity of the approach – the behaviour of the systems can be estimated in a reliable way and the representational service they provide can be seen as playing a central role within any advanced information management system.

In order to sketch a first, intuitive picture of the idea we look at the following scenario:

Let us assume we are talking about organizations, i.e. universities, companies, and research institutes applying for research projects, or more precisely *Esprit* projects. The legal status of these organizations is determined by their locations

(or rather where they have their formal residences), by the number of employees, etc., which finally makes them eligible for Esprit projects and constitutes Esprit Consortia. What kind of interaction features would we expect from an "intelligent" information system?

First, a support for building a formal model of this application domain, i.e. for describing the abstract entities such as *universities* or *CEC-companies* is required. We may also need a control of the dependencies between these descriptions in the modelling phase in order to be able to estimate the reasoning processes in the next phases. Then we would like to enter information about concrete objects in our domain such as:

> *The German Research Center for AI and the Technical University Berlin are members of the Esprit Project 42 Consortium.*

No deeper, implementation dependent knowledge of the underlying scheme should be required for such entries. All information should be describable in a logic-oriented way. Finally, we would like to retrieve our information in as comfortable a manner as possible:

> *What are the Esprit Consortia which have only small and medium enterprises (SMEs) as their members?*

On the whole, entries and queries should be describable in a logic-oriented way using complex descriptions of *what* we want to know rather than requiring references as to *where* to find it. In the following sections we will see how this is reflected in a uniform approach for knowledge base access.

2 The Terminological Reasoning Approach

The terminological approach is based on a clear distinction between intensional and extensional descriptions. Although originally introduced as an epistemological category, the distinction turns out to be useful from a technical information processing point of view as well. It offers a clean methodology for distinguishing between the level for reasoning about abstract classes and the level for reasoning about objects which instantiate these classes. On the one hand, this brings the notion close to the conventional database-like distinction between *database scheme* and *database extensions*. On the other hand, however, the language designed for intensional descriptions (which could be called a *knowledge base scheme language*) is an expressive language within which complex descriptions can be built, thus constituting a highly complex data model which – for a semantically well-founded language – includes a complex reasoning machinery.

2.1 Term Description Language

The language designed for intensional descriptions is called the *Term Description Language*. It contains a repertoire of constructs which may vary among the different incarnations of this system family, however, in its core at least the following can be identified:

Terminology

Intensional descriptions are introduced as *term equations* with *names* as left-hand sides and composite *terms* as right-hand sides. There are two kinds of introductions, *primitive* ones, indicated by :<, and *defined* ones, indicated by :=, an alternative we will explain further below. A sequence of such introductions makes up the *terminology*, or the *terminological model* of a domain.[1]

$$
\begin{array}{lcl}
\langle \textit{term-tell} \rangle & ::= & \langle \textit{concept-}\textsc{name} \rangle :< \langle \textit{concept} \rangle \\
& | & \langle \textit{concept-}\textsc{name} \rangle := \langle \textit{concept} \rangle \\
& | & \langle \textit{role-}\textsc{name} \rangle :< \langle \textit{role} \rangle \\
& | & \langle \textit{attribute-set-}\textsc{name} \rangle := \langle \textit{attribute-set} \rangle
\end{array}
$$

For building terms we distinguish between *classes* and *roles* where *classes* denote sets and *roles* denote relations between these sets. Among the different kinds of classes we use *concepts* for intensional descriptions which – by virtue of semantically based subsumption – form an abstraction hierarchy:

$$
\begin{array}{lcl}
\langle \textit{term} \rangle & ::= & \langle \textit{class} \rangle \\
& | & \langle \textit{role} \rangle \\
\langle \textit{class} \rangle & ::= & \langle \textit{concept} \rangle \\
& | & \langle \textit{attribute-set} \rangle \\
& | & \langle \textit{number-set} \rangle
\end{array}
$$

In the simplest case a *class* is introduced as a *primitive concept* by stating necessary conditions which determine its membership.

```
University :< Organization
```

A university is – necessarily – an organization.

[1] The semantics of the term description language is given in the Appendix.

Concept Terms

In order to produce composite descriptions out of such *primitive* introductions, *concepts* can be joined with other *concepts*, or their relations to other *concepts* can be restricted w.r.t. range and number of fillers.[2] If a composite description contains all necessary and sufficient conditions for an object to instantiate it the description is introduced as a *defined concept*.

⟨*concept*⟩ ::= ⟨*concept*-NAME⟩
| ⟨*concept*⟩ **and** ⟨*concept*⟩
| **all** (⟨*role*-NAME⟩ , ⟨*class*⟩)
| **atleast** (⟨NUMBER⟩ , ⟨*role*-NAME⟩)
| **atmost** (⟨NUMBER⟩ , ⟨*role*-NAME⟩)
| **all1** (⟨*role*-NAME⟩ , ⟨*class*⟩)
| **anything** | **nothing**

```
Esprit-Eligible   :=   Company
                       AND ALL1(has-residence, CEC-country)
```

An Esprit-Eligible company is defined as a company which has its residence in a CEC-country.

The notion of forming a structured description, whose meaning is solely determined by its internal structure is a central characteristic for the terminological approach. It allows us to explicate all implicitly given subsumption relations between *classes*, and it allows us to deduce the correct class membership from the set of features known about an instantiating *object*. We will discuss these inferences, which are referred to as *classification* and *realization*, in more detail below.

Role Terms

Roles denote relations between concepts. In analogy to the *primitive concept* hierarchy they form a hierarchy of *primitive roles*.

⟨*role*⟩ ::= ⟨*role*-NAME⟩
| ⟨*role*⟩ **and** ⟨*role*⟩
| **domain** (⟨*concept*⟩)
| **range** (⟨*class*⟩)

[2] The number of role fillers can be given by specifying upper or lower bounds of a number interval. For roles having at least one filler we use the abbreviation **all1**.

```
Consortium        :<   ANYTHING
has-members       :<   DOMAIN(Consortium)
                       AND RANGE(Organization)
```

Has-members is a relation between consortium and organization.

The global view of the *role* hierarchy which is implied by this kind of *role* introduction should be contrasted with the view of *roles* specifying certain local restrictions at *concepts* (we already made use of this above):

```
Esprit-Consortium :=   Consortium
                       AND ALL1(has-members, Esprit-Eligible)
```

An Esprit Consortium is defined as a consortium which has only Esprit-Eligible members.[3]

Local restrictions may not be in conflict with the global *role* hierarchy.

Attribute Set and Number Set Terms

While *concepts* and *roles* constitute the classical representational core of a term description language, some variants for representing *concepts* are added: *attribute-sets* are used for dealing with sets of attribute values in cases where a set can better be represented by enumerating all of its elements.[4] The key word **attribute** denotes the set of all possible attribute values.

⟨*attribute-set*⟩ ::= **aset** (⟨*attribute*-NAME⟩$^+$)
| **attribute**

```
European-Country  :=   ASET(Belgium, Denmark, England,
                       France, Germany, Greece, Ireland, Italy,
                       Luxemburg, Netherlands, Portugal, Spain,
                       Bulgaria, Czechoslovakia, Finland,
                       Hungary, Yugoslavia, Norway, Austria,
                       Poland, Romania, Russia, Sweden,
                       Switzerland, Turkey)
CEC-Country       :=   ASET(Belgium .. Spain, European-Country)
```

[3] In fact, the actual definition of an Esprit-Consortium is:

Each Consortium must include at least two independent industrial organisations from the Community not established in the same Member State (cf. [14]).

Dealing with these kinds of additional constraints requires a language which supports more complex descriptional techniques for *roles*. Quantz provides the complete solution in [43].

