Ontology Engineering beyond the Modeling of Concepts and Relations

Steffen Staab and Alexander Maedche[1]

Abstract. This papers presents an approach for modeling large-scale ontologies. We extend well-established methods for modeling concepts and relations by transportable methods for modeling ontological axioms. The gist of our approach lies in the way we treat the majority of axioms. They are categorized into different types and specified as complex objects that refer to concepts and relations. Considering language and system particularities, this first layer of representation is then translated into the target representation language. This two-layer approach benefits engineering, because the intended meaning of axioms is captured by the categorization of axioms. Classified object representations allow for versatile access to and manipulations of axioms via graphical user interfaces.

1 Introduction

Ontologies have shown their usefulness in application areas such as intelligent information integration, information brokering, or knowledge-based systems, to name but a few. The role of ontologies is to capture domain knowledge and provide a commonly agreed upon understanding of a domain. The common vocabulary of an ontology, defining the meaning of terms and their relations, is usually organized in a taxonomy and contains modeling primitives such as concepts, relations, and axioms. A couple of representation mechanisms and ontology engineering environments have been developed that allow for the representation and engineering of ontologies. In fact, these languages and tools have matured considerably over the last few years.

Nevertheless, while support for modeling of concepts and relations has been extensively provided through convenient graphical user interfaces, the same cannot be said about the modeling of axioms. Often axiom specification in ontology modeling environments is restricted to what subsumption offers in a description logics framework (McGuinness & Patel-Schneider, 1998) or to what the ontology engineer encodes in some kind of first-order logic language (Blázquez *et al.*, 1998), or axiom modeling is neglected at all (e.g. (Grosso *et al.*, 1999)). This situation is detrimental to the modeling of large-scale ontologies, because it aggravates engineering and maintainance of large sets of axioms.

Another drawback, along similar lines, arises from the fact that the ontology engineer obliges to a particular symbol representation of axioms too early in the development process. If need arises to switch from one representation language to another one, many ontology engineering efforts are lost forever. Though there are a few approaches that translate between representation languages (e.g., OKBC (Chaudhri *et al.*, 1998), or ODE (Blázquez *et al.*, 1998)), these approaches typically fail to produce the desired results. The reason is that a language like first-order predicate logic allows for many syntactic variations to denote the same semantic meaning of an axiom and the translation from first-order logic into a target representation then easily fails to be consistent over a range of syntactic variations — if it provides a semantic at all. This, however, is a major pitfall, since the semantics of ontology definitions is mostly void without the specification of axioms.

With our ontology engineering approach we pursue the modeling of ontologies such that graphical means exploited for modeling of concepts and relations scale up to axiom specifications. The core idea is to use a categorization and an object representation of axioms that organize axioms and that provide a compact, intuitively accessible representation. Additionally, our approach facilitates translations of many axioms specifications into various target languages, because categorization of axioms is centered around their *semantic meaning* rather than their syntactic representation, yielding a better clue at how to adapt to a particular representation — or even to a specific application with a proprietary inference engine. Thus, our approach continues a line that has been developed from (McCarthy & Roth, 1969) over (Brachman, 1979) to Gruber (Gruber, 1993). We continue this development, but improve actual engineering practice as we considerably extend the possibilities that Gruber categorized in his Frame Ontology and as we directly use these categorizations as a basis for graphical engineering and editing.

In the following, we briefly survey existing tools and methods for modeling ontologies, which also served as a starting point for our own work. Then we describe the core idea of our approach and illustrate with several complex examples, which extend the scope of previous works and which we conceive of as particularly interesting to ontology engineering, of how to realize our approach.

2 Foundations and Related Work

Our approach described in this paper is based on well-agreed upon methods for modeling concepts and relations. Common to all ontology engineering environments we know of is an object-oriented model that may be browsed and extended by corresponding views onto concepts and relations (cf., Benjamins *et al.* (1999) for an up-to-date survey of systems). Typically, single/multiple taxonomies of concepts, *i.e.* concepts that inherit properties from single/multiple parents, provide the graphical and methodological backbone to all these approaches. Treatment of relations varies to the extent to which relations are considered as first-order entities. Like concepts, relations usually come with several properties of their own, *e.g.* names and documentation. Concepts are linked by relations to other concepts or to built-in types, *i.e.* simple, system-defined, concepts.

