Making $\mathcal{CATS}$ out of kittens: description logics with aggregates

Giuseppe De Giacomo and Maurizio Lenzerini
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italia
{degiacomo,lenzerini}@assi.dis.uniroma1.it

Abstract

Based on the research done in the last decade, attempts have been made to propose description logics as unifying formalisms for the various class-based representation languages used in different areas. These attempts have made apparent that sound, complete, and decidable description logics still suffer from several limitations, regarding modeling classes of aggregate objects, expressing general inclusion axioms, and the ability of navigating links between classes. In this paper, we propose a powerful description logic overcoming the above limitations and we show that its reasoning tasks are decidable in worst case exponential time.

1 Introduction

Description logics are AI formalisms that allow one to represent domain knowledge by focusing on classes of objects [Brachman,1977] and their relationships [Woods,1975], and offering specialized inferences on the class structure.

The research developed in the last decade offers a quite complete picture of several issues related to the expressive power of the logics and the computational complexity of the reasoning tasks (see [Woods and Schmolze,1992]). Based on the outcome of this research, attempts have been made to propose description logics as unifying formalisms for the various class-based representation languages used in different areas, such as semantic networks, feature logics, conceptual and object-oriented database models, type systems, and other formalisms used in software engineering [Bergamaschi and Sartori,1992; Piza *et al.*,1992; Borgida,1992; Calvanese *et al.*,1994; Schreiber *et al.*,1993]. However, these attempts have made apparent that description logics equipped with sound, complete, and terminating reasoning procedures still suffer from several limitations that are not acceptable when representing complex domains in the different fields mentioned above. Here is a list of the most important limitations.

- The domain of interpretation is flat, in the sense that the logics consider the world as constituted by elementary objects (grouped in concepts) and binary relations between them. One consequence of this property is that N-ary relations are not supported (an exception is the logic proposed in [Schmolze,1989], for which no complete decision procedure was proposed). In fact, N-ary relations have been shown to be important in several contexts (see [Catarci and Lenzerini,1993]), especially in databases and natural language. For example, 'exam' is correctly modeled as a ternary relation over 'student', 'professor' and 'course'. Note that supporting N-ary relations means that the logic offers suitable mechanisms for their definition and characterization. For example, one has to ensure that no pair of 'exam' instances connect the same triple of objects; also, one may want to assert that students linked to graduate courses by the relation exam are graduate students. These kinds of properties cannot be represented by simply modeling the N-ary relation in terms of N binary relations.

- Usually, general inclusion axioms are not supported. Although inclusion axioms are essential when we want to assert properties of classes and relations, as required in complex domains, most of the research on description logics either deals with class descriptions only, or impose severe restrictions, such as acyclicity, on axioms. Exceptions are, for example, [Nebel,1991; Baader,1991; Schild,1991; De Giacomo and Lenzerini,1994; Buchheit *et al.*,1993]. An important outcome of this research is that reasoning with axioms is computationally hard, even for the simplest description logics (weaker than $\mathcal{FL}^-$). All these works, however, limit their attention to axioms on concepts, and do not consider the problem of expressing inclusion axioms on relations.

- Relationships between classes are generally described by means of poor representation mechanisms. In fact, when trying to use description logics for capturing representation formalisms used in different fields, one realizes that at least three features are essential: the ability of *navigating* relationships (say of a semantic network or an entity-relationship schema) in both directions; the ability of stating cardinality constraints of general forms on relationships; the possibility of conceiving relationships as sets, thus applying set theoretic operators on

them (including the notorious role value map [Woods and Schmolze,1992]).

The aim of the present work is to devise a description logic, called $\mathcal{CATS}$, that finally addresses the above issues. The basic ingredients of $\mathcal{CATS}$ are classes and links. In contrast to traditional description logics, *classes* are abstractions not only for a set of individuals (corresponding to the usual notion of concept, called simple class here), but also for sets that have aggregates as instances (called aggregate classes). There are two types of aggregates: property aggregates and instance aggregates. A *property aggregate* is an abstraction for an object that is considered as an aggregation of other objects, one for each attribute belonging to a specified set [Smith and Smith,1977]. A typical example of such an aggregate is a date, which is seen as an aggregation of three objects, one for the attribute day, one for the attribute month, and one for the attribute year. Another example of property aggregate is an exam, which again is seen as an aggregation of three objects (one professor, one student and one course). This makes clear that N-ary relations can be modeled as classes whose instances are aggregates. An *instance aggregate* is an abstraction of a group of other objects belonging to a certain class [Brodie and Ridjanovic,1984]. A typical example of such an aggregate is a team, which can be seen as a group of players. Like any other description logics, $\mathcal{CATS}$ allows one to form *complex classes* by applying suitable constructors to both simple and aggregate classes. Notably, $\mathcal{CATS}$ includes a form of role value map, and the most general form of number restrictions (called qualified).