[4] While this kind of definition is also known as *extensional definition* it should not be confused with the extensional level where we are dealing with instances.

In the abbreviated definition we take advantage of the total order of the elements given in the initial definition.

For a more convenient way of dealing with ranges of numbers *number-sets* are used. The key word **number** denotes the set of all integers.

⟨*number-set*⟩ ::= ⟨NUMBER⟩
| <⟨NUMBER⟩ | >⟨NUMBER⟩
| **number**

```
sme := Company AND ALL1(has-employees, <50)
```

A small and medium enterprise is defined as a company with at most 50 employees.

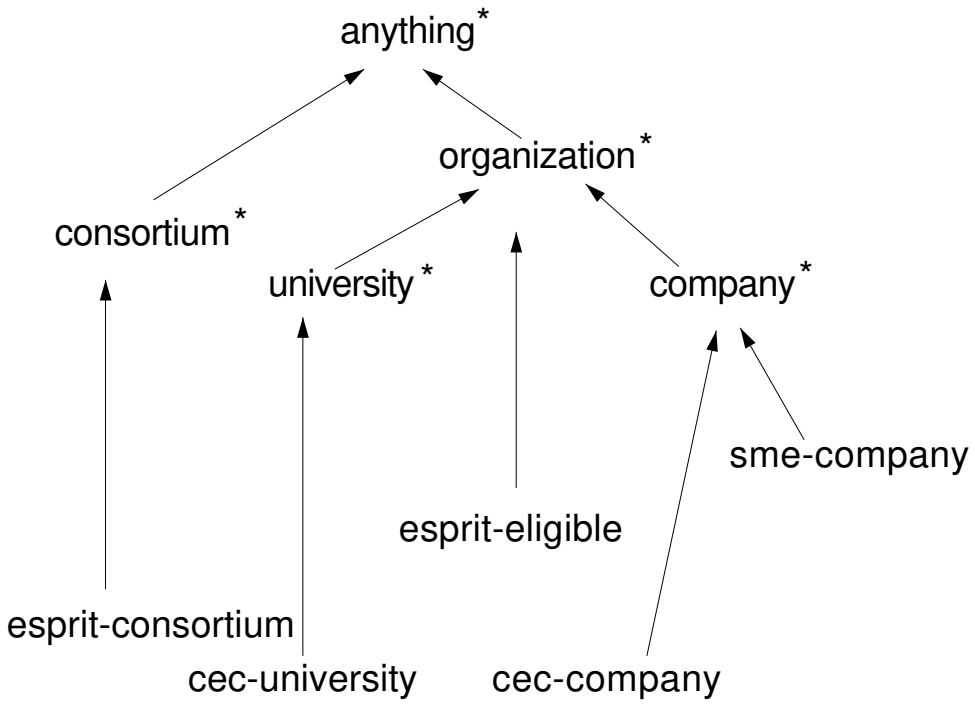

Figure 1: Concept Hierarchy (primitive concepts are indicated by an asterisk).

In addition to the constructs described above most existing systems based on term description languages support a large variety of additional functionalities, e.g. for getting knowledge base scheme information, revising the scheme, dealing with simple set operations on attribute-sets and number-sets, and more.

Let us look at the example now in a more complete version:

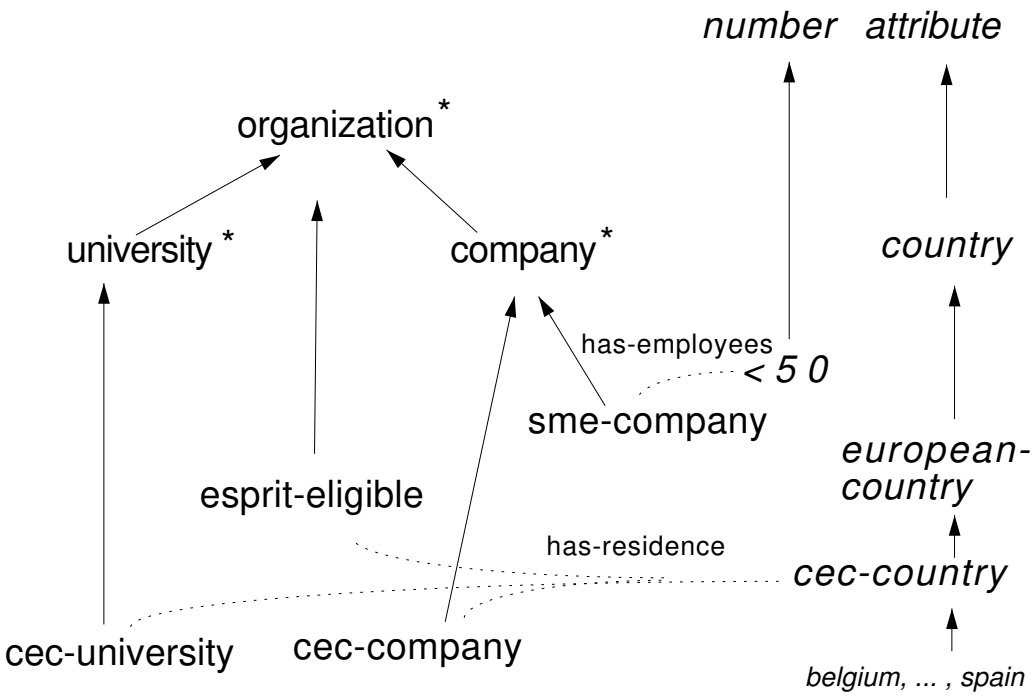

Figure 2: Concepts with Role Restrictions

```
European-Country  := ASET(Belgium .. Turkey)
CEC-Country       := ASET(Belgium .. Spain, European-Country)
Organization      :< ANYTHING
has-employees     :< DOMAIN(Organization) AND RANGE(NUMBER)
has-residence     :< DOMAIN(Organization) AND RANGE(ATTRIBUTE)
has-name          :< DOMAIN(Organization) AND RANGE(ATTRIBUTE)
Consortium        :< ANYTHING
has-members       :< DOMAIN(Consortium) AND RANGE(Organization)
University        :< Organization
CEC-University    := University AND ALL1(has-residence, CEC-Country)
Company           :< Organization
CEC-company       := Company AND ALL1(has-residence, CEC-Country)
SME-company       := Company AND ALL1(has-employees, <50)
CEC-SME-company   := SME-company AND CEC-company
European-SME      := SME-company AND ALL1(has-residence, European-Country)
Esprit-Eligible   := Organization AND ALL1(has-residence, CEC-Country)
Esprit-Consortium := Consortium AND ALL1(has-members, Esprit-Eligible)
```

The sequence of *term equations* forms a *concept* and a *role* hierarchy (and also the trivial, but useful *attribute-set* and *number-set* hierarchies). Seen as a data structure the concept hierarchy forms a labelled directed acyclic graph which – for a slightly extended version of our example sequence – are shown in Figures 1 and 2.