As foundation for our work we have decided to offer views

[1] {staab, maedche}@aifb.uni-karlsruhe.de, Institute AIFB, Karlsruhe University, Germany, http://www.aifb.uni-karlsruhe.de/WBS, Tel.: +49-721-608 {4751,6558}, Fax: +49-721-693717

ECAI 2000. *14th European Conference on Artificial Intelligence*
Workshop on Applications of Ontologies and Problem-Solving Methods
R.V. Benjamins, A. Gomez-Perez, N. Guarino, M. Uschold (eds.)

onto concepts and relations similar to the ones available in Protégé (Grosso *et al.*, 1999). While one may conceive of some minor extensions of this approach, e.g., to account for finer distinctions like the ones between SUBCONCEPTOF and PROPERSUBCONCEPTOF ("$\sqsubset$" vs. "$\sqsubseteq$" in description logics), our overall approach for modeling concepts and relations is consistent with all of the above mentioned systems and representation languages (and several more). Figure 1 depicts our ontology engineering environment OntoEdit.

Now, whereas the modeling of concepts and relations is well-agreed upon, quite the contrary holds for the modeling of axioms. The majority of ontology modeling environments simply ignores axioms and, hence, delegates their modeling to a separate *encoding phase* in that axioms are hand-coded with an ASCII editor (e.g., (Swartout *et al.*, 1996; Grosso *et al.*, 1999)). While this is readily feasible when only few axioms are necessary, modeling becomes extremely difficult with the proliferation of axioms.

This problem has been recognized and partially addressed, *viz.* through Ontolingua (Fikes, Farquhar, & Rice, 1997) and in an even more sophisticated manner by ODE (Blázquez *et al.*, 1998). ODE aims at the *knowledge level* of ontology modeling rather than at the *symbol level* and, hence, considers axioms as entities that also need to be treated by graphical views supporting documentation and names for axioms. However, Blázquez *et al.* (1998) still require the formulation of axioms in particular target languages, e.g. first-order predicate logic. Like OKBC (Chaudhri *et al.*, 1998) and Ontolingua (Gruber, 1993) they try to translate axiom specifications between different representation languages in order to overcome the barrier an ontology engineer encounters when he has to move from one representation language to the other. In practice, translation of axiom specifications often fails to deliver the desired results, because literal translations usually do not work, and, hence, what is needed is the recognition of the *meaning* of axioms and its translation.[2] Since the recognition of axiom meanings would entail the proper comparison of axiom models, this task however is undecidable. Note that this problem also persists with ontologies specified in description logics. In this case, axioms are often brought to bear through particular modeling strategies that exploit the description logics subsumption mechanism (e.g., Hahn, Schulz, & Romacker (1999)) or through production rules that enforce axioms in the A-Box. In addition to translation problems, such strategies require considerable modeling discipline from the ontology engineer.

We have decided to borrow the best from the ODE approach and also to treat axioms as first-order entities. In order to counter problems with language independence, structuring of axiom specifications, and modeling strategies, we have conceived an approach for axiom specifications that builds on an "*axioms are objects*" view elaborated in the following.

[2] In the worst case, the effects of n axioms ($n > 1$) in one language is equivalent to the effects of m axioms ($m > 1, m \neq n$) in another language.

