Terminological Reasoning by Query Evaluation: A Formal Mapping of a Terminological Logic to an Object Data Model*

U. Reimer and **P. Lippuner**
Swiss Life
Information Systems Research Group
CH–8022 Zürich Switzerland
⟨name⟩@swssai.uu.ch

M. Norrie and **M. Rys**
Swiss Federal Institute of Technology (ETH)
Dept. of Computer Science
CH–8092 Zürich, Switzerland
⟨name⟩@inf.ethz.ch

Abstract

The paper starts by giving concise introductions into the terminological logic FRM and the object data model COCOON. It then briefly outlines a semantic-preserving mapping from FRM class descriptions to COCOON types and classes and shows how the terminological inference of classification is mapped to a set of equivalent COCOON queries. Since these queries can (mostly) be submitted as a whole to the underlying database system we can take full advantage of all the results on query optimisation, on providing efficient physical access structures, as well as on parallelisation that are available in the database area to make terminological inferences more efficient. This will play a crucial role in realising knowledge base systems capable of dealing with very large knowledge bases.

1 Introduction

The fields of knowledge representation and databases are converging: The former is more and more concerned with efficiency for supporting large knowledge bases, while the latter is increasingly interested in providing higher representation constructs that better serve the construction of a domain model. Consequently, it seems to be a fruitful endeavour to combine the approaches of both areas. Our approach to combining the strengths of knowledge representation and database approaches takes advantage of the conceptual similarity of terminological logics and object data models. We realise a knowledge base system by mapping a terminological logic to an object data model which has an efficient implementation on top of a relational storage system [NRL+94]. To ensure that the potential for optimisation provided by the database system will really be available for the terminological system, the mapping from terminological structures to object structures preserves as much of the semantics of the terminological logic as possible.

There are a few former approaches to mapping terminological logics (or frame models) to data models. In the mappings to the relational data model described in [HMM87] and [SB89], one frame structure corresponds to several database structures. As a consequence, there is little correspondence between the representation structures the database system manages and the original frame structures. Therefore, the database system is deprived of most of its optimisation capabilities.

Another former approach to mapping a frame model to a data model is described in [RS89]. It preserves the frame structure as a complex object structure in the nested relational model to which it is mapped. The major drawback with that approach results from the lack of type polymorphism in the nested relational model because this makes it difficult to host the concept hierarchy of the frame model.

Some of the existing data models that support complex objects provide constructs that are similar to constructs of a terminological logic (e.g. [KL89, BGL+91]). Their main difference is that they do not provide terminological reasoning services (besides inheritance), although offering deductive question answering.

Sections 2 and 3 introduce the basic concepts of the terminological logic FRM and the object data model COCOON used in our approach. Section 4 describes the mapping of the terminological inference of classification to COCOON queries and illustrates the mapping of class descriptions of FRM to type and class constructs of COCOON. Section 5 concludes the paper.

2 Basic Constructs of the Terminological Logic FRM

The syntactic constructs and the model-theoretic semantics of FRM [RL95, Rei85] are given in Figure 1. We distinguish two kinds of relations, namely properties and semantic relationships. A *property* denotes a relation between individuals and string or integer values (see the constructs **all-p** and **exist-v**). A *semantic relationship* denotes a relation between individuals (see the constructs **all-r**, **exist-c** and **exist-i**).

Unlike other terminological logics, FRM only al-

*The work reported here was supported by the Swiss Priority Programme for Computer Science (Schwerpunktprogramm Informatik) under grant No. 5003-034347.