2.2 Object Description and Retrieval

In the process of formalizing the domain for further processing, so far, we have a collected a number of *term equations* which constitute the *terminological model* of the domain. The next step is to introduce assertions about *objects* which instantiate the *classes* of the terminological model. In the simplest case an assertion is specified by introducing a *unique name* (it may also have synonyms) as identifier and specifying the *class* the *object* instantiates:

```
u-52  =   University
c-99  =   Company
```

A sequence of such *object equations* is called the *assertional knowledge base*. In addition, specifications may consist of complex assertional terms integrating composite terms of the term description language:

$$
\begin{array}{rcl}
\langle \textit{object-tell} \rangle & ::= & \langle \textit{object-}\textsc{name} \rangle = \langle \textit{class-expression} \rangle \\
\langle \textit{object-ask} \rangle & ::= & \langle \textit{object-}\textsc{var} \rangle = \textbf{getall}\ \langle \textit{class-expression} \rangle \\
\langle \textit{class-expression} \rangle & ::= & \langle \textit{concept} \rangle \\
 & | & \langle \textit{attribute-set} \rangle \\
 & | & \langle \textit{concept} \rangle\ \textbf{with}\ \langle \textit{role} \rangle : \langle \textit{value-expression} \rangle \\
\langle \textit{value-expression} \rangle & ::= & \langle \textit{object-}\textsc{name} \rangle \\
 & | & \textbf{close}\ (\ \langle \textit{value-expression} \rangle\ ) \\
 & | & (\ \textbf{all}\ \langle \textit{class-expression} \rangle\ ) \\
 & | & \langle \textit{value-expression} \rangle\ \textbf{and}\ \langle \textit{value-expression} \rangle
\end{array}
$$

In fact, it is the salient point of the approach presented here that the knowledge base access language also allows the integration of descriptive parts of the intensional level. Starting with the terminological scheme given in our example we can enter information about instantiating objects, e.g. an object C-99:

C-99 *is a company.*
C-99 *has 30 employees.*
C-99 *has its residence in Italy.*

All these entries are taken in conjunction. In general, the object identifier may point to a *composite description* of the term description language without requiring any information about the *exact name* of the class this object ultimately belongs to. Determining the membership is done by the process of *realization* which can be seen as the counterpart of *classification* on the extensional level. Querying, e.g., for *all Esprit-Eligible SMEs* which are known so far in the knowledge base, we can expect that object C-99 will be retrieved which is achieved by

classification where, in this case, it is explicated from the above definitions that *all European SME companies* are subsumed by the set of *Esprit-eligible companies*.

In addition, the language also shows descriptional means called *value expressions* by which information on the fillers of roles can be determined. If e.g. we use an **and** in the value expression that restricts a role, those objects are retrieved that have the fillers in conjunction as their role fillers.[5]

In general, an assertional language designed for practical usage should provide many additional features including operators for anonymous referencing, recursive nesting, chaining, and others.[6]

2.3 Characteristics of the Terminological Approach

Recalling the main principles of most systems designed under the terminological paradigm we should stress the features of a *well-defined semantics*, the *specialized reasoning* style for various reasoning types, and the *class-based* and *object-based organization* of domain entities.[7]

We should review how these essential elements influence two main aspects of an information system, expressiveness characteristics and inferential capabilities.

Expressive Knowledge Base Language

As already pointed out, the term description language can be seen as a language for the specification of the knowledge base scheme (in analogy to a data definition language), and the language part for assertional *objects* as a knowledge base assertion and query language (in analogy to a data manipulation language). However, the knowledge base access language we presented within the previous section focusses on a close integration of both description levels, the intensional description language, and description of extensional ground facts.[8] How would

[5] By using such expressions without the **close** constructor partial knowledge about such role fillers can also be dealt with, i.e. the obect retrieved in this case may have additional role fillers, too.

[6] Kindermann has designed and implemented such an extended language for the BACK system, as described in [42].

[7] There is some confusion in the literature as to the appropriate expression here. We use the expression *object-based* here as a metaphor for the system's epistemological characteristics. Often such systems are also called *object-centered*, *concept-based*, or *frame-based* (sometimes the expression *frame-based* systems is used only for primitive object hierarchies).

[8] The idea of a uniform language differs from merely offering a number of single tools and lower level access functions and network editing utilities. The language integration aspect becomes significant in particular for knowledge base revision: It makes a difference whether the system offers a number of modification and revision facilities leaving part of the responsibility to the user, or if monotonic and non-monotonic update oprations are integrated into the syntax and semantics of a uniform interface language.

such a system measure up in a typical database scenario?

In conventional database approaches it is necessary that all names be known beforehand when any information service is asked for. For the task of navigation or information retrieval most information on how to identify the corresponding DB tables has to be already present. Or – in other words – once you have to know all the names and you have to tell the system exactly where to look for the information you hardly need the answer any more. In contrast, the terminological approach is more flexible and easy-to-use. Of course names are used here, too, namely the names for *classes* and *roles* introduced as *primitives*. However, using the technique of building definitions frees the user from the use of names to a large extent since the system is actually *reasoning* on descriptions, detecting equivalences, etc.

Moreover, dealing with descriptions in the way presented here is not only an advantage for knowledge base access by a user. In an extended scenario, we could imagine external system components accessing the representational service. A component collecting data and monitoring information from sensors or from permanently incoming messages would produce a vast number of descriptions. Since these descriptions would be generated automatically, the need for reasoning on descriptions becomes evident.[9]

The language expressiveness can also be exploited in improving conventional query answering techniques. Instead of producing only answers in the form of an extensional enumeration of the resulting set it is convenient to also incorporate intensional parts into the answer, thus providing a more dense type of query answering which is also able to deal with partial knowledge [8]. The semantically well-founded integration of the extensional and intensional levels within the terminological approach would seem to be a good starting point for more advanced answer generating.

Inferential Capabilities

Turning back to the initial, rather ambitious expectations about the range that knowledge representation is supposed to cover, the service offered by a terminological system may look somewhat disappointing. And even if one accepts the restriction that terminological systems focus mainly on representing *definitional* knowledge, the impression does not improve since even this service is not complete. Lacking any non-standard logical characteristics, the system seems to be able to deal only with *technical*, or *artificial* definitions, in contrast to *natural* definitions where often other descriptional methods are applied such as prototypes,

[9] For these applications – as for many others – the use of descriptions dependent on temporal relations is highly desirable. The integration of this type of reasoning within the terminological approach has been investigated in [48].

defaults, or examples (cf. [18] for a detailed critique).

However, this impression is misleading. It is obvious that basing a system on a formal semantics and opting for a computationally tractable system first leads to a restriction to a well-known logics. In addition, the terminological reasoner is only conceived as a core component, and starting with a good standard maintenance of descriptions is a sound basis for extensions to be built on top of these descriptions. Let us review the core of the inferential service provided by terminological reasoning:

- *Consistency checking* is performed for all intensional descriptions. Relations between classes are *inherited* to all specializations of these classes. Consistency conflicts and incoherencies caused by combining complex descriptions are detected. Checking for valid descriptions can be exploited in the scheme definition phase as well when querying for instances, thus providing an efficient kind of *query validation.*
- *Classification* is the process of finding the correct place for composed terminological descriptions in the hierarchy. Together with *realization*, i.e. the process of finding the best description for asserted objects, classification forms the central inference of the reasoning process.[10]
- *Completion* of partial descriptions of objects can be performed if appropriate information is given at the intensional level (e.g. in the form of maximal number restrictions), or if *closing* expressions are used at the extensional level (stating that the specified values are the only values filling the specified roles).

All these inferences are strictly deductive and definitional. In addition, it is necessary to consider further techniques which state relationships which go beyond the purely definitional ones. For example, a statement such as

> *companies are disjoint from universities*

means that a composition of both concepts `(sme AND large-company)` may be a proper definition on its own, however, there exists no instantiation by any object, a kind of disjointness which differs from definitional disjointness. Similarly, it is obvious that a rule such as

> *a company with at most 10 employees is also a company with at most 20 employees*

[10] As we will see in the next section the ability to detect "forgotten" links in the scheme definition phase is only the minor contribution of classification. Its role is of more importance in the realization phase.

should be treated differently from a rule such as

a SME-type company is also a dynamical company

since the former contains purely definitional relations. In our example language these relationships can be expressed by the *disjointness* and *implication* constructs.