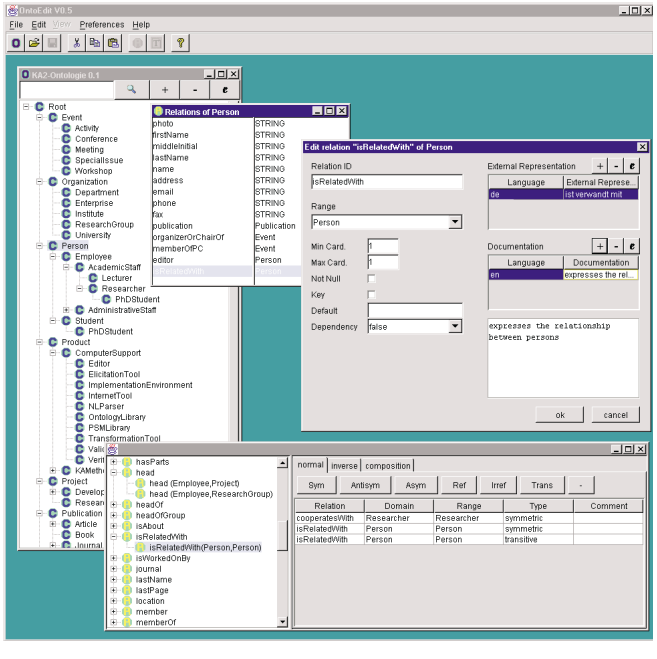

Figure 1. OntoEdit

3 Modeling of Axioms

The motivation of our approach is the specification of axioms such that they remain easily representable and manipulable in an ontology engineering environment. The specification needs to be language independent (to whatever extent this is possible at all), and it must attribute axioms with properties suited for organization.

3.1 Axioms are Objects, too

Representation of axioms for varying target languages turns out to be difficult for all interesting representation languages — and very much the same holds for axiom representation and organization on a screen. The reason is that typically some kind of non-propositional logic is involved that deals with quantifiers and quantifier scope. Axioms are difficult to grasp, since the representation of quantifier scope and its likes is usually what the nitty-gritty details of a particular syntax are about. However, a closer look at the bread and butter issues of ontology modeling reveals that many axioms that need to be formulated aim at much simpler purposes than arbitrary logic structures. Indeed, we have found that many axioms in our applications belong to one of a list of major axiom categories:

1. Axioms for a relational algebra
   (a) Reflexivity of relations
   (b) Irreflexivity of relations
   (c) Symmetry of relations
   (d) Asymmetry of relations
   (e) Antisymmetry of relations
   (f) Transitivity of relations
   (g) Inverse relations
2. Composition of relations[3]
3. (Exhaustive) Partitions[4]
4. Axioms for subrelation relationships
5. Axioms for part-whole reasoning
6. Nonmonotonicity
7. Axioms for temporal and modal contexts

Our experience in several ontology engineering projects has been that often many, though not all, axioms deal with structures that

[3] E.g., GRANDFATHEROF is composed by the relations FATHEROF and PARENTOF.

[4] E.g., concepts *Mammal* and *Fish* share no instances.

appear over and over again — though they must often be realized differently in various representation languages. In fact, axiom specifications even turned out to vary for different inference machines working on the basically same representation language. Hence, our approach distinguishes the structures that are repeatedly found in axiom specifications from the corresponding description in a particular language. We describe axioms as complex objects that refer to concepts and relations. A translation step then realizes axioms on a second layer in a particular target representation language.

This two-layer approach directly supports the objectives that we argued for in the beginning of this section. The first layer allows us to abstract from many representation particularities and the categorization let us get a better grasp at the meaning of an axiom (or a set of axioms). It makes axioms more readily available for GUI interaction, and, hence, directly supports the engineering of axioms at the *knowledge level* rather than at the symbol level of a particular language. The second layer may appear to be overhead only at the first glimpse, however it adds a degree of flexibility and modularity that greatly profits the ontology engineering task. Flexibility comes from the particular translation strategy chosen for particular types of axioms. As different inference mechanisms differ even for identical representation languages the second layer may be tuned to optimize overall performance of ontology (or knowledge base) services without even touching on the first layer. For instance, transitive inferences may be optimized for incremental updates in one application. In addition, axioms may be formulated without considering ubiquitous parameters, like ones for time or modal context, that might otherwise jam axiom specifications that are not affected by time or context at all (cf. subsections on nonmonotonicity and contexts).

In order to elucidate our approach, we proceed through a few intriguing examples of our categorizations of axiom specifications listed above. We give examples that illustrate the benefits of our approach by translation to two prominent ontology modeling mechanisms, viz. a frame representation (F(rame)-Logic (Kifer, Lausen, & Wu, 1995; Decker, 1998)) and a description logics representation (LOOM (Woods & Schmolze, 1992; MacGregor, 1994)). Exploiting the expressiveness of F-Logic, we specify translation axioms that work directly on the object representation of axioms when possible, thus describing a formally concise and executable translation. For syntactically trickier translation steps and for description logics we simply indicate the results of translation in the second layer.

The reader should note that we do neither consider these two languages to comprise all possible ontology specification languages nor think that the axiom types we mention exhaust all relevant types. Rather we believe that experiences in particular domains will push for further extensions. Hence, our main purpose is to acquaint the reader with our *principle methodology* that is transportable to other target languages and other axiom types, when need arises.