Syntactic form	Interpretation
$a \doteq t$	$\varepsilon[a] = \varepsilon[t]$
$a \dot{\leq} t$	$\varepsilon[a] \subseteq \varepsilon[t]$
(**and** $c_1 \ldots c_n$)	$\bigcap_{i=1}^{n} \varepsilon[c_i]$
(**all-p** $prop\, r_1 \ldots r_n$)	$\{x \in D \mid \exists y : \langle x, y\rangle \in \varepsilon[prop] \wedge$ $\forall y : (\langle x, y\rangle \in \varepsilon[prop] \Rightarrow y \in (\varepsilon[r_1] \cup \ldots \cup \varepsilon[r_n]))\}$
(**all-r** $rel\, c_1 \ldots c_n$)	$\{x \in D \mid \exists y : \langle x, y\rangle \in \varepsilon[rel] \wedge$ $\forall y : (\langle x, y\rangle \in \varepsilon[rel] \Rightarrow y \in (\varepsilon[c_1] \cup \ldots \cup \varepsilon[c_n]))\}$
(**exist-v** $prop\, v$)	$\{x \in D \mid \langle x, v\rangle \in \varepsilon[prop]\}$
(**exist-c** $rel\, c$)	$\{x \in D \mid \exists y \in \varepsilon[c] : \langle x, y\rangle \in \varepsilon[rel]\}$
(**exist-i** $rel\, i$)	$\{x \in D \mid \langle x, \varepsilon[i]\rangle \in \varepsilon[rel]\}$
(**at-least** $rp\, n$)	$\{x \in D \mid \lVert\{y \in D : \langle x, y\rangle \in \varepsilon[rp]\}\rVert \geq n\}$
(**at-most** $rp\, n$)	$\{x \in D \mid \lVert\{y \in D : \langle x, y\rangle \in \varepsilon[rp]\}\rVert \leq n\}$
thing	D

Figure 1: Syntax and Semantics of FRM

lows class descriptions that refer to other classes by their name and not by including their structure. This restriction does not have any effect on the expressiveness. It only requires that every concept class being used must independently be introduced and assigned a name. However, FRM provides an extended syntax for class descriptions that may occur as queries to a knowledge base (see [RLN+95]).

Terminological logics have evolved from frames and semantic networks. One difference is that terminological logics offer a greater flexibility for formulating class descriptions. This syntactic flexibility makes it difficult to define a mapping of a terminological logic to any data model because there is no fixed concept structure. However, any class description formulated in FRM can be interpreted as a frame structure, i.e., as consisting of *slots* and *slot entries*. Thus, the FRM constructs **all-p** and **all-r** correspond to slots. We call **all-p** *property slots* and **all-r** *relationship slots*. The construct **exist-c** specifies a concept class as a slot entry and **exist-i** an individual as a slot entry. **exist-v** sets a value as a slot entry in a property slot.

Since the syntax of FRM as introduced above allows to introduce a slot entry without (explicitly) defining a corresponding slot, we must consider the implications shown in Figure 2 to properly interpret an FRM class descriptions as a frame with slots and entries. For example, the first implication given there states that the introduction of a slot entry (by the **exist-c** construct) implicitly introduces a slot (as expressed by the **all-r** construct). Thus, the following two class definitions would be semantically equivalent:

$g \doteq$ (**exist-c** manufactured-by big-company)

$h \doteq$ (**and** (**all-r** manufactured-by **thing**)
(**exist-c** manufactured-by big-company))

In Section 4 we assume the existence of a normalisation function *norm* that augments a class description with all implied features. For example, with respect to the class descriptions g and h above we get the equivalence $\varepsilon[norm(g)] = \varepsilon[norm(h)]$. The normalisation function covers many further implications not shown in Figure 2.

3 Basic Constructs of the Object Model COCOON

The constructs of the terminological logic FRM are mapped to the object data model COCOON and its associated language COOL [SLR+94]. COCOON resembles a functional data model in that object properties are modelled as single- and multi-valued functions. However, it also supports the dynamic grouping of objects into a class hierarchy based on predicates over object properties (cf. Fig.3).

The COOL query and update language is based on an algebra of operations over classes and can be considered as an extension of the nested relational algebra [ScS91]. The basic operations are *select*, *project*, *extend* (provides object type extension) and the set-based operations of *union*, *intersection* and *difference* (cf. Fig.6). The language also supports type guards for dynamic type checking.

Update operations may change the properties, class memberships and even the structure of database objects during their lifetime. Since COCOON allows objects to be grouped into classes based on their properties, objects are automatically reclassified within the class hierarchy after updates.

4 Mapping The Classification Inference

Figure 3 gives an example of our mapping of FRM concept descriptions to COCOON types and classes. Due to the limited space, the mapping is not described in this paper but we provide remarks where appropriate (for a detailed des cription see [RLN+95]). In the following, we give a brief description of how the classification inference of FRM is mapped to appropriate COCOON queries.