However, this integration of rules is sketched here only as a first step of extending the terminological approach – in this case by applying forward chaining rules to classes [8, 45]. Additional inferential modes have already been investigated, e.g. probabilistically weighted implication [36] and default techniques [30]. If such further modes are added a terminological reasoner can also be viewed as a core component of a classical rule-based reasoner: For production rules the *left-hand sides* of the rules are obviously equivalent to complex descriptions, thus being good candidates for *right-hand sides* of terminological term quations as presented above. Since the techniques for dealing with *right-hand sides* of these equations are well understood, we gain a solid structuring basis for complex interdependencies which also overcomes the limitations of the simple rule model.

Although experiences in various case studies[11] have – in principle – shown the usefulness of description and query techniques, a broader utilization of an expressive access language depends on such additional integration with other programming paradigms. Further work is therefore needed in extending the framework by adding non-deductive reasoning modes, thus finally exploiting the descriptive capabilities of a terminological language in an inferentially powerful information system.

3 Persistency Management by Semantic Indexing

Our presentation so far has focussed on the abstract logical level of terminological logics. In this section we will try to show how this logical structure can be exploited in information processing. A number of advantages should be immediately obvious. The *classification* inference, for instance, can be employed in the process of designing a database scheme by creating a partial ordering over the concepts – the concept hierarchy. A graphical depiction of the hierarchy can then be used to verify that the concepts end up in the right places. Additionally, it is possible to transform the schema into a *minimal* form, if desired [6].

[11] See e.g. [16] for a case study on using the BACK system for modelling the legal structure of a large company group, and [17] on using the ARGON system within a software information system.

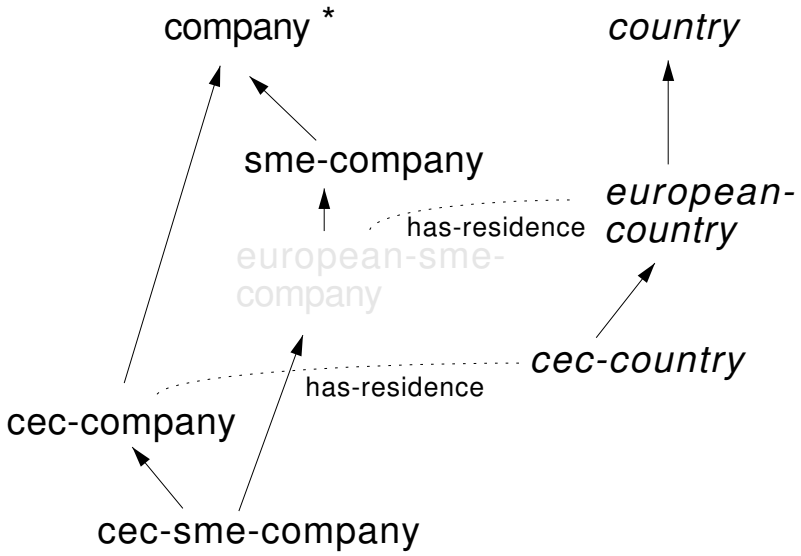

Figure 3: Extended Concept Hierarchy

Although such an application of terminological reasoning is certainly an advantage, it is not the only one and not the most important one. The more important application of classification in the database context is *query-processing* and *indexing*.

As we have seen above, queries are formulated in essentially the same language as the one used for defining the database schema, and so do not require navigation through the schema but are entirely declarative. Furthermore, it is possible to apply classification to queries as well. An obvious benefit one gains by classifying a query into the schema is that the query will be validated, i.e., it will be checked whether the concept used to express the query can possibly denote something. This is only the case if the query concept is different from the least concept, the empty concept called `Nothing`. For instance, if the query contained the concept

```
SME-companies AND ALL1(has-employees,>50),
```

it would be rejected because it is equivalent to `Nothing`.

Provided the query is semantically well-formed, i.e., not equivalent to `Nothing`, it is necessary to retrieve the objects that conforms to the concept description expressed in the query. The most simple-minded way to implement retrieval would be to scan all objects and to check whether they satisfy the concept description, i.e., whether they are an *instance* of the query-concept, and to return as the answer-set the set of all instances of the query-concept. Although this is conceptually correct, it is also quite inefficient.

In conventional database architectures, indexing is a technique to avoid scanning all instances in order to evaluate a query, but to select a small subset which is tested against the query. For instance, in the example in the last section, we would index with respect to `has-name`, `has-residence`, and `has-employees`, provided that we would like to access organizations by their names, residence, and their number of employees, respectively. Setting up such index structures is one of the important physical design considerations and can have dramatic effects on the performance of a database system. There are some disadvantages of this conventional technique, however. First, the indices must be set up once and for all and reorganization is costly and cumbersome. Second, the logical database schema is relatively neutral with regard to indexing. Which fields are to be indexed is mostly a pragmatic issue, although such information could come from information about how the user views the application domain.

Terminological logics offer a way to organize indexing along the "semantic" dimension. The idea is to set up the index parallel to the way the terminology is organized, i.e., for each concept (pointers to) all objects that are instances of this concept are stored.[12] Using this organization which exploits the "semantic" space of the terminology – and for this reason we call it *semantic indexing* – makes query processing simple and straightforward.

A query is processed in the same way as a new concept definition is handled, namely, the query-concept is temporarily classified into the concept hierarchy. As a result, either an *equivalent* concept or the set of *immediate subsumers* and *immediate subsumees* is returned. In the former case, the answer set is simply the set of all objects that are instances of the concept equivalent with the query-concept. In the latter case, the union of the sets of instances of the immediate subconcepts are included in the answer set and only objects that are instances of the immediate superconcept but not instances of the immediate subconcepts have to be checked against the query concept.

As an example, let us assume that we want to retrieve all European SME's, i.e., all instances of the concept

```
SME-company AND ALL1(has-residence, European-Country).
```

Classifying this concept into the hierarchy returns `SME-company` as the only immediate superconcept and `CEC-SME-company` as the only immediate subconcept (see the details depicted in Fig. 3). Using this result, all instances of `CEC-SME-company` are returned and all instances of `SME-company` that are not instances of `CEC-SME-company` are checked against the query-concept and included in the answer-set if they satisfy the condition.

[12] How to implement such a structure on top of a relational database system is analyzed in [23, 24].

Obviously, this technique of semantic indexing can also be used to reorganize the index structure dynamically in a transparent way. For instance, if (semantically equivalent) queries are posed very often, the system can introduce an anonymous concept on its own, extending the indexing structure in this way. Furthermore, the introduction of such anonymous concepts can also be done on information-theoretic grounds, i.e., the database can use self-organization techniques in order to tune its performance (see, e.g., [28]).

There is a price to pay for such an organization, however. Every time, a new fact about an instance is entered into the database, the instance – and perhaps other related instances as well – have to be checked and the index structure has to be reorganized, a process often called *realization*.

As an example, assume we incrementally enter information about the company `c-99` as informally described in the previous Section:

```
c-99  =    Company with has-employees:close(30)
c-99  =    Company with has-residence:close(Italy).
```

Basically, the first statement asserts that the company `c-99` has 30 employees, and the second statement asserts that the company is located in Italy. The both `close` expressions are used to state that the specified values are the only values filling the specified roles of the described object. After the first statement `c-99` will be recognized as a `SME-company`, after the second statement as a `CEC-SME-company`.

As a side-effect this process detects violations of integrity constraints expressed in concept-definitions, namely, when an object is recognized as an instance of the "empty" concept `Nothing`. For example, when instead of the second statement above the following assertion was put into the database

```
c-99  =    CEC-Company with has-residence:Poland,
```

an inconsistency would be detected. However, asserting that `c-99` is a `Company` *and* a `University` at the same time would not lead to an inconsistency since these concepts were not defined to be *disjoint* in the schema. In fact, a private university may be considered as both a university and a company.