Links are abstractions for atomic, basic, and complex relationships between classes. An *atomic link* (denoted simply by a name, and also called attribute) is the most elementary mean for establishing a relationship between classes. A *basic* link is formed by applying certain constructors (like inverse, union, intersection and difference) to atomic links. A *complex* link is formed by applying more complex constructors (like chaining, transitive closure, and identity) to basic links.

A knowledge base in $\mathcal{CATS}$ is simply a set of inclusion axioms. We point out that $\mathcal{CATS}$ allows inclusion assertions to be stated on classes of all kinds (simple, aggregate and complex), and on basic links, with no limitation (for example on cycles). A particular care is put in devising $\mathcal{CATS}$ so that its reasoning tasks remain decidable and even with the same computational complexity as the simplest description logics where inclusion axioms are allowed. Indeed, making use of the results in [De Giacomo and Lenzerini,1994], we have proved that computing logical implication (and satisfiability) in $\mathcal{CATS}$, is both EXPTIME-hard and decidable with exponential time in the worst case.

2 The description logic $\mathcal{CATS}$

As we said above, the language of $\mathcal{CATS}$ supports classes and links. Classes are partitioned into simple classes and aggregate classes, which are further distinguished in property aggregate and instance aggregate classes. Links are partitioned into atomic (also called attributes), basic, and complex.

Let a nonempty finite alphabet $\mathcal{A}$ of atomic classes (classes denoted simply by a name, no matter if simple or aggregate), and a nonempty finite alphabet $\mathcal{U}$ of attributes be available. We use A for a generic element of $\mathcal{A}$, U (possibly with subscript) for a generic element of $\mathcal{U}$, C (possibly with subscript) for a generic class, b (possibly with subscript) for a generic basic link, and L (possibly with subscript) for a generic complex link. The language of $\mathcal{CATS}$ has the following syntax ($n, k \geq 1$):

$$
\begin{array}{lcl}
C & ::= & A \mid \tau(U_1, \ldots, U_n) \mid \chi(C, U_1, \ldots, U_n) \mid \sigma(C) \mid \\
 & & C_1 \sqcap C_2 \mid \neg C \mid \forall L.C \mid (\leq k\, b.C) \mid (\leq k\, b^-.C) \mid \\
 & & (b_1 \subseteq b_2) \mid (b_1^- \subseteq b_2^-) \\
b & ::= & U \mid \ni \mid b_1 \cup b_2 \mid b_1 \setminus b_2 \\
L & ::= & b \mid L_1 \circ L_2 \mid L_1 \cup L_2 \mid L^* \mid L^- \mid id(C)
\end{array}
$$

We use a (possibly with subscript) for b and b^-, and we adopt the following abbreviations: $\top \doteq A \sqcup \neg A$, $\bot \doteq \neg\top$, $\tau \doteq \tau(U_1) \sqcup \cdots \sqcup \tau(U_m)$ (where $\{U_1, \ldots, U_m\} = \mathcal{U}$), $\sigma \doteq \sigma(\top)$, $C_1 \sqcup C_2 \doteq \neg(\neg C_1 \sqcap \neg C_2)$, $\exists L.C \doteq \neg\forall L.\neg C$, $\emptyset \doteq \ni \setminus \ni$, $(\geq k\, a.C) \doteq \neg(\leq k+1\, a.C)$, $a_1 \cap a_2 \doteq a_1 \setminus (a_1 \setminus a_2)$, and $(a_1 = a_2) \doteq (a_1 \subseteq a_2) \sqcap (a_2 \subseteq a_1)$. Parentheses are used to disambiguate expressions.