Let $\preceq$ denote the subsumption relation and let $\lhd$ be its transitive reduction. The concept hierarchy can then be conceived of as an undirected graph where the nodes represent all introduced concepts (C) and the edges represent the relation $\lhd$. Thus, classifying a concept c means to determine the following two sets:

$$L_c = \{l \in C \mid l \lhd c\} \qquad U_c = \{u \in C \mid c \lhd u\}$$

(**exist-c** r c)	implies	(**all-r** r **thing**)	
(**exist-i** r i)	implies	(**all-r** r **thing**)	
(**exist-v** p v)	implies	(**all-p** p $*$)	
(**at-least** rp n)	implies	(**all-r** rp **thing**)	if rp is a relation, (**all-p** rp $*$) else
(**at-most** rp n)	implies	(**all-r** rp **thing**)	if rp is a relation, (**all-p** rp $*$) else

Figure 2: Some of the Implications being Considered by a Normalisation Function for Class Descriptions

Sun-Del $\dot{\leq}$ (**and** (**all-p** costs $[0, 100000]$)
(**at-most** costs 1)
(**all-r** receives Company Person)
(**at-most** receives 1)
(**all-r** goods Workstation)
(**all-r** delivers Company)
(**exist-i** delivers Sun))

define type sun-del =
costs : **integer**,
receives : **objects**,
goods : **set of objects**,
delivers : **set of objects**;

define class Sun-Del : sun-del **where**
costs ≥ 0 **and** costs ≤ 100000 **and**
receives $\subseteq$ (Person $\cap$ Company) **and**
delivers $\subseteq$ Company **and**
Sun $\in$ delivers **and**
goods $\subseteq$ Workstation;

Figure 3: A Concept Class Introduction (left) and its corresponding type and class definitions (right)

The elements of L_c are called the *most general subconcepts* of c, and the elements of U_c the *most specific superconcepts* of c. As the computation of the two sets L_c and U_c is symmetric we only discuss the ca se of L_c. It can be computed by traversing the concept hierarchy bottom-up and determing all subconcepts of c that have no superconcept which is a subconcept of c. This traversal can be done by different variations of the common ıt depth-first search algorithm [BHN+92]. Apart from such modifications, the main algorithm used in existing systems is always the same: Classification is done by traversing the concept hierarchy while testing subsumption relations.

In our approach, we compute the set L_c completely differently. Instead of searching the concept hierarchy for the appropriate position we obtain L_c as the result of two COCOON queries:

1. The first query Q_1 yields all subconcepts of c: $L_c^+ = \{l \in C \mid l \preceq c\}$
2. The second query Q_2 yields the most general concepts from L_c^+: $L_c = \{l \in L_c^+ \mid l \lhd c\}$

To formulate Q_1 we first have a closer look at the subsumption relation of FRM. There is a well-defined set of update operations that, when applied to a concept c, lead to a more specific concept $\tilde{c}$, i.e. $\tilde{c} \preceq c$:

I: Concept Level (applicable to any concept):
  * Add a new slot to the concept.

II: Slot Level (applicable to any slot of a given concept):
  - ◇ In case of a property slot: Restrict the set of permitted entries to a subset. In case of a relationship slot (**all-r** r $c_1 \ldots c_n$): Remov e one or more of the range classes $c_1, \ldots, c_n$ and/or specialise a range class.
  - ◇ Add further slot entries.
  - ◇ In case of a relationship slot and a class occurring as an entry (**exist-c** construct), specialise this class, or substitute it by an instance of it, thus substituting the **exist-c** construct with an **exist-i** construct.
  - ◇ Restrict the cardinality to a smaller interval.

We are now able to define the subsumption relation *syntactically* by referring to the concept descriptions (instead of the usual model-theoretic definition). To this end, we require for $c_1 \preceq c_2$ to hold that c_1 can be obtained from c_2 by applying one or more of the above operations. The corresponding definition (in a declarative fashion) is given in Figure 4. It makes use of the notation introduced in Table 1 and of the predicate $inst(i, c)$ which is true if i is an instance of the class c. The completeness of this subsumption definition very much depends on the normalisation function discussed in Section 2. ifficult to be handled. Since we are still working on the normalisation function, our subsumption algorithm is currently not complete.