4 Related Work

The approach described so far is not unique. A number of other research projects aim at similar goals and use comparable techniques. In the following, some of this work will be surveyed. Furthermore, a number of theoretical results concerning reasoning in terminological logics have been achieved recently which sometimes apply directly to work done in the related area of semantic data models and complex object data models.

4.1 Terminological Representation and Reasoning Systems

Besides focussing on the development of a number of terminological knowledge representation systems which were built to support AI applications,[13] some research groups explicitly address the problem of supporting database management with terminological reasoning techniques, for instance, the BACK system described in the previous sections (see also [51, 42, 44]).

The first such approach was probably the RABBIT system [50], which supports the user in selecting an appropriate restaurant using a technique called *query by reformulation*. This system was implemented on top of KL-ONE and used classification as a means for selecting the appropriate concepts in the concept network. It should be noted that this system did not differentiate between concepts and instances (or schema and data, respectively), but represented concepts and instances in a uniform way.

Subsequently, an information retrieval system called ARGON [40] was developed using the concepts and ideas of the RABBIT system. In contrast to RABBIT, however, in ARGON there is a clear-cut distinction between concepts and instances, and, more importantly, the system uses the technique of *semantic indexing* described above, which is implemented as part of the underlying representation system KANDOR [37]. ARGON was tested successfully on a small database of AI researchers (1500 individuals), on a TTL chip catalog [15],[14] and a knowledge-based software information system [17].

A more recent development is the CLASSIC system [8, 10], which takes into account recently achieved theoretical results concerning expressivity and tractability of terminological representation formalisms. In particular, the terminological logic used incorporates *co-reference constraints* – a quite powerful construction also used in similar formalisms such as ψ-terms [2] and *feature logic* [35]. This system is also aimed more at database applications because it supports updates of instances and means of applying (forward-chaining) rules to database entities. These rules do not act only as integrity constraints but they are also used to conclude additional information about database entities.[15]

[13] For instance, NIKL [21], KRYPTON [11], LOOM [30], QUIRK [7], SB-ONE [25]. For the state of the art and future developments, see [41].

[14] An interesting aspect of the TTL catalog application is that queries are always interpreted intensionally according to different levels of abstractions.

[15] Basically, these rules are similar to the implicational rules described in Section 2.3.

4.2 Semantic Data Models and Complex Object Data Models

In the past decade it has become common knowledge in the database research field that traditional record-based database systems have many serious limitations (see, e.g., [22]), and a number of so-called "semantic" data models have been proposed in order to overcome these limitation by providing mechanisms and constructs that allow modelling of the kinds of relationships that naturally occur in an application. In a certain sense, the recent wave of object-oriented database models can be understood as a natural extension of this approach.

Semantic data models and object-oriented databases have in common the phenomenon that complex descriptions are used to describe types and/or classes and, consequently, reasoning about these descriptions became a research topic. Some of the approaches in this direction are directly influenced by work done in the area of terminological reasoning. For instance, terminological reasoning has been exploited in the design of entity-relationship schemata [5]. As another example, the semantic data model CANDIDE [4] employs terminological representation and reasoning techniques on all levels, namely, for schema design, query processing, and indexing – which is not surprising since CANDIDE is based on the representation system KANDOR mentioned above.

Independently from terminological logics, research in semantic data models and object-oriented database models started to analyze the structure of types induced by their defining descriptions for the purpose of *type inference* and *type checking*. One example is IFO [1], where an *dominance* ordering over "derived types" is computed. Another example is the object-oriented data model O_2 [26, 27]. O_2 permits the composition of class descriptions using some basic types, such as integers and strings, a tuple constructor, a set constructor, and class disjunctions. The semantics is based on set theory similar to the semantics used in the area of terminological logics. Furthermore, a *refinement* ordering is defined over classes quite similar to the *subsumption* ordering discussed above. Although it is not possible to reconstruct O_2 in the terminological logic described here, more powerful language containing concept disjunction (see, e.g., [20]) could be used for this purpose. As pointed out above, however, although these database models might look similar, and indeed use similar algorithms, the intention for computing subsumption orderings is different in these approaches. It is not used to support query-evaluation or integrity control as we have sketched it in the previous sections, but only to do type checking.

One approach employing the technique of query-evalution by classification is a non-standard database application for the management of chemical structures [28]. Here subgraph-isomorphism induces a partial ordering on labeled graphs, and this partial ordering is used to efficiently store and retrieve such graphs. The difference to the technique of semantic indexing described above is that there is

no distinction between concepts and instances. Query-graphs are simply classified into the partial ordering and the immediately preceding and immediately succeeding graphs are considered as the answer.

4.3 Theoretical Issues

After having seen what can be done with terminological logics, one may pose the question of what algorithms do look like, how efficient they are, and in which way this depends on the expressiveness of the terminological logic chosen. In order to answer these questions, it is necessary to specify a rigorous and unambiguous semantics for the logics. The presentation of terminological logics in the first part of the paper has been deliberately informal. However, an intuitive and plausible Tarski-style semantics following the ideas first spelled out in [9] can be straightforwardly specified and is provided in the appendix. Based on that, the complexity of terminological reasoning has been investigated by a number of researchers and it has turned out that depending on the expressiveness of the terminological logic, the complexity of subsumption determination ranges from polynomial-time to undecidability. We will not go into the details, but only refer to some of the important results.

First of all, the ability to define concepts introduces a perhaps surprising source of complexity. To see this, one should note that in order to compute subsumption between two concepts the definitions of these concepts is usually "expanded" before the concepts are compared, i.e., all defined terms are substituted by their definitions until the expressions do not contain a defined term any more. This can be only done if there are no *cycles* in the terminology, i.e., if the defined terms do not refer directly or indirectly to themselves.

This expansion can lead to an exponential blow up in the worst case, however. Worse yet, most probably (under the assumption that P $\neq$ NP) there is no algorithm that computes subsumption between two concepts defined in a terminology in polynomial time, provided the terminological logic contains at least the operators `AND` and `ALL` [34]. This is shown by reducing equivalence of cycle-free nondeterministic finite automatons to subsumption determination. The same proof technique can be used to show that this complexity result also holds for terminological logics that contain concept disjunction (`OR`) instead of concept conjunction (`AND`), and by that it follows that the determination of the *refinement* ordering in O_2 as described in [27] is NP-hard as well – a fact which apparently was not noticed by the authors.[16] The polynomial algorithm for computing *refinement* in [27] turns out to be incomplete. One example of unnoticed refinement in O_2

[16] As a matter of fact, the the *refinement* problem in O_2 is PSPACE-complete since in O_2 cycles are permitted, so the results cited below apply.

is the following. Assume three incomparable classes `X`, `Y`, `Z`, and the following declarations:

```
A     =    X OR Y
B     =    Y OR Z
C     =    X OR Y OR Z
```

`A OR B` is obviously a subtype, a refinement, of `C`. However, the algorithm in [27] does not notice that.[17]

In any case, the apparent intractability of the subsumption problem in terminologies seems not to be relevant from a practical point of view. In almost all cases subsumption can be determined efficiently. Indeed it is possible to specify some reasonable restrictions on the form of a terminology that permit a provably polynomial algorithm [34].

Second, there might be the question of what happens if we drop the restriction mentioned above that terminologies have to be cycle-free. As it turns out, this is possible without giving up too much. If the terminological logic is intended to be used for *defining* concepts, a fixpoint – in this case a greatest fixpoint – semantics is the appropriate means to formalize the meaning of cyclic concepts (see [33, 3]).[18] The complexity result mentioned above leads to PSPACE-hardness in this case.