3.2 Axioms for a relational algebra

The axiom types that we have shown above are listed such that easier axioms come first and harder ones appear further down in the list. Axiom specifications that are referred to as "axioms for a relational algebra" rank among the simplest ones. They describe axioms with rather local effects, because their implications only affect one or two relations. We here show one simple example of these in order to explain the basic approach and some syntax. The principle approach easily transfers to all axiom types from 1.(a)-(g) to 3.

Let us consider an example for symmetry. A common denotation for the symmetry of a relation ISRELATED (such as used for "Homer Simpson is related to Bart Simpson") in first-order predicate logic boils down to:

(1) $\forall X, Y \quad \text{ISRELATED}(X, Y) \leftarrow \text{ISRELATED}(Y, X).$

In F-Logic, this would be a valid axiom specification, too. Most often, however, modelers that use F-Logic take advantage of the object-oriented syntax. A concept definition in F-Logic for *Person* being a *LivingBeing* with attribute ISRELATED is given in (2), while a fact that Homer is a *Person* who ISRELATED to Bart appears like in (3).

(2) *Person*::*LivingBeing*[ISRELATED $\Rightarrow\!\!>$ *Person*].

(3) Homer:*Person*[ISRELATED $\twoheadrightarrow$ Bart].

Hence, a rule corresponding to (1) is given by (4).

(4) $\forall X, Y \quad Y[\text{ISRELATED} \twoheadrightarrow X] \leftarrow X[\text{ISRELATED} \twoheadrightarrow Y].$

In contrast, a description logics language like LOOM[5] provides a modeling primitive for specifying symmetry:

(5) `(defrelation` ISRELATED `:characteristics (:symmetric))`

In our approach, we denote symmetry as a predicate that holds for particular relations:

(6) SYMMETRIC(ISRELATED).

For a particular language like F-Logic, one may then derive the implications of symmetry by a general rule and, thus, ground the meaning of the predicate SYMMETRIC in a particular target language. The corresponding transformation rule (here in F-Logic) states that if for all symmetric relations R and object instances X and Y it holds that X is related to Y via R, then Y is also related to X via R.

(7) $\forall R, X, Y \quad Y[R \twoheadrightarrow X] \leftarrow \text{SYMMETRIC}(R)$ and $X[R \twoheadrightarrow Y]$.

This small example already shows three advantages. First, the axiom specification (6) is rather language independent. Second, our approach for denoting symmetry is much sparser than its language specific counterparts. And, third, symmetry now constitutes a class of its own and one may easily give a GUI view that lists all relations that are symmetric or that are not. These steps are summarized in Table 1 (with the exception of PL1).

1	SYMMETRIC(ISRELATED)
2	$\forall X, Y \quad X[\text{ISRELATED} \twoheadrightarrow Y] \leftarrow Y[\text{ISRELATED} \twoheadrightarrow X].$
3	`(defrelation` ISRELATED `:characteristics (:symmetric))`
4	$\forall R, X, Y \quad Y[R \twoheadrightarrow X] \leftarrow \text{SYMMETRIC}(R)$ and $X[R \twoheadrightarrow Y]$.

Table 1. Symmetry

For easier understanding, we will reuse this table layout for subsequent examples.

[5] There are other description logics languages like FaCT that provide more comprehensive support for terminological reasoning, however, they often come with their own problems, e.g. FaCT offers no A-Box reasoning (Horrocks, 1998).

3.3 Axioms for subrelation relationships

A major requirement for ontologies is the ability to structure relations into hierarchies. Natural language applications (but also other areas like medical domains) rely on very specific relations that denote background knowledge, like a *Hotel* HASDOUBLEROOM *DoubleRoom*. In order to bridge from a conceptually high-level linguistic expression like "has", which closely resembles the general HASPART relation, to the more specific HASDOUBLEROOM, via other relations like HASROOM and HASPHYSICALPART, information about the relation hierarchy must be retrievable from the ontology modeling framework (cf. (Romacker, Markert, & Hahn, 1999)). An object representation of corresponding axioms, which closely resembles its description logics counterpart, provides the structural information about relations, allows for a corresponding visualization (cf. relation hierarchy in Figure 1), and may be easily translated by a general axiom into a target representation language (cf. Table 2).