Based on the syntactic definition of the subsumption relation it is now straightforward to formulate the COCOON query Q_1 that determines L_c^+ for a given concept description c. It consists of an intersection of mutually independent subqueries that can be computed concurrently. Each subquery deals with one of the slots of the concept to be classified. The resulting query schema for a slot s_i is shown in Figure 5[1]. As the whole condition (ISA) is mapped

[1]The query is formulated using functions defined for objects in the meta-schema, each object being a description of one object class in the COCOON database. We do not go into the details of the meta-schema here and use the function names of Table 1 with a subscript "ms" so that the correspondence to definition (ISA) can be seen. The functions *supc* and *subc* yield all superclasses,

$$c_1 \preceq c_2 \Leftrightarrow \forall s \in slots(c_2) : (s \in slots(c_1) \ \wedge \ prange(c_1, s) \subseteq prange(c_2, s) \ \wedge \qquad \text{(ISA)}$$
$$\forall r_1 \in rrange(c_1, s) : \exists r_2 \in rrange(c_2, s) : r_1 \preceq r_2 \ \wedge$$
$$\forall e_2 \in entries\text{-}c(c_2, s) : (\exists e_1 \in entries\text{-}c(c_1, s) : e_1 \preceq e_2 \vee$$
$$\exists e_1 \in entries\text{-}i(c_1, s) : inst(e_1, e_2)) \ \wedge$$
$$entries\text{-}i(c_1, s) \supseteq entries\text{-}i(c_2, s) \ \wedge$$
$$entries\text{-}v(c_1, s) \supseteq entries\text{-}v(c_2, s) \ \wedge$$
$$minCard(c_1, s) \geq minCard(c_2, s) \ \wedge$$
$$maxCard(c_1, s) \leq maxCard(c_2, s))$$

Figure 4: Syntactic Definition of the Subsumption Relationship

$slots(c)$	$= \{ rp \mid c \dot{\leq}$ (**and** ... (**all-r** $rp\ c_1 \ldots c_n$) ...) or $c \dot{\leq}$ (**and** ... (**all-p** $rp\ range$) ...) $\}$
$rrange(c, s)$	$= \{ c_1, \ldots, c_n \mid c \dot{\leq}$ (**and** ... (**all-r** $s\ c_1 \ldots c_n$) ...) $\}$
$prange(c, s)$	$= r$, where $c \dot{\leq}$ (**and** ... (**all-p** $s\ r$) ...)
$entries\text{-}c(c, s)$	$= \{ c_e \mid c \dot{\leq}$ (**and** ... (**exist-c** $s\ c_e$) ...) $\}$
$entries\text{-}i(c, s)$	$= \{ i \mid c \dot{\leq}$ (**and** ... (**exist-i** $s\ i$) ...) $\}$
$entries\text{-}v(c, s)$	$= \{ v \mid c \dot{\leq}$ (**and** ... (**exist-v** $s\ v$) ...) $\}$
$minCard(c, s)$	$= n$, where $c \dot{\leq}$ (**and** ... (**at-least** $s\ n$) ...)
$maxCard(c, s)$	$= n$, where $c \dot{\leq}$ (**and** ... (**at-most** $s\ n$) ...)

Table 1: Functions for Accessing Parts of a (Normalised) Concept Definition (analogously for $\doteq$)

to a single query we can take full advantage of the query optimiser in the underlying database system.

Since classification is an inference on the structure of concept descriptions, this query schema accesses the *meta-schema*. As discussed in Section 4, most of the information about an FRM concept is encoded in the class predicate of the corresponding object class. However, the class predicate is just a string-valued attribute in the meta-schema and can only be queried as a whole. This means that certain structures of a concept description cannot be queried directly. Therefore, we extended the COCOON meta-schema by an application-specific part where we store the information about the concept classes in a well-structured way (in a certain sense, we model FRM in COCOON). While the meta-schema extension has to be administered by the mapping algorithm the "standard" part of the meta-schema is updated automatically by the COCOON system.

Figure 6 shows part of the COCOON query that returns the set of all subconcepts of 'Sun-Del' (see Figure 3) in the current knowledge base. This part completely deals with the slot 'delivers'.