Third, ignoring the complexity introduced by defining concepts, one may ask how difficult subsumption determination is if the concepts are already "expanded". Obviously, this depends on the expressiveness of the term description language. As first shown in [9] and [31], only very simple term description languages permit a polynomial subsumption algorithm. Subsequently, researchers concentrated on developing ways to circumvent this complexity trap by radically restricting the expressiveness (as exercised, e.g., in KRYPTON [11], and CLASSIC [8]), by using weaker semantics that permit tractable subsumption determination [38], or by restricting the inferential capabilities in a pragmatic way leading to *sound* and fast but *incomplete* systems, such as BACK [31] and LOOM [29]). Only recently, complete algorithms for a number of powerful term description languages for which the subsumption problem is NP-hard have been developed [47, 20]. It is yet unclear, however, whether these algorithms are feasible, i.e., whether the worst case does not show up for "naturally" occurring concepts.

Interestingly, terminological logics seem to be decidable in most cases. There seems to be only one construction which causes trouble, namely, so-called *role-value-maps*, also called *co-reference constraints*, which allow one to state that the

[17] Although disappointing, this result does not affect the O_2 system as it exists. As pointed out to us by C. Lecluse, disjunctive classes are not part of the implemented system, but they were only considered as one possible extension of the system.

[18] For natural language applications other options may be more adequate, however.

role-fillers of two "role-chains" are identical. For instance, if we want to define a subconcept of `Father` such that his `last-name` is identical with the `last-name` of his `children`, a role-value-map has to be used. In general, such constructions lead to undecidability [39, 46]. However, if one constrains the interpretation of roles used in co-reference constraints to be single-valued, subsumption becomes decidable again, a fact exploited in CLASSIC.[19] This holds only as long as there are no cycles in the terminology, though. Co-reference constraints over single-valued rules *plus* cycles in the terminology are again undecidable with respect to subsumption [33]. Unfortunately, using the same argument, it can be shown that *consistency* of a CLASSIC database is in general undecidable. Technically speaking, it is undecidable whether there exists a consistent completion of a CLASSIC database.[20]

Accepting the complexity of subsumption, there remains the question of how efficient classification and the associated technique of semantic indexing may be. Unfortunately, there is no straightforward answer. Only very little is known about the average complexity of inserting new elements into a partial order – a fact which may seem surprising in face of how much is known about total orders and about trees. It is obvious that semantic indexing as described in Section 3 is more efficient than sequentially scanning all facts. However, it is not clear how much efficiency one gains or loses compared with conventional indexing techniques. Nevertheless, there is some empirical evidence that using partial orders instead of indices over simple properties speeds up access considerably – at least in the case of the nonstandard databases for chemical structures [28].

Finally, there is the question of what price we have to pay for semantic indexing at the time when instances are entered into the database. Again this depends on the expressiveness of the terminological logic used. In general, *realization*, i.e., computing the classes an object is an instance of, is not easier than subsumption. As a matter of fact, in a standard *open world* database, subsumption can be reduced to realization. On the other hand, realization is usually not much harder than subsumption since from a theoretical point of view a slight modification of the subsumption algorithm can be used [19]. For *closed world* databases the picture is sometimes different, however [32, Chap. 4].

[19] As a matter of fact, co-reference constraints over single-valued roles are computationally very well-behaved, which leads to a polynomial subsumption algorithm in CLASSIC – an insight gained from the analyses of so-called feature logics (see, e.g., [49]).

[20] It is an open problem whether consistency in a terminological logic containing no co-reference constraints is decidable if rules are added to the terminological logic.

5 Conclusion

Representation and reasoning with terminological logics, a subfield of knowledge representation, appears to have reached a point where the techniques developed in this field seem to be applicable to databases and information system. First of all, these logics support the representation of complex relationships, a necessity for any database model that intends to go beyond record structures. Second, reasoning in these logics can be exploited for various important tasks in database and information management, such as schema design, query evaluation and indexing. In particular, the last point seems to be very interesting since indexing can be done in a way which parallels the semantic structure of the schema, a reason for calling this technique *semantic indexing*. Third, the theoretical foundations of terminological logics have been analyzed to a depth that makes the results applicable to other areas in computer science. In particular, we were able to identify a problem in the object-oriented database model O_2 by applying results from the analyses of terminological logics.

Although this sounds quite promising, it will probably take some time before terminological logics will be applied in database and information systems. There are a number of important theoretical and practical problems which have to be solved first. For instance, the problem of finding efficient physical design methods for systems based on terminological logics has not been tackled yet. Also there is the problem of designing *complete* (in some sense) query languages.

Summarizing, although we do not have an instant solution for all problems appearing in the area of database and information management, we hope to have shown that the paradigm of terminological logics has something to offer to future information systems.

Appendix A: Syntax

KB Terminological Schema (TBox)

$$
\begin{array}{lcl}
\langle \textit{term-tell} \rangle & ::= & \langle \textit{concept-}\textsc{name} \rangle \quad :< \langle \textit{concept} \rangle \\
& | & \langle \textit{concept-}\textsc{name} \rangle \quad := \langle \textit{concept} \rangle \\
& | & \langle \textit{attribute-set-}\textsc{name} \rangle \quad := \langle \textit{attribute-set} \rangle \\
& | & \langle \textit{role-}\textsc{name} \rangle \quad :< \langle \textit{role} \rangle \\
& | & \textbf{disjoint}(\langle \textit{concept} \rangle, \langle \textit{concept} \rangle) \\
& | & \textbf{implies}(\langle \textit{concept} \rangle, \langle \textit{concept} \rangle) \\
\\
\langle \textit{term-ask} \rangle & ::= & \textbf{describe}(\langle \textit{term} \rangle, \langle \textit{term} \rangle) \\
& ::= & \textbf{subsumes}(\langle \textit{class} \rangle, \langle \textit{class} \rangle)
\end{array}
$$

KB Objects (ABox)

⟨*object-tell*⟩ ::= ⟨*object*-NAME⟩= ⟨*class-expression*⟩
⟨*object-ask*⟩ ::= ⟨*object*-VAR⟩ = **getall** ⟨*class-expression*⟩

⟨*class-expression*⟩ ::= ⟨*concept*⟩
| ⟨*attribute-set*⟩
| ⟨*concept*⟩ **with** ⟨*role*⟩ **:** ⟨*value-expression*⟩
⟨*value-expression*⟩ ::= ⟨*object*-NAME⟩
| **close(**⟨*value-expression*⟩**)**
| **(all** ⟨*class-expression*⟩**)**
| ⟨*value-expression*⟩ **and** ⟨*value-expression*⟩

Term Description Language (TDL)

⟨*term*⟩ ::= ⟨*class*⟩
| ⟨*role*⟩
⟨*class*⟩ ::= ⟨*concept*⟩
| ⟨*attribute-set*⟩
| ⟨*number-set*⟩

⟨*concept*⟩ ::= ⟨*concept*-NAME⟩
| ⟨*concept*⟩ **and** ⟨*concept*⟩
| **all(**⟨*role*-NAME⟩**,**⟨*class*⟩**)**
| **atleast(**⟨NUMBER⟩**,**⟨*role*-NAME⟩**)**
| **atmost(**⟨NUMBER⟩**,**⟨*role*-NAME⟩**)**
| **all1(**⟨*role*-NAME⟩**,**⟨*class*⟩**)**
| **anything** | **nothing**

⟨*role*⟩ ::= ⟨*role*-NAME⟩
| ⟨*role*⟩ **and** ⟨*role*⟩
| **domain(**⟨*concept*⟩**)**
| **range(**⟨*class*⟩**)**

⟨*attribute-set*⟩ ::= **aset(**⟨*attribute*-NAME⟩$^+$**)**
| **attribute**

⟨*number-set*⟩ ::= ⟨NUMBER⟩
| <⟨NUMBER⟩ | >⟨NUMBER⟩
| **number**

Non-Terminals containing the substrings NAME, NUMBER, or VAR indicate terminal names, numbers, or variables of the host language.

