Knowledge Representation

Language and Conceptualization, WS 2003–2004
Dr. Sabine Bartsch
Darmstadt University of Technology

Aybala Celebi, Jean-Pierre Schwickerath, Jördis Hensen

February 11th, 2004

1 Ontology — `http://www.jfsowa.com/ontology/`

The subject of ontology is the study of the categories of things that exist or may exist in some domain.
The product of such a study, called an **ontology**, is a catalog of the types of things that are assumed to exist in a domain of interest **D** from the perspective of a person who uses a language **L** for the purpose of talking about **D**.
The types in the ontology represent the predicates, word senses, or concept and relation types of the language **L** when used to discuss topics in the domain **D**.
An uninterpreted logic, such as predicate calculus, conceptual graphs, or KIF, is ontologically neutral. It imposes no constraints on the subject matter or the way the subject may be characterized. By itself, logic says nothing about anything, but the combination of logic with an ontology provides a language that can express relationships about the entities in the domain of interest.
An informal ontology may be specified by a catalog of types that are either undefined or defined only by statements in a natural language. A formal ontology is specified by a collection of names for concept and relation types organized in a partial ordering by the type-subtype relation.
Formal ontologies are further distinguished by the way the subtypes are distinguished from their supertypes: an **axiomatized ontology** distinguishes subtypes by axioms and definitions stated in a formal language, such as logic or some computer-oriented notation that can be translated to logic; a **prototype-based ontology** distinguishes subtypes by a comparison with a typical member or prototype for each subtype. Large ontologies often use a mixture of definitional methods: formal axioms and definitions are used for the terms in mathematics, physics, and engineering; and prototypes are used for plants, animals, and common household items.

Erdmann applied the method of formal concept analysis (FCA), developed by Bernhard Ganter and Rudolf Wille (TU-Darmstadt) (1999) and implemented in an automated tool called Toscana.

ALIGNMENT A mapping of concepts and relations between two ontologies A and B that preserves the partial ordering by subtypes in both A and B. If an alignment maps a concept or relation x in ontology A to a concept or relation y in ontology B, then x and y are said to be **equivalent**. The mapping may be partial: there could be many concepts in A or B that have no equivalents in the other ontology. Before two ontologies A and B can be aligned, it may be necessary to introduce new subtypes or supertypes of concepts or relations in either A or B in order to provide suitable targets for alignment. No other changes to the axioms, definitions, proofs, or computations in either A or B are made during the process of alignment. Alignment does not depend on the choice of names in either ontology. For example, an alignment of a Japanese ontology to an English ontology might map the Japanese concept Go to the English concept Five. Meanwhile, the English concept for the verb **go** would not have any association with the Japanese concept Go.

AXIOMATIZED ONTOLOGY A terminological ontology whose concept and relation types are distinguished by axioms and definitions that are stated in logic or in some computer-oriented language that could be automatically translated to logic. There is no restriction on the complexity of the logic that may be used to state the axioms and definitions. The distinction between terminological and axiomatized ontologies is one of degree rather than kind. Axiomatized ontologies tend to be smaller than terminological ontologies, but their axioms and definitions can support more complex inferences and computations. Examples of axiomatized ontologies include formal theories in science and mathematics, the collections of rules and frames in an expert system, and specifications of conceptual schemas in languages like SQL.

DIFFERENTIAE The properties that distinguish a subtype from other types that have a common supertype. The term comes from Aristotle's method of defining new types by stating the **genus** or supertype and stating the properties that distinguish the new type from its supertype. Aristotle's method of definition has become the de facto standard for natural language dictionaries, and it is also widely used for AI knowledge bases and object-oriented programming languages.

KNOWLEDGE BASE An informal term for a collection of information that includes an ontology as one component. Besides an ontology, a knowledge base may contain information specified in a declarative language such as logic or expert-system rules, but it may also include unstructured or unformalized information expressed in natural language or procedural code.

MIXED ONTOLOGY An ontology in which some subtypes are distinguished by axioms and definitions, but other subtypes are distinguished by prototypes. The top levels of a mixed ontology would normally be distinguished by formal definitions, but some of the lower branches, such as plants, animals, and common household objects might be distinguished by prototypes.