As introduced above, query Q_2 of our classification inference is concerned with extracting the most general subconcepts L_c from the set of all subconcepts L_c^+. Assuming that the variable L holds the result from Q_1 (i.e., the set L_c^+), query Q_2 can be formulated in COCOON as: $\textbf{select}[\emptyset = supc(l) \cap L](l : L)$.

5 Conclusions

We proposed to map terminological inferences to queries of an object data model. The resulting, complex queries can be split into several subqueries and evaluated independently. Thus, besides making use of more standard database optimisation techniques, like query optimisation and specialised access structures, we can also exploit parallelisation. We expect this implementation of subsumption to be much more efficient for large knowledge bases than a standard implementation, while for small knowledge bases the overhead introduced nd the database system will be greater than the efficiency gained by the optimisations.

We are currently implementing the mappings described in this paper. To provide efficient retrieval and update services the object model COCOON is mapped to a relational storage system which makes use of massive data replication (to minimise retrieval costs) and parallelisation of update operations (to minimise update costs). In a subsequent step, we will set up experiments to evaluate the efficiency gain and to pinpoint further possible improvements.

or subclasses, resp. The function $objects(c)$ returns all objects in the class c.

References

[BGL+91] S. Benzschawel, E. Gehlen, M. Ley, T. Ludwig, A. Maier, B. Walter: LILOG-DB: Database Support for Knowledge Based Systems. In: O. Herzog, C.-R. Rollinger (eds): Text Understanding in LILOG. Berlin: Springer-Verlag, 1991, pp.501-594.

[BHN+92] F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, E. Franconi: An Empirical Analysis of Optimization Techniques for Termninological Representation Systems. In: B. Nebel, C. Rich, W. Swartout (eds): Principles of Knowledge Representation and Reasoning. Proc. of the Third Int. Conf., 1992, pp.270-281.

[HMM87] Th. Härder, N. Mattos, B. Mitschang: Abbildung von Frames auf neuere

$$
\begin{array}{l}
\mathbf{select}\big[\emptyset \neq \mathbf{select}[name(s) = name(s_i) \textbf{ and} \\
\qquad prange_{ms}(s) \subseteq prange_{ms}(s_i)]\ (s : slots(c_{sub}))\big](c_{sub}\text{:Concepts}) \quad \cap \\
\mathbf{select}\big[\emptyset \neq \mathbf{select}[name(s) = name(s_i) \textbf{ and} \\
\qquad rrange_{ms}(s) = \mathbf{select}[\emptyset \neq (supc(r) \cup r) \cap rrange_{ms}(s_i)](r : rrange_{ms}(s))] \\
\qquad\quad (s : slots(c_{sub}))\big](c_{sub}\text{:Concepts}) \quad \cap \\
\mathbf{select}\big[\emptyset \neq \mathbf{select}[name(s) = name(s_i) \textbf{ and} \\
\qquad entries\text{-}c_{ms}(s_i) = \\
\qquad \mathbf{select}\ [\emptyset \neq (subc(e) \cup objects(e) \cup e) \cap entries\text{-}c_{ms}(s)](e : entries\text{-}c_{ms}(s_i))] \\
\qquad\quad (s : slots(c_{sub}))\big](c_{sub}\text{:Concepts}) \quad \cap \\
\mathbf{select}\big[\emptyset \neq \mathbf{select}[name(s) = name(s_i) \textbf{ and} \\
\qquad entries\text{-}i_{ms}(s) \supseteq entries\text{-}i_{ms}(s_i)]\ (s : slots(c_{sub}))\big](c_{sub}\text{:Concepts}) \quad \cap \\
\mathbf{select}\big[\emptyset \neq \mathbf{select}[name(s) = name(s_i) \textbf{ and} \\
\qquad entries\text{-}v_{ms}(s) \supseteq entries\text{-}v_{ms}(s_i)]\ (s : slots(c_{sub}))\big](c_{sub}\text{:Concepts}) \quad \cap \\
\mathbf{select}\big[\emptyset \neq \mathbf{select}[name(s) = name(s_i) \textbf{ and} \\
\qquad minCard_{ms}(s) \geq minCard_{ms}(s_i)]\ (s : slots(c_{sub}))\big](c_{sub}\text{:Concepts}) \quad \cap \\
\mathbf{select}\big[\emptyset \neq \mathbf{select}[name(s) = name(s_i) \textbf{ and} \\
\qquad maxCard_{ms}(s) \leq maxCard_{ms}(s_i)]\ (s : slots(c_{sub}))\big](c_{sub}\text{:Concepts})
\end{array}
$$