Appendix B: Semantics

In the following, we specify the set-theoretic semantics of the TDL. The symbols v, v_i are used for class terms, c, c_i for concept terms, a, a_i for attribute set terms, t, t_i for attribute value terms, n, n_i for number set terms, p, p_i for number terms, and r, r_i for role terms. $|S|$ denotes the cardinality of the set S. d, e are used to denote elements of V, and i denotes an element of I.

Definition 1 *Let D, A, and I be mutually disjoint sets, where D, the* domain, *is some arbitrary set, A is the set of attribute values, and I is the set of all integers. Let V be the union of D, A, and I. An* **extension function $\mathcal{E}$** *is a function mapping classes to subsets of V, concepts to subsets of D, attribute sets to subsets of A, attribute values to elements of A, number sets to subsets of I, numbers to elements of I, and roles to subsets of $D \times V$, such that for all nonnegative integers m:*

$$
\begin{aligned}
\mathcal{E}[\textbf{anything}] &= D \\
\mathcal{E}[\textbf{nothing}] &= \emptyset \\
\mathcal{E}[\mathsf{c}_1 \textbf{ and } \mathsf{c}_2] &= \mathcal{E}[\mathsf{c}_1] \cap \mathcal{E}[\mathsf{c}_2] \\
\mathcal{E}[\textbf{all}(\mathsf{r}, \mathsf{v})] &= \left\{ d \in D : \forall \langle d, e \rangle \in \mathcal{E}[\mathsf{r}] \Rightarrow e \in \mathcal{E}[\mathsf{v}] \right\} \\
\mathcal{E}[\textbf{atleast}(m, \mathsf{r})] &= \left\{ d \in D : |\{ e : \langle d, e \rangle \in \mathcal{E}[\mathsf{r}] \}| \geq m \right\} \\
\mathcal{E}[\textbf{atmost}(m, \mathsf{r})] &= \left\{ d \in D : |\{ e : \langle d, e \rangle \in \mathcal{E}[\mathsf{r}] \}| \leq m \right\} \\
\mathcal{E}[\textbf{all1}(\mathsf{r}, \mathsf{v})] &= \mathcal{E}[\textbf{all}(\mathsf{r}, \mathsf{v})] \cap \mathcal{E}[\textbf{atleast}(1, \mathsf{r})] \\
\\
\mathcal{E}[\mathsf{r}_1 \textbf{ and } \mathsf{r}_2] &= \mathcal{E}[\mathsf{r}_1] \cap \mathcal{E}[\mathsf{r}_2] \\
\mathcal{E}[\textbf{domain}(\mathsf{c})] &= \mathcal{E}[\mathsf{c}] \times V \\
\mathcal{E}[\textbf{range}(\mathsf{v})] &= D \times \mathcal{E}[\mathsf{v}] \\
\\
\mathcal{E}[\textbf{attribute}] &= A \\
\mathcal{E}[\textbf{aset}(\mathsf{t}_1 \ldots \mathsf{t}_n)] &= \{ \mathcal{E}[\mathsf{t}_i] : 1 \leq i \leq n \} \\
\\
\mathcal{E}[\textbf{number}] &= I \\
\mathcal{E}[> \mathsf{p}] &= \left\{ i \in I : i > \mathcal{E}[\mathsf{p}] \right\} \\
\mathcal{E}[< \mathsf{p}] &= \left\{ i \in I : i < \mathcal{E}[\mathsf{p}] \right\}
\end{aligned}
$$

Definition 2 *Let $\mathcal{E}$ be any extension function and τ any term-introduction or restriction. Let x_1 be a name and x_2 be some term and let c_1 and c_2 be two concept names that were introduced by $\mathsf{c}_i :< \ldots$. Then $\mathcal{E}$* **satisfies** *τ iff*

$$
\begin{aligned}
\mathcal{E}[\mathsf{x}_1] &= \mathcal{E}[\mathsf{x}_2] && \text{if } \tau \text{ is of the form } \mathsf{x}_1 := \mathsf{x}_2 \\
\mathcal{E}[\mathsf{x}_1] &\subseteq \mathcal{E}[\mathsf{x}_2] && \text{if } \tau \text{ is of the form } \mathsf{x}_1 :< \mathsf{x}_2 \\
\mathcal{E}[\mathsf{c}_1] \cap \mathcal{E}[\mathsf{c}_2] &= \emptyset && \text{if } \tau \text{ is of the form } \textbf{disjoint}(\mathsf{c}_1, \mathsf{c}_2)
\end{aligned}
$$

$\mathcal{E}$ satisfies a set of term-introductions or restrictions Θ iff $\mathcal{E}$ satisfies all elements of Θ.

Definition 3 *Let Θ be a set of term introductions and restrictions. Let x_1 and x_2 be two terms of the same syntactic category. Then x_1* **is subsumed by** *x_2 iff for all $\mathcal{E}$ that satisfy Θ:*

$$\mathcal{E}[\mathsf{x}_1] \subseteq \mathcal{E}[\mathsf{x}_2]$$

References

[1] S. Abiteboul and R. Hull. IFO: A formal semantic database model. *ACM Transactions on Database Systems*, 12(3):525–565, Dec. 1987.

[2] H. Aït-Kaci. *A Lattice-Theoretic Approach to Computations Based on a Calculus of Partially Ordered Type Structures.* PhD thesis, University of Pennsylvenia, Philadelphia, Pa., 1984.

[3] F. Baader. Terminological cycles in KL-ONE-based KR-languages. In *Proceedings of the 8th National Conference of the American Association for Artificial Intelligence*, Boston, Mass., 1990.

[4] H. W. Beck, S. K. Gala, and S. B. Navathe. Classification as a query processing technique in the CANDIDE semantic data model. In *Proceedings of the International Data Engineering Conference, IEEE*, pages 572–581, Los Angeles, Cal., Feb. 1989.

[5] S. Bergamaschi, L. Cavedoni, C. Sartori, and P. Tiberio. On taxonomic reasoning in E/R environments. In C. Batini, editor, *Proceedings of the 7th International Conference on Entity Relationship Approach*, pages 301–312, 1988.

[6] S. Bergamaschi, C. Sartori, and P. Tiberio. On taxonomic reasoning in conceptual design. Technical Report 68, CIOC-CNR, Bologna, Italy, Mar. 1990.

[7] H. Bergmann and M. Gerlach. QUIRK: Implementierung einer TBox zur Repräsentation von begrifflichem Wissen. WISBER Memo 11, 2nd ed., Project WISBER, Department of Computer Science, Universität Hamburg, Hamburg, Germany, June 1987.

[8] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: a structural data model for objects. In *Proceedings of the 1989 ACM SIGMOD International Conference on Mangement of Data*, pages 59–67, Portland, Oreg., June 1989.

[9] R. J. Brachman and H. J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of the 4th National Conference of the American Association for Artificial Intelligence*, pages 34–37, Austin, Tex., Aug. 1984.

[10] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In J. Sowa, editor, *Principles of Semantic Networks*. Morgan Kaufmann, Los Altos, Cal., 1990. To appear.