1	SUBRELATIONOF(HASDOUBLEROOM,HASROOM)
2	$\forall X, Y \quad X[\text{HASROOM} \twoheadrightarrow Y] \leftarrow$ $X[\text{HASDOUBLEROOM} \twoheadrightarrow Y]$.
3	(`defrelation` HASDOUBLEROOM `:is-primitive` HASROOM)
4	$\forall R, Q, X, Y \quad X[Q \twoheadrightarrow Y] \leftarrow$ SUBRELATIONOF(R, Q) and $X[R \twoheadrightarrow Y]$.

Table 2. Subrelation relationship

3.4 Axioms for part-whole reasoning

Two aspects of reasoning on part-whole relations have received special attention in the ontology engineering community. The first issue that has been debated is whether transitivity should be considered a general property of partonomies. We here only refer to recent work of Hahn, Schulz, & Romacker (1999) and Lambrix & Padgham (2000) who survey this issue. Our approach provides the modeler with the flexibility to turn transitivity constraints of part-whole (sub-)relations on or off as she prefers (cf. the paragraph on axioms for a relational algebra).

The second issue in the debate is about partonomic reasoning. Rector *et al.* (1997) distinguish between role propagation and concept specialization. As for the first, properties of parts may sometimes be propagated to become properties of their wholes. Assume that *ColorOfCarBody* is defined as the COLOROF the *CarBody* and the *CarBody* is defined as a PHYSICALPARTOF the *Car*. While the color of the car body is also the color of the car, the same does not hold for the color of the seats, in spite of the fact that there is no structural difference to be found in these two examples. Similarly, when the engine does not work, the car does not work either, but when the car radio is broken, the car may still run. Hence, it is necessary to specify axioms that propagate particular roles, i.e. properties, from parts to wholes. This must be possible even over several PARTOF relations, precisely speaking over the transitive closure of immediate subrelations of PARTOF, e.g. PHYSICALPARTOF. In our approach, we encode the transitive closures analogously to (6) and state that a particular role may be propagated in an object representation that is straightforward and comparatively easy to understand:

(8) PARTONOMICROLEPROPAGATION(*ColorOfCarBody*, COLOROF, *CarBody*).

In a typical application there will be dozens of definitions like this. It is easy to imagine that their representation-specific counterparts are much more cumbersome to understand. In fact, predicate logic denotation needs to specify at least two axioms for each role that needs to be propagated, *viz.* one at the conceptual level and one at the instance level (cf. Table 3). Description logics provides no direct support[6], but the corresponding implications may be achieved by a rather complicated encoding via additional concept nodes (cf. (Hahn, Schulz, & Romacker, 1999)) or via user-coded functions (Lambrix & Padgham, 2000).

1	PARTONOMICROLEPROPAGATION(*ColorOfCarBody*, COLOROF,*CarBody*)
2	Instance level: $\forall X, Y, Z, S, R \quad X[R \twoheadrightarrow Z] \leftarrow$ $X[R \twoheadrightarrow Y]$ and SUBRELATIONOF(S,PART-OF) and $Y[S \twoheadrightarrow Z]$. Concept level: $\forall X, Y, Z, S, R \quad X[R \Rightarrow\!\!\!\!\!\twoheadrightarrow Z] \leftarrow$ $X[R \Rightarrow\!\!\!\!\!\twoheadrightarrow Y]$ and SUBRELATIONOF(S,PART-OF) and $Y[S \Rightarrow\!\!\!\!\!\twoheadrightarrow Z]$.
3	Concept triples encode partonomic reasoning into taxonomic reasoning or program code performs propagation.
4	Instance level: $\forall C, D, X, Y, Z, S, R \quad X[R \twoheadrightarrow Z] \leftarrow$ PARTONOMICROLEPROPAGATION(C, R, D) and $X : C$ and $Y : D$ and $X[R \twoheadrightarrow Y]$ and SUBRELATIONOF(S,PARTOF) and $Y[S \twoheadrightarrow Z]$. Concept level: analogous to the instance level

Table 3. Part-Whole Reasoning – (i) role propagation

The second aspect of partonomic reasoning that is reflected upon by Rector *et al.* (1997) is concept specification through partonomic reasoning. To cut a long discussion short, we only mention here that this type of axiom specification also relies on the PARTONOMICROLEPROPAGATION specified in Table 3 and show a corresponding F-Logic translation rule in Table 4.