The semantics for the language of $\mathcal{CATS}$ is based on an *interpretation* $\mathcal{I} = (\mathcal{O}^\mathcal{I}, \cdot^\mathcal{I})$, where $\mathcal{O}^\mathcal{I}$ is the *universe* of the interpretation, and $\cdot^\mathcal{I}$ is the *interpretation function* over such a universe. Differently from the usual notion of interpretation, $\mathcal{O}^\mathcal{I}$ is a nonempty set of *polymorphic objects*, which means that every object in $\mathcal{O}^\mathcal{I}$ has none, one, or both of the following two forms:

1. The form of *tuple*: when an object has this form, it can be considered as a property aggregation, which is formally defined as a partial function from $\mathcal{U}$ to $\mathcal{O}^\mathcal{I}$. We use the term tuple to denote an object in $\mathcal{O}^\mathcal{I}$ that has the form of tuple, and we write $\langle U_1 : o_1, \ldots, U_n : o_n \rangle$[1] to denote any tuple t such that, for each $i \in \{1, \ldots, n\}$, $t(U_i)$ is defined and equal to o_i (which is called the U_i-component of t). Note that the tuple t may have other components as well, besides the U_i-components.
2. The form of *set*: when an object o has this form, it can be considered as an instance aggregate, which is formally defined as a nonempty finite collection of objects in $\mathcal{O}^\mathcal{I}$, with the following proviso: the view of o as a set is unique, in the sense that there is only one finite collection of objects of which o can be considered an aggregation, and no other object o' is the aggregation of the same collection. We use the term set to denote an object in $\mathcal{O}^\mathcal{I}$ that has the form of set, and we write $\{\!|o_1, \ldots, o_n|\!\}$ to denote the collection whose members are exactly $o_1, \ldots, o_n$.

Objects having none of these forms are called elementary objects - i.e., individuals with no structure.

[1] This notation makes it clear that a tuple is indeed a function assigning one element of $\mathcal{O}^\mathcal{I}$ to some of the elements of $\mathcal{U}$.

The interpretation function $\cdot^{\mathcal{I}}$ is defined as follows:

- It assigns to $\ni$ a subset of $\mathcal{O}^{\mathcal{I}} \times \mathcal{O}^{\mathcal{I}}$ such that for each $\{\!|\ldots, o, \ldots|\!\} \in \mathcal{O}^{\mathcal{I}}$, we have that $(\{\!|\ldots, o, \ldots|\!\}, o) \in \ni^{\mathcal{I}}$.
- It assigns to every attribute U a subset of $\mathcal{O}^{\mathcal{I}} \times \mathcal{O}^{\mathcal{I}}$ such that, for each $\langle \ldots, U : o, \ldots \rangle \in \mathcal{O}^{\mathcal{I}}$, $(\langle \ldots, U : o, \ldots \rangle, o) \in U^{\mathcal{I}}$, and there is no $o' \in \mathcal{O}^{\mathcal{I}}$ different from o such that $(\langle \ldots, U : o, \ldots \rangle, o') \in U^{\mathcal{I}}$. Note that this implies that every U in a tuple is functional for the tuple.
- It assigns to every basic link a subset of $\mathcal{O}^{\mathcal{I}} \times \mathcal{O}^{\mathcal{I}}$ such that the following conditions are satisfied:

$$\begin{array}{lcl} (b_1 \cup b_2)^{\mathcal{I}} & = & b_1^{\mathcal{I}} \cup b_2^{\mathcal{I}} \\ (b_1 \setminus b_2)^{\mathcal{I}} & = & b_1^{\mathcal{I}} - b_2^{\mathcal{I}} \\ (b^-)^{\mathcal{I}} & = & \{(o, o') \mid (o', o) \in b^{\mathcal{I}}\}. \end{array}$$

- It assigns to every complex link a subset of $\mathcal{O}^{\mathcal{I}} \times \mathcal{O}^{\mathcal{I}}$ such that the usual conditions for $\circ$, $\cup$, *, $^-$, and id are satisfied:

$$\begin{array}{l} (L_1 \cup L_2)^{\mathcal{I}} = L_1^{\mathcal{I}} \cup L_2^{\mathcal{I}} \\ (L_1 \circ L_2)^{\mathcal{I}} = L_1^{\mathcal{I}} \circ L_2^{\mathcal{I}} \\ (L^*)^{\mathcal{I}} = (L^{\mathcal{I}})^* \\ (L^-)^{\mathcal{I}} = \{(o, o') \in \mathcal{O}^{\mathcal{I}} \times \mathcal{O}^{\mathcal{I}} \mid (o', o) \in R^{\mathcal{I}}\} \\ id(C)^{\mathcal{I}} = \{(o, o) \in \mathcal{O}^{\mathcal{I}} \times \mathcal{O}^{\mathcal{I}} \mid o \in C^{\mathcal{I}}\}. \end{array}$$