[11] R. J. Brachman, V. Pigman Gilbert, and H. 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, Los Angeles, Cal., Aug. 1985.

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

[13] M. L. Brodie. Future Intelligent Information Systems: AI and Database Technologies Working Together. In M. L. Brodie and J. Mylopoulos, editors, *Readings in Artificial Intelligence and Databases*. Morgan Kaufmann, San Mateo, Cal., 1988.

[14] Commission of the European Communities. Information Package for the Submission of Proposals to the European Strategic Programme for Research and Develpoment in Information Technology. Internal Information, DG XIII-A2, September 1989.

[15] F. Corella. Semantic retrieval and levels of abstraction. In L. Kerschberg, editor, *Expert Database Systems—Proceedings From the 1st International Workshop*, pages 91–114. Benjamin/Cummings, Menlo Park, Cal., 1986.

[16] M. Damiani, S. Bottarelli, M. Migliorati, and C. Peltason. Terminological Information Management in ADKMS. In *ESPRIT '90 Conference Proceedings*, Dordrecht, The Netherlands, 1990. Kluwer Academic Publishers. To appear.

[17] P. Devanbu, P. G. Selfridge, B. W. Ballard, and R. J. Brachman. A knowledge-based software information system. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 110–115, Detroit, Mich., Aug. 1989.

[18] J. Doyle and R. S. Patil. Two dogmas of knowledge representation: Language restrictions, taxonomic classifications, and the utility of representation services. Technical Memo MIT/LCS/TM-387.b, Laboratory for Computer

Science, Massachusetts Institute of Technology, Cambridge, Mass., Sept. 1989.

[19] B. Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In H. Marburger, editor, *GWAI-90. 14th German Workshop on Artificial Intelligence*. Springer-Verlag, Berlin, Germany, 1990.

[20] B. Hollunder and W. Nutt. Subsumption algorithms for concept description languages. DFKI Report RR-90-04, German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany, 1990.

[21] T. S. Kaczmarek, R. Bates, and G. Robins. Recent developments in NIKL. In *Proceedings of the 5th National Conference of the American Association for Artificial Intelligence*, pages 978–987, Philadelphia, Pa., Aug. 1986.

[22] W. Kent. Limitations of record-based information models. *ACM Transactions on Database Systems*, 4(1):107–131, 1979.

[23] C. Kindermann. Class instances in a terminological framework – an experience report. In H. Marburger, editor, *GWAI-90. 14th German Workshop on Artificial Intelligence*. Springer-Verlag, Berlin, Germany, 1990.

[24] C. Kindermann and P. Randi. Object Recognition and Retrieval in the BACK System. KIT Report 83, Department of Computer Science, Technische Universität Berlin, Germany, To Appear.

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

[26] C. Lécluse, P. Richard, and F. Velez. O_2, an object-oriented data model. In *Proceedings of the 1988 ACM SIGMOD International Conference on Mangement of Data*, pages 424–433, Chicago, Ill., 1988.

[27] C. Lécluse, P. Richard, and F. Velez. Modelling complex structures in object-oriented databases. In *Proceedings of the 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database-Systems*, pages 360–367, Mar. 1989.

[28] R. A. Levinson. A self-organizing pattern retrieval system for graphs. In *Proceedings of the 4th National Conference of the American Association for Artificial Intelligence*, pages 203–206, Austin, Tex., Aug. 1984.

[29] R. MacGregor. A deductive pattern matcher. In *Proceedings of the 7th National Conference of the American Association for Artificial Intelligence*, pages 403–408, Saint Paul, Minn., Aug. 1988.

[30] R. MacGregor and 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.

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

[32] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Germany, 1990.

[33] B. Nebel. Terminological cycles: Semantics and computational properties. In J. Sowa, editor, *Principles of Semantic Networks*. Morgan Kaufmann, San Mateo, Cal., 1990. To appear.

[34] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990. A preliminary version is available as IWBS Report 82, IBM Germany Scientific Center, IWBS, Stuttgart, Germany, October 1989.

[35] B. Nebel and G. Smolka. Representation and reasoning with attributive descriptions. In K.-H. Bläsius, U. Hedtstück, and C.-R. Rollinger, editors, *Sorts and Types in Artificial Intelligence*. Springer-Verlag, Berlin, Germany, 1990. To appear. Also available as IWBS Report 81, IBM Germany Scientific Center, IWBS, Stuttgart, Germany, September 1989.

[36] J. Heinsohn and B. Owsnicki-Klewe. Probabilistic inheritance and reasoning in hybrid knowledge representation systems. In W. Hoeppner, editor, *GWAI-88. 12th German Workshop on Artificial Intelligence*, pages 51–60. Springer-Verlag, Berlin, West Germany, 1988.

[37] 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. An extended version including a KANDOR system description is available as AI Technical Report No. 37, Palo Alto, Cal., Schlumberger Palo Alto Research, October 1984.

[38] P. F. Patel-Schneider. A four-valued semantics for terminological logics. *Artificial Intelligence*, 38(3):319–351, Apr. 1989.

[39] P. F. Patel-Schneider. Undecidability of subsumption in NIKL. *Artificial Intelligence*, 39(2):263–272, June 1989.

[40] P. F. Patel-Schneider, R. J. Brachman, and H. J. Levesque. ARGON: Knowledge representation meets information retrieval. In *Proceedings of the 1st Conference on Artificial Intelligence Applications*, pages 280–286, Denver, Col., 1984.

[41] P. F. Patel-Schneider, B. Owsnicki-Klewe, A. Kobsa, N. Guarino, R. MacGregor, W. S. Mark, D. McGuinness, B. Nebel, A. Schmiedel, and J. Yen. Term subsumption languages in knowledge representation. *The AI Magazine*, 11(2):16–23, 1990.

[42] C. Peltason, A. Schmiedel, C. Kindermann, and J. Quantz. The BACK system revisited. KIT Report 75, Department of Computer Science, Technische Universität Berlin, Germany, Sept. 1989.

[43] J. Quantz. Draft Paper. KIT Report 84, Department of Computer Science, Technische Universität Berlin, Germany, To Appear.

[44] J. Quantz and C. Kindermann. Implementation of the BACK system version 4. KIT Report 78, Department of Computer Science, Technische Universität Berlin, Germany, Sept. 1990.

[45] K. Schild. Towards a theory of frames and rules. KIT Report 76, Department of Computer Science, Technische Universität Berlin, Germany, Dec. 1989.

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

[47] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with unions and complements. *Artificial Intelligence*, 1990. To appear. A preliminary version of this paper is available as IWBS Report 68, IBM Germany Scientific Center, IWBS, Stuttgart, Germany, June 1989.

[48] A. Schmiedel. A temporal terminological logic. In *Proceedings of the 8th National Conference of the American Association for Artificial Intelligence*, pages 640–645, Boston, Mass., 1990.

[49] G. Smolka. A feature logic with subsorts. *Journal of Logic Programming*, 1990. To appear. Also available as LILOG Report 33, IBM Germany Scientific Center, IWBS, Stuttgart, Germany, May 1988.

[50] F. N. Tou, M. D. Williams, R. E. Fikes, A. Henderson, and T. Malone. RABBIT: An intelligent database assistant. In *Proceedings of the 2nd National Conference of the American Association for Artificial Intelligence*, pages 314–318, Pittsburgh, Pa., Aug. 1982.

[51] K. von Luck, B. Nebel, C. Peltason, and A. Schmiedel. The anatomy of the BACK system. KIT Report 41, Department of Computer Science, Technische Universität Berlin, Germany, Jan. 1987.

Acknowledgement

We are grateful to Sonia Bergamaschi and the colleagues of our project groups for helpful comments on earlier drafts.