2	Concept level: $\forall X, Y, Z, S, R, W \quad X :: W \leftarrow$ $X[R \Rightarrow\!\!\!\!\!\twoheadrightarrow Y]$ and SUBRELATIONOF(S,PARTOF) and $Y[S \Rightarrow\!\!\!\!\!\twoheadrightarrow Z]$ and $W[R \Rightarrow\!\!\!\!\!\twoheadrightarrow Z]$.

Table 4. Part-Whole Reasoning – (ii) concept specification

3.5 General axioms

The approach we have described so far is not suited to cover every single axiom specification one may think of. Hence, we still must allow for axioms that are specified in a particular representation language like first-order predicate logic and we must try to translate these axioms, possibly with human help, into their target representation language, e.g., description logics. For such general axiom specifications, we do not gain or loose anything compared to related approaches like ODE (Blázquez *et al.*, 1998) on first sight. On second sight, we have found that there is a "*twilight zone*" of axiom types and properties that may not be reasonably represented as demonstrated for axiom types 1 through 5, but that benefit from the principal methodology of separating engineering from target representation! The following two subsections elaborate on this.

3.6 Nonmonotonicity

Axioms may be organized to interact in a nonmonotonic manner. For instance, consider a rule like (9) and its exception (10).[7]

[6] Hahn, Schulz, & Romacker (1999) showed that the support provided by Rector *et al.* (1997) for GRAIL is insufficient.

[7] We conceive that this particular example should rather be modeled through nonmonotonic inheritance. It has been chosen for its simplicity and paedagogical value rather than for its adequacy. Real world examples may, e.g.,

(9) Bird Rule: $\forall X\ X[\text{CANFLY}\twoheadrightarrow \textit{True}] \leftarrow X : \textit{Bird}$.

(10) Penguin Rule: $\forall X\ X[\text{CANFLY}\twoheadrightarrow \textit{False}] \leftarrow$
$X : \textit{Penguin}$.

A translation and compilation step may use the conceptual background knowledge that the second axiom is an exception of the first in order to produce the logic program that implements the intended set of axioms, (10), (11), and (12). Thereby, a tree view visualizes the background knowledge that the premise of the Penguin Rule leads to a different conclusion and, hence, really denotes an exception of the Bird Rule. It does so in an intuitive manner by grouping the Penguin Rule under the Bird Rule, just like a subrelation is grouped under its superrelation in Figure 1.

(11) Compiled Bird Rule: $\forall X\ X[\text{CANFLY}\twoheadrightarrow \textit{True}] \leftarrow$
$X : \textit{Bird}$ `and Not` $\text{EXCEPTION}(X)$.

(12) Exception Rule: $\forall X\ \text{EXCEPTION}(X) \leftarrow$
$X : \textit{Penguin}$.

Like in previous examples, we may well claim that we allow more intuitive access to the relevant axiomatic propositions. This time, however, we need to break up the internal *syntactic* structures of axioms in order to distinguish between premises and implications both of which may exhibit arbitrary logical structures that may not easily be abstracted to an object representation like axioms types 1. to 5.

The example demonstrates a very simple strategy for nonmonotonic reasoning through means of the closed-world assumption. Our basic principle aims at conceptual abstraction through advanced modeling strategies rather than through omnipotent logical theories, while still leaving the door open for more sophisticated nonmonotonic reasoning. In fact, regarding our F-Logic inference engine (Decker, 1998), the strategy is also efficient — inferencing terminates in polynomial time with a well-founded semantics when no function symbols are used.

3.7 Temporal and modal contexts

Temporal and/or modal contexts are often indispensable for practical applications. However, they often make it difficult to "find the forest for all the trees" when it comes to determining and/or maintaining the meaning of a large set of axioms. Similar to the strategy used by Peterson, Andersen, & Engel (1998) who extracted database schemata from CYC, we abstract from axioms that include temporal parameters (like t_b for begin and t_e for end with, e.g., t_b included and t_e excluded from the time interval) towards versions that are easier to understand. I.e. the ontology engineer models an axiom like (13), stating that programs under the GNU licence come for free, and indicates that this axiom belongs to the group of axioms that preserve time boundaries t_b, t_e by ticking off a check box.