- It assigns to every class a subset of $\mathcal{O}^{\mathcal{I}}$ in such a way that the following conditions are satisfied ($\sharp\{\}$ denotes the cardinality of a set): - $A^{\mathcal{I}} \subseteq \mathcal{O}^{\mathcal{I}}$
  - $\tau(U_1, \ldots, U_n)^{\mathcal{I}} = \{\langle U_1 : o_1, \ldots, U_n : o_n \rangle \in \mathcal{O}^{\mathcal{I}} \mid o_1, \ldots, o_n \in \mathcal{O}^{\mathcal{I}}\}$
  - $\chi(C, U_1, \ldots, U_n)^{\mathcal{I}} = S \subseteq \tau(U_1, \ldots, U_n) \cap C^{\mathcal{I}}$ and no distinct $s, s' \in S$ have the same $U_1, \ldots, U_n$-components
  - $\sigma(C)^{\mathcal{I}} = \{\{\!|o_1, \ldots, o_n|\!\} \in \mathcal{O}^{\mathcal{I}} \mid o_1, \ldots, o_n \in C^{\mathcal{I}}\}$
  - $(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
  - $(\neg C)^{\mathcal{I}} = \mathcal{O}^{\mathcal{I}} - C^{\mathcal{I}}$
  - $(\forall L.C)^{\mathcal{I}} = \{o \in \mathcal{O}^{\mathcal{I}} \mid \forall o'.(o, o') \in R^{\mathcal{I}} \supset o' \in L^{\mathcal{I}}\}$
  - $(\leq k\, a.C)^{\mathcal{I}} = \{o \in \mathcal{O}^{\mathcal{I}} \mid \sharp\{(o, o') \in a^{\mathcal{I}} \wedge o' \in C^{\mathcal{I}}\} \leq k\}$
  - $(a_1 \subseteq a_2)^{\mathcal{I}} = \{o \in \mathcal{O}^{\mathcal{I}} \mid \{o' \mid (o, o') \in a_1^{\mathcal{I}}\} \subseteq \{o' \mid (o, o') \in a_2^{\mathcal{I}}\}\}$.

A $\mathcal{CATS}$ TBox $\mathcal{K}$ is a finite set of inclusion assertions of the form $C_1 \sqsubseteq C_2$, where C_1 and C_2 are classes in $\mathcal{CATS}$ (we write $C_1 \equiv C_2$ for $C_1 \sqsubseteq C_2, C_2 \sqsubseteq C_1$). As usual, an interpretation $\mathcal{I}$ is a model of $C_1 \sqsubseteq C_2$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, and $\mathcal{K} \models C_1 \sqsubseteq C_2$ (read as $\mathcal{K}$ logically implies $C_1 \sqsubseteq C_2$), if each model of all assertions in $\mathcal{K}$ is also a model of $C_1 \sqsubseteq C_2$. As mentioned, we have the following result.

Theorem 1 *Logical implication in $\mathcal{CATS}$ is EXPTIME-complete.*

3 Discussion

Let us discuss the most important modeling capabilities of $\mathcal{CATS}$ by means of one example.

