Ontological Foundations for Experimental Science Knowledge Bases (excerpt)

Natalya Fridman Noy[*]
Carole D. Hafner

2. Principles of ontology design

Ontology is an ancient term used by philosophers to mean "a particular theory about the nature of being or reality" (Woolf 1981). Ontology design starts with creating a conceptualization of the domain: one that specifies the possible objects or entities about which knowledge can be expressed (the fundamental categories), and the relationships that can hold among them. Every knowledge model is committed to some conceptualization, implicitly or explicitly. An explicit specification of this conceptualization is called an ontology (Gruber 1993). In artificial intelligence, an ontology must be formally specified, so that it can be interpreted by a computer program. As discussed in the previous section, the desire to share and re-use knowledge bases has created a need for the knowledge-representation community to adopt common ontological conventions, which in turn has motivated recent interest in the principles of ontology design.

In this section, we briefly review the major elements of the artificial intelligence approach to ontology design[1]: taxonomy, structured concepts, and axioms. We use a simple example, the familiar "blocks world" domain, to illustrate the process of conceptualization and some of the issues that arise in creating a formal ontology. We briefly describe two large-scale general-knowledge ontology projects, CYC and WordNet, which are very different from each other, illustrating the range of work currently being done in the field. A survey and in-depth discussion of the field of ontology design can be found in (Fridman Noy and Hafner 1997).

2.1 Creating an ontology: an example

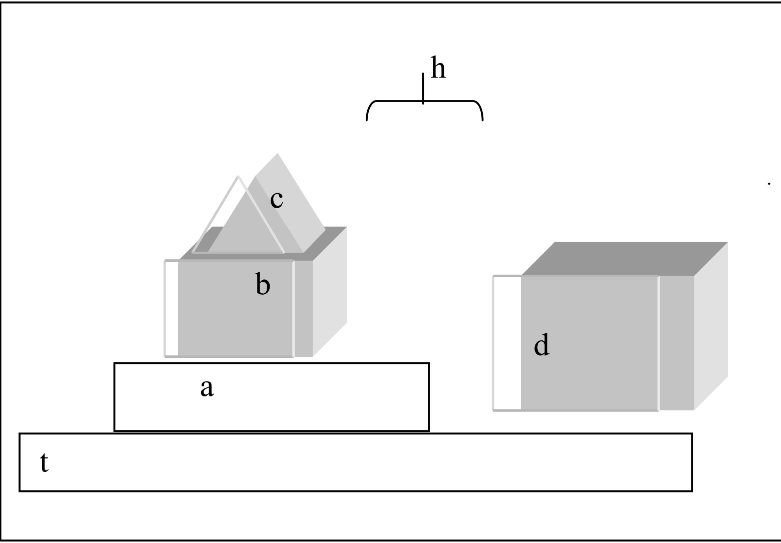
For an example of what an ontology is, consider the simple universe of the blocks world, consisting of a number of differently shaped blocks on a table, which can be stacked up in various ways by a robot hand. In designing our ontology, we may decide that we do not need to represent some concepts such

[*] Currently at: Stanford Medical Informatics, 251 Campus Drive, MSOB X-249, Stanford, CA 94305; e-mail: noy@smi.stanford.edu

[1] The view of ontology presented in this section (and used in the rest of the paper) is accepted by many but not all ontology researchers; we introduce it here to clarify the underlying assumptions of our research.

as the material the blocks are made of, their weight, etc.; in that case, we will not be able to reason about these properties of the blocks. The choice of ontology determines what a system can know and reason about.

To begin conceptualizing the blocks world, we need three categories of objects—`Block`, `Hand`, and `Table`—along with sub-categories of blocks such as `Brick`, `Cube`, and `Pyramid.` We can represent these categories formally as predicates: `Block(x), Table(x), Pyramid(x),`and so on. A simple relation between a block `b` and an object `x` is `Supported(b,x),` which holds if block `b` is directly supported by object `x`. We can define another relation `Above(b,x)` which holds if `b` is any one of a stack of blocks on top of `x` (the Figure 1



Objects:
a, b, c, d, t, h

Relations:
Table (t), Brick (a), Cube (b)
Pyramid (c), Cube (d), Hand (h)
Supported (b, a), Supported (c, b)
Supported (d, t), Supported (a, t)
Above (a, t), Above (b, t),
Above(c, t), Above (d, t),
Above (b, a), Above (c, a),
Above (c, b)

Properties:
Clear (c), Clear (d), Bottom (a),
Bottom (d)

Figure 1. A scenario described by the blocks world ontology.

transitive closure of the `Supported` relation). We can also define other properties of blocks in this world: `Clear(b)` is true if there is nothing resting on block `b`, and `Bottom(b)` is true when block `b` is resting on the table. We can use this ontology to describe formally a particular blocks-world scenario, as shown in Figure 1.

This blocks-world ontology is not yet sufficiently developed to support the intended application: a robot hand that moves blocks around and stacks them up in different configurations. For example, we may need an additional category in the ontology, `Location`, to represent the locations of blocks. We could then introduce a function `Loc(x)` whose value is the location of an object. And we need to represent actions performed by the robot: for example `Pickup(b)` and `Putdown(b)` for picking up and putting down a block, and `Move(b,l)` for moving a block to a location.

2. 2 Elements of ontologies

2.2.1. Concept taxonomy

The creation of an explicit classification hierarchy of concepts is usually the first step in ontology design. For a small domain-specific ontology such as the blocks world, this process may be straightforward; but given the more general goals of ontology design (knowledge sharing and re-use), designing the high-level organization of a large taxonomy presents significant challenges. Virtually all ontologies contain such taxonomic structures as a high-level division of concepts into objects and events. The treatment of other kinds of concepts, including locations, times, and abstract terms such as numbers and ideas, varies from one ontology to another.