(13) $\forall X\ X[\text{COSTS}\twoheadrightarrow 0] \leftarrow$
$X : \textit{Program}[\text{LICENSEMODE}\twoheadrightarrow \text{GNU}]$.

Then, (13) may be compiled into axiom (14), which includes time parameters, for actual inference purposes.

(14) $\forall X\ \text{COSTS}(X, 0, t_b, t_e) \leftarrow$
$X : \textit{Program}$ `and` $\text{LICENSEMODE}(X, \text{GNU}, t_b, t_e)$.

Exactly the same strategy may be exploited for modeling (modal) contexts. One type of axiom that may be produced from (13) by a particular translation strategy is (15) stating that if a person believes that if X has a GNU license mode during a particular time interval then he also believes that X comes for free during that time.

(15) $\forall X, Y, P\ \text{BEL}(P, \text{COSTS}(X, 0, t_b, t_e)) \leftarrow \text{BEL}(P, X :$
$\textit{Program}$ `and` $\text{LICENSEMODE}(X, \text{GNU}, t_b, t_e))$.

Naturally, this example shall only be indicative for the range of possibilities that may be used to denote formal contexts (McCarthy, 1993).[8]

4 Discussion

We have presented an approach towards modeling axioms for ontologies. Our approach is geared towards specifications of axioms that are easily representable and manipulable in our ontology engineering environment OntoEdit. Thereby, we achieve a high degree of language independence and valuable means for axiom organizations. We reach these objectives through a *methodology* that classifies axioms into axiom types according to their *semantic meaning*. Each type receives an object representation that abstracts from particular syntax (as far as possible) and keeps only references to concepts and relations necessary to distinguish one particular axiom of one type from another one of the same type.

The two layers also allow the distinction between *knowledge-level* and *implementation*, thus, opening up numerous possibilities for optimizing performance and, hence, for scaling-up to large ontologies. Even when the limits of object representations are reached, the two-layer approach may be exploited for hard tasks like engineering nonmonotonicity or axioms that work in temporal and modal contexts.

Thus, our approach also extends Gruber's work who tried to categorize KIF expressions into denotations of his Frame Ontology (Gruber, 1993). We continue his development, but improve actual engineering practice, as we directly use these categorizations as a basis for graphical engineering and editing (cf. Figure 1) and as we go beyond traditional predicate logic.

The public release of Upper Cyc©Ontology (Lenat, 1995) also contains a useful set of axiom categorizations. However, this set of categorizations does not go beyond the type of categories we have provided in (1a-g), because axioms are defined using simple inheritance (e.g. the businessPartner relation is a SymmetricBinaryPredicate) without the possiblity of parametrization (such as required for an axiom defining part-whole reasoning).

The examples that we have illustrated are highly relevant for practical ontology engineering tasks that we have experienced in several projects. In fact, our proposal for engineering *part-whole reasoning* axioms outperforms its competitors as far as conceptual adequacy and succinctness of formulation is concerned by a far stretch.

Our approach continues a line that has developed from (McCarthy & Roth, 1969) over (Brachman, 1979) to (Gruber, 1993). The general pursuit aims at the epistomological level for axioms rather than at

be found in (Morgenstern, 1998). Also note that we skip a LOOM formulation as it could not support more general types of nonmonotonic axioms — our specific, very simple, example could be represented with a `:default` slot value attached to the roles CANFLY of the concepts *Bird* and *Penguin*.

[8] A LOOM formulation could, e.g., reify the relations LICENSEMODE and COSTS as concepts in order to account for time intervals and use the `change-kb` construct to distinguish and switch between different modal contexts. For the sake of brevity we do not elaborate this here.

the symbol level. However, most research was focused on providing *one* single, "correct" solution to the problem of engineering terminological systems. We do not restrict our approach to terminological axioms or to a particular type of logic that we would use exclusively for engineering and representation. Rather we have pursued a flexible extensible environment for engineering ontologies, axioms in particular, that should be tailored according to the application. Also, our approach avoids the problem to recognize the appropriate axiom category, which instantaneously arises when a general purpose language is to be translated into another general purpose language. We have given several complex examples that are interesting in themselves and that support our claim.