$\top \sqsubseteq (\texttt{father} \cap \texttt{mother} \subseteq \emptyset) \sqcap$
$\quad (\texttt{father} \cap \texttt{children} \subseteq \emptyset) \sqcap$
$\quad (\texttt{children} \cap \texttt{mother} \subseteq \emptyset)$
$\top \sqsubseteq \forall \texttt{father}^- \cup \texttt{mother}^- \cup \texttt{children}^-.\texttt{Family}$
$\texttt{Date} \equiv \chi(\texttt{Date}, \texttt{day}, \texttt{month}, \texttt{year})$
$\exists \texttt{date}^-.\top \sqsubseteq \texttt{Date}$
$\exists \texttt{day}^-.\top \sqsubseteq \texttt{Day}$
$\exists \texttt{month}^-.\top \sqsubseteq \texttt{Month}$
$\exists \texttt{year}^-.\top \sqsubseteq \texttt{Year}$
$\exists \texttt{city}^-.\top \sqsubseteq \texttt{City}$
$\texttt{Day} \sqcup \texttt{Month} \sqcup \texttt{Year} \sqsubseteq \neg\tau \sqcap \neg\sigma$
$\texttt{Mayor} \equiv \exists \texttt{mayor}^-.\top$
$\exists \texttt{mayor}.\top \sqsubseteq \texttt{City}$
$\texttt{City} \sqsubseteq \tau(\texttt{name}, \texttt{state}, \texttt{country}, \texttt{mayor}) \sqcap$
$\quad \chi(\texttt{City}, \texttt{name}, \texttt{state}, \texttt{country}) \sqcap \chi(\texttt{City}, \texttt{mayor})$
$\texttt{Family} \sqsubseteq \sigma(\texttt{Person}) \sqcap \tau(\texttt{father}, \texttt{mother}, \texttt{date}, \texttt{city}) \sqcap$
$\quad \chi(\texttt{Family}, \texttt{father}, \texttt{mother}, \texttt{date}) \sqcap$
$\quad (\ni = \texttt{father} \cup \texttt{mother} \cup \texttt{children})$
$\texttt{StillFamily} \sqsubseteq \texttt{Family} \sqcap \chi(\texttt{StillFamily}, \texttt{father}, \texttt{mother})$
$\texttt{PhdFamily} \equiv (\geq 3\ \ni .\texttt{PhdPerson}) \sqcap (\leq 1\ \ni .\neg\texttt{PhdPerson})$
$\texttt{Person} \sqsubseteq (\exists \texttt{children}^-.\top) \sqcap (\leq 1\, \texttt{children}^-.\top)$
$\texttt{ChildOfMayor} \equiv \exists \texttt{children}^- \circ \texttt{father}.\texttt{Mayor}$
$\texttt{VeryPhd} \equiv \forall(\texttt{children}^- \circ (\texttt{father} \cup \texttt{mother}))^*.\texttt{PhdPerson}$

Figure 1: Families, persons, and cities

Figure 1 shows a TBox $\mathcal{K}$ modeling a world with persons, families and cities. The following observations help understanding the expressive power of $\mathcal{CATS}$.

- Objects are polymorphic. For example, every instance of `Family` (representing families resulting from a marriage) can be seen both as a set of persons, and as a tuple with attributes `father`, `mother`, `date` (of marriage) and `city` (of marriage). Note, however, that assertions can be used to impose that the instances of a certain class (`Day`, `Month` and `Year` in our example) can only be seen as elementary objects.
- Inclusion assertions on classes are used with no limitation. In particular, they can be stated for all kinds of classes, and cycles are allowed in the TBox. Notably, inclusion assertions can also be stated for basic links: indeed, $\top \sqsubseteq (b_1 \subseteq b_2)$ forces b_1 to a subset of b_2 in every model of $\mathcal{K}$. Inclusion assertions of this kind are used in the example to specify the properties of the attributes `father`, `mother` and `children`.
- N-ary relations are supported. Any instance of `Family` can indeed be considered as a relation with four arguments. The χ constructor is used to define keys for (N-ary) relations: for example, the fact that every instance of `Family` is an instance of $\chi(\texttt{Family}, \texttt{father}, \texttt{mother}, \texttt{date})$ implies that the three attributes form a key for the class. On the other hand, `StillFamily`, representing families whose father and mother are still married, has a more specialized key, constituted by the attributes `father` and `mother`. Observe that several keys can

be defined for a class (see `City`).

- Qualified number restrictions and role value maps on basic links can be used without any limitation. Indeed, $(\ni = \texttt{father} \cup \texttt{mother} \cup \texttt{children})$ is a role value map on basic links.
- Complex links can be used for modeling interesting relationships. For example, the relationship `hasfather` between a person and her/his father is captured in $\mathcal{K}$ by $\texttt{children}^- \circ \texttt{father}$ (similarly for `hasmother`). Also, `ancestor` is captured by $(\texttt{hasfather} \cup \texttt{hasmother}) \circ (\texttt{hasfather} \cup \texttt{hasmother})^*$ (see the definition of `VeryPhd`).