Figure 5: Query Schema for a Slot s_i of a Concept Description c (Used to Determine all Subconcepts of c)

$$
\begin{array}{l}
\mathbf{select}\big[\emptyset \neq \mathbf{select}[name(s) = \text{`delivers'} \textbf{ and} \\
\qquad allr_{ms}(s) = \mathbf{select}[\emptyset \neq (supc(r) \cup r) \cap \{\text{`Company'}\}](r : allr_{ms}(s))] \\
\qquad\quad (s : slots(c_{sub}))\big](c_{sub}\text{:Concepts}) \quad \cap \\
\mathbf{select}\big[\emptyset \neq \mathbf{select}[name(s) = \text{`delivers'} \textbf{ and} \\
\qquad exists\text{-}i_{ms}(s) \supseteq \{\text{`Sun'}\}]\ (s : slots(c_{sub}))\big](c_{sub}\text{:Concepts}) \quad \cap \\
\mathbf{select}\big[\emptyset \neq \mathbf{select}[name(s) = \text{`delivers'} \textbf{ and} \\
\qquad maxCard_{ms}(s) \leq 1]\ (s : slots(c_{sub}))\big](c_{sub}\text{:Concepts}) \quad \cap \\
\mathbf{select}\big[\emptyset \neq \mathbf{select}[name(s) = \text{`receives'} \textbf{ and} \ldots
\end{array}
$$

Figure 6: Part of the COCOON Query that Yields all Subconcepts of Concept 'Sun-Del' in Figure 3

Datenmodelle. In: K. Morik (Ed): GWAI-87. 11th German Workshop on Artificial Intelligence. Berlin: Springer, 1987, pp. 396-405.

[KL89] M. Kifer, G. Lausen: F-Logic: A Higher-Order Language for Reasoning about Objects, Inheritance, and Scheme. In: Proc. of the ACM SIGMOD Int. Conf. on Management of Data, 1989, pp.134-146.

[NRL+94] M. C. Norrie, U. Reimer, P. Lippuner, M. Rys, H.-J. Schek: Frames, Objects and Relations: Three Semantic Levles for Knowledge Base Systems. In: Proc. Workshop on Reasoning about Structured Objects: Knowledge Representation meets Databases (KRDB-94), Saarbrücken, Germany, 1994, pp.53-57.

[Rei85] U. Reimer: A Representation Construct for Roles. In: Data & Knowledge Engineering, Vol.1, No.3, 1985, pp.233-251.

[RL95] U. Reimer, P. Lippuner: Syntax und Semantik von FRM. Working Paper, 1995, Information Systems Research Group, Swiss Life, CH–8022 Zurich.

[RLN+95] U. Reimer, P. Lippuner, M. Norrie, M. Rys: Terminological Reasoning by Query Evaluation: A Formal Mapping of a Terminological Logic to an Object Data Model. Extended Version. Technical Report, 1995, Information Systems Research Group, Swiss Life, CH–8022 Zurich.

[RS89] U. Reimer, H.J. Schek: A Frame-Based Knowledge Representation Model and its Mapping to Nested Relations. In: Data & Knowledge Engineering, Vol.4, No.4, 1989, pp.321-352.

[SB89] R. Studer, S. Börner: An Approach to Manage Large Inheritance Networks within a DBS Supporting Nested Relations. In: S. Abiteboul, P.C. Fischer, H.-J. Schek (Eds): Nested Relations and Complex Objects in Databases. Berlin: Springer, 1989, pp. 229-239.

[ScS91] M.H. Scholl, H.-J. Schek: From Relations and Nested Relations to Object Models. In: M.S. Jackson, A.E. Robinson (eds): Aspects of Database Systems. Butterworth-Heinemann, 1991, pp.202-225.

[SLR+94] M.H. Scholl, C. Laasch, C. Rich, H.-J. Schek, M. Tresch: The COCOON Object Model, Technical Report 211 (revised version), Dept. of Computer Science, ETH Zurich, CH-8092 Zurich.