Figure 2 illustrates the concept hierarchy of the blocks-world example from the previous section.

Assuming there is an explicit taxonomy of concepts, we need to ask the next question: how is the taxonomy organized? In (Fridman Noy and Hafner 1997) we found three different answers: some ontologies adopt the approach of having everything in a single tree-like concept hierarchy with multiple inheritance. The links in the hierarchy are IS-A links and the division of a concept into subconcepts is disjoint. Other ontologies use a multi-dimensional approach, specifying several parallel dimensions along which one or more high-level categories are sub-categorized. For example, *Real* versus *Abstract*, *Individual* versus *Collective*, and so on. In this case categories are specified by various combinations of values along these dimensions. For instance, HERD can be categorized as being *Real* and *Collective*, whereas IDEA is *Abstract* and *Individual* (Dahlgren 1988). A third major approach to taxonomy organization is having a large number of small local taxonomies that may be linked together via relations or axioms.

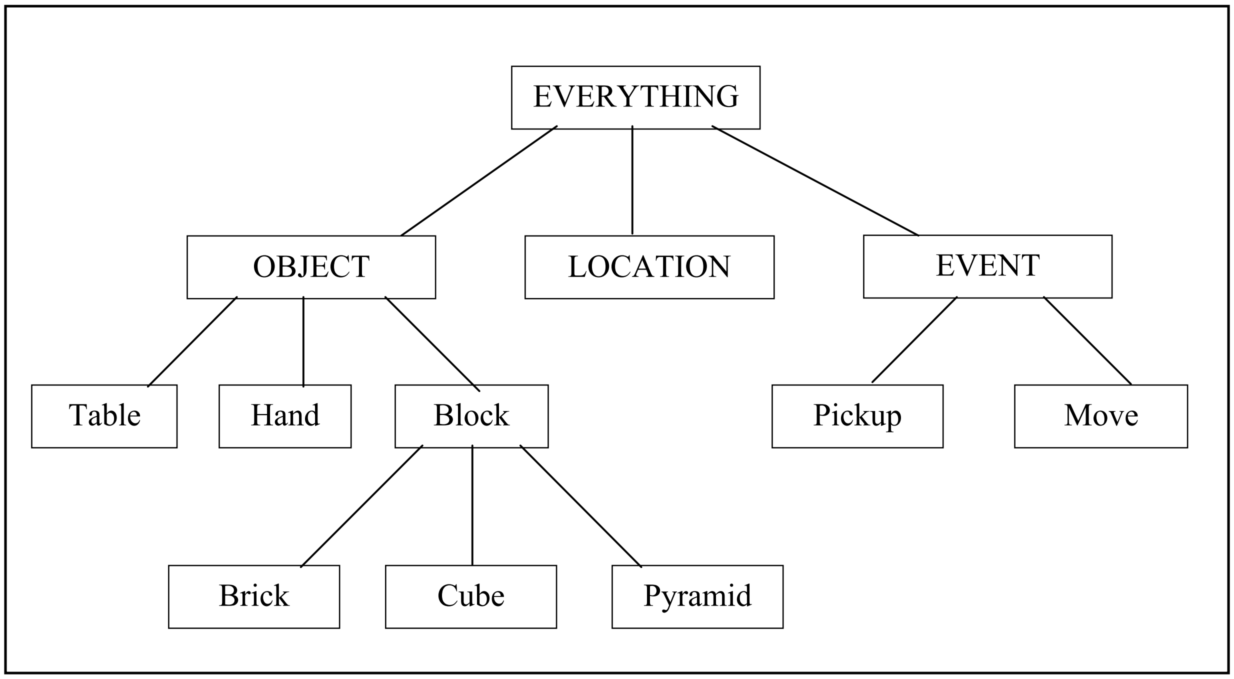


Figure 2. A taxonomy for the blocks world ontology described in Section 2.1.

Incompatibility of high-level taxonomic structure poses an obstacle to integration of different ontologies. A. Campbell and S. Shapiro in (Campbell and Shapiro 1995) discuss the idea of a "mediation interface" that can translate statements made in one ontology to another ontology. They compare top levels of a number of ontologies in order to determine how similar or different they are and, hence how feasible it would be to integrate them. Two of the criteria they use is how tangled and how sparse or dense the top-level hierarchy is. For example, a simple tree-like structure with little or no multiple inheritance, such as the WordNet taxonomy (Figure 5), would not be considered tangled, whereas a system that employs the multi-dimensional approach, such as Cyc (Figure 4) would have a highly tangled taxonomy.

2.2.2 Structured concepts

Defining concepts as objects with internal structure representing prototypical properties, components, and roles introduces additional complexity and power into ontology design. Structured concepts can be formally represented using slot-filler structures known as *frames* (Minsky 1981). For example, we can introduce a property slot called `Color` for blocks, with a finite set of possible fillers: {red, green, blue . . . }. Adding this slot makes the color of a block an inherent part of its definition, rather than an incidental relationship. The same approach can also be used to represent expected parts or components of objects, such as the seeds of an apple. In the blocks-world ontology, objects of type `Location` might be defined as having three numerical components: `X-coordinate`, `Y-coordinate`, and `Z-coordinate`. If we want blocks to have individual names and to refer to them by their names, we would add a `Name` slot to the internal structure definition of the `Block` concept.

In addition to properties and components, some concept definitions (especially those representing events) include slots for essential participants, called "roles". For example, in representing the concept of a `Sale` in an ontology for commerce, we need to define roles for the `Buyer`, the `Seller`, and the `Goods` being sold. In the blocks world, we can define the `Move` action as a structured concept with two roles: the block being moved, and the location