As an example of inference that can be draw from $\mathcal{K}$, observe that:

$$\mathcal{K} \models \exists \texttt{children}^-.\top \sqsubseteq \texttt{Person} \sqcap \exists \ni^- .\exists \texttt{father}.\top.$$

Indeed, note that every instance of $\exists \texttt{children}^-.\top$ is also an instance of $\exists \texttt{father}^- \cup \texttt{mother}^- \cup \texttt{children}^-.\top$ and therefore is an instance of `Family`. This means that $\mathcal{K} \models \exists \texttt{children}^-.\top \sqsubseteq \exists \texttt{children}^-.\texttt{Family}$. Observe that $\mathcal{K} \models \texttt{Family} \sqsubseteq (\texttt{children} \subseteq \ni)$, and, since $\mathcal{K} \models \texttt{Family} \sqsubseteq \forall \ni .\texttt{Person}$ (because $\mathcal{K} \models \texttt{Family} \sqsubseteq \sigma(\texttt{Person})$), we have that $\mathcal{K} \models \exists \texttt{children}^-.\top \sqsubseteq \exists \texttt{children}^-.(\forall \texttt{children}.\texttt{Person})$, which implies that $\mathcal{K} \models \exists \texttt{children}^-.\top \sqsubseteq \texttt{Person}$. The fact that $\mathcal{K} \models \exists \texttt{children}^-.\top \sqsubseteq \exists \ni^- .\exists \texttt{father}.\top$ easily follows from the fact that every `Family` is a tuple with attribute `father`.

4 Conclusions

It is our opinion that the work described in this paper makes description logics accomplish the necessary leap in order to be well equipped for the new challenging applications they are faced with. Our first investigations show that $\mathcal{CATS}$ can indeed capture and extend most class-based representation formalisms used in different areas as AI, databases, software engineering, etc.. One main issue still remains to be addressed, namely, the possibility of adding to $\mathcal{CATS}$ suitable constructs for expressing finiteness of nested aggregates, and, correspondingly, suitable techniques for reasoning in finite models (in the style of [Calvanese *et al.*,1994]). This will be the subject of further research.

References

[Baader, 1991] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proc. of IJCAI-91*, Sydney, Australia, 1991.

[Bergamaschi and Sartori, 1992] S. Bergamaschi and C. Sartori. On taxonomic reasoning in conceptual design. *ACM Trans. on Database Systems*, 17(3):385–422, 1992.

[Borgida, 1992] A. Borgida. From type systems to knowledge representation: Natural semantics specifications for description logics. *J. of Intelligent and Cooperative Information Systems*, 1(1):93–126, 1992.

[Brachman, 1977] R. J. Brachman. What's in a concept: Structural foundations for semantic networks. *International Journal of Man-Machine Studies*, 9(2):127–152, 1977.

[Brodie and Ridjanovic, 1984] M. L. Brodie and D. Ridjanovic. On the design and specification of database transactions. In *On Conceptual Modelling*, pages 277–306. Springer-Verlag, 1984.

[Buchheit *et al.*, 1993] M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *J. of Artificial Intelligence Research*, 1:109–138, 1993.

[Calvanese *et al.*, 1994] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class based representation formalisms. In *Proc. of KR-94*, pages 109–120, Bonn, 1994. Morgan Kaufmann, Los Altos.

[Catarci and Lenzerini, 1993] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *J. of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.

[De Giacomo and Lenzerini, 1994] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of AAAI-94*, pages 205–212. AAAI Press/The MIT Press, 1994.

[Nebel, 1991] B. Nebel. Terminological cycles: Semantics and computational properties. In *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, Los Altos, 1991.

[Piza *et al.*, 1992] B. Piza, K.-D. Schewe, and J. W. Schmidt. Term subsumption with type constructors. In *Proc. of CIKM-92*, pages 449–456, Baltimore, 1992.

[Schild, 1991] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI-91*, pages 466–471, Sydney, 1991.

[Schmolze, 1989] J. G. Schmolze. Terminological knowledge representation systems supporting n-ary terms. In *Proc. of KR-89*, pages 432–443. Morgan Kaufmann, Los Altos, 1989.

[Schreiber *et al.*, 1993] G. Schreiber, B. Wielinga, and J. Breuker. *KADS: A principled approach to knowledge-based system development.* Academic Press, 1993.

[Smith and Smith, 1977] J. M. Smith and D. C. P. Smith. Database abstractions: Aggregation and generalization. *ACM Transactions on Database Systems*, 2(2):105–133, 1977.

[Woods and Schmolze, 1992] W. A. Woods and J. G. Schmolze. The KL-ONE family. In *Semantic Networks in Artificial Intelligence*, pages 133–178. Pergamon Press, 1992.

[Woods, 1975] W. A. Woods. What's in a link: Foundations for semantic networks. In *Representation and Understanding: Studies in Cognitive Science*, pages 35–82. Academic Press, 1975.