PROTOTYPE-BASED ONTOLOGY A terminological ontology whose types and subtypes are distinguished by prototypes rather than definitions and axioms in a formal language. Before a prototype-based ontology can be considered formal, there must be some method for measuring the similarity of any two entities that can be classified according to the types of the ontology. Given such a measure, every type t in the ontology must be assigned a **prototype** or typical instance p. Then an entity x can classified by the following recursive procedure:

- Suppose that x has been classified as an instance of some type t, which has subtypes s sub 1 , ... , s sub n.
- Measure the similarity of x to the prototypes p sub 1 , ... , p sub n for each subtype of t.
- Classify x as an instance of that subtype s sub i whose prototype p sub i is most similar to x by the measure used for the ontology.

For any entity x, this procedure is invoked with x compared to the immediate subtypes of the universal type =BE. After x has been classified as an instance of any type t, the procedure is invoked recursively to classify x further as some subtype of t. The procedure stops when x is classified as an instance of a type whose only proper subtype is the absurd type =CA.

TERMINOLOGICAL ONTOLOGY An ontology whose concepts and relations need not be fully specified by axioms and definitions that determine the necessary and sufficient conditions for their use. The concepts may be partially specified by relations such as subtype-supertype or part-whole, which determine the relative positions of the concepts with respect to one another, but which do not completely define them. Although a terminological ontology may be expressed in logic, the versions of logic required are usually simpler, less expressive, and more easily computable than full first-order predicate calculus.

2 KIF — `http://logic.stanford.edu/kif/dpans.html`

This dpANS specifies the syntax and semantics of Knowledge Interchange Format (KIF) and a syntactic variant of KIF in “infix” form.
Knowledge Interchange Format (KIF) is a language designed for use in the interchange of knowledge among disparate computer systems (created by different programmers, at different times, in different languages, and so forth).
KIF is not intended as a primary language for interaction with human users (though it can be used for this purpose). Different computer systems can interact with their users in whatever forms are most appropriate to their applications (for example Prolog, conceptual graphs, natural language, and so forth).
KIF is also not intended as an internal representation for knowledge within computer systems or within closely related sets of computer systems (though the language can be used for this purpose as well). Typically, when a computer system reads a knowledge base in KIF, it converts the data into its own internal form (specialized pointer structures, arrays, etc.). All computation is done using these internal forms. When the computer system needs to communicate with another computer system, it maps its internal data structures into KIF.
The purpose of KIF is roughly analogous to that of Postscript. Postscript is commonly used by text and graphics formatting systems in communicating information about documents to printers. Although it is not as efficient as a specialized representation for documents and not as perspicuous as a specialized wysiwyg display, Postscript is a programmer-readable representation that facilitates the independent development of formatting programs and printers. While KIF is not as efficient as a specialized representation for knowledge nor as perspicuous as a specialized display (when printed in its list form), it too is a programmer-readable language and thereby facilitates the independent development of knowledge-manipulation programs.
The following categorical features are essential to the design of KIF.

1. The language has declarative semantics. It is possible to understand the meaning of expressions in the language without appeal to an interpreter for manipulating those expressions. In this way, KIF differs from other languages that are based on specific interpreters, such as Emycin and Prolog.

2. The language is logically comprehensive – at its most general, it provides for the expression of arbitrary logical sentences. In this way, it differs from relational database languages (like SQL) and logic programming languages (like Prolog).

3. The language provides for the representation of knowledge about knowledge. This allows the user to make knowledge representation decisions explicit and permits the user to introduce new knowledge representation constructs without changing the language.

In addition to these essential features, KIF is designed to maximize the following additional features (to the extent possible while preserving the preceding features).

1. Implementability. Although KIF is not intended for use within programs as a representation or communication language, it should be usable for that purpose if so desired.

2. Readability. Although KIF is not intended primarily as a language for interaction with humans, human readability facilitates its use in describing representation language semantics, its use as a publication language for example knowledge bases, its use in assisting humans with knowledge base translation problems, etc.