REFERENCES

Benjamins, R.; Duineveld, A. J.; Stoter, R.; Weiden, M. R.; and Kenepa, B. 1999. Wondertools? A comparative study of ontological engineering tools. In *Proc. of the 12th Int. Workshop on Knowledge Acquisition, Modeling and Mangement (KAW'99), Banff, Canada, October 1999*.

Blázquez, M.; Fernández, M.; García-Pinar, J. M.; and Gómez-Pérez, A. 1998. Building ontologies at the knowledge level using the ontology design environment. In *Proc. of the 11th Int. Workshop on Knowledge Acquisition, Modeling and Mangement (KAW'98), Banff, Canada, October 1998*.

Brachman, R. 1979. On the epistomological status of semantic networks. *Associative Networks* 3–50.

Chaudhri, V. K.; Farquhar, A.; Fikes, R.; Karp, P. D.; and Rice, J. P. 1998. OKBC: A programmatic foundation for knowledge base interoperability. In *Proc. of AAAI-98*, 600–607.

Decker, S. 1998. On domain-specific declarative knowledge representation and database languages. In *Proc. of the 5th Knowledge Representation meets Databases Workshop (KRDB98)*, 9.1–9.7.

Fikes, R.; Farquhar, A.; and Rice, J. 1997. Tools for assembling modular ontologies in Ontolingua. In *Proc. of AAAI 97*, 436–441.

Grosso, E.; Eriksson, H.; Fergerson, R. W.; Tu, S. W.; and Musen, M. M. 1999. Knowledge modeling at the millennium — the design and evolution of Protégé-2000. In *Proc. the 12th International Workshop on Knowledge Acquisition, Modeling and Mangement (KAW'99), Banff, Canada, October 1999*.

Gruber, T. 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition* 5(1):199–220.

Hahn, U.; Schulz, S.; and Romacker, M. 1999. Partonomic reasoning as taxonomic reasoning in medicine. In *Proc. of AAAI-99*, 271–276.

Horrocks, I. 1998. Using an expressive description logic: FaCT or fiction? In *Proc. of KR-98*, 636–647.

Kifer, M.; Lausen, G.; and Wu, J. 1995. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM* 42.

Lambrix, P., and Padgham, L. 2000. Conceptual modeling in a document management environment using part-of reasoning in description logics. *Data & Knowledge Engineering* 32(1):51–86.

Lenat, D. 1995. A large-scale investment in knowledge infrastructure. *Communications of the ACM* (11).

MacGregor, R. 1994. A description classifier for the predicate calculus. In *Proc. of AAAI-94*, 213–220.

McCarthy, J., and Roth, P. 1969. Some philosophical problems from the standpoint of artificial intelligence. *B. Meltzer & D. Michie, Eds. Machine Intelligence*.

McCarthy, J. 1993. Notes on formalizing context. In *Proc. of IJCAI-93*, 555–562.

McGuinness, D., and Patel-Schneider, P. 1998. Usability issues in knowledge representation systems. In *Proc. of AAAI-98*, 608–614.

Morgenstern, L. 1998. Inheritance comes of age: Applying nonmonotonic techniques to problems in industry. *Artificial Intelligence* 103:1–34.

Peterson, B.; Andersen, W.; and Engel, J. 1998. Knowledge bus: Generating application-focused databases from large ontologies. In *Proc. of the 5th Knowledge Representation meets Databases Workshop (KRDB98)*.

Rector, A.; Bechhofer, S.; Goble, C.; Horrocks, I.; Nowlan, W.; and Solomon, W. 1997. The GRAIL concept modelling language for medical terminology. *Journal of AI in Medicine* 9(2):139–171.

Romacker, M.; Markert, M.; and Hahn, U. 1999. Lean semantic interpretation. In *Proc. of IJCAI-99*, 868–875.

Swartout, B.; Patil, R.; Knight, K.; and Russ, T. 1996. Toward distributed use of large-scale ontologies. In *Proc. the 10th International Workshop on Knowledge Acquisition, Modeling and Mangement (KAW'96), Banff, Canada, November 9-14, 1996*.

Woods, W., and Schmolze, J. 1992. The KL-One family. *Computers and Mathematics with Applications* 23(2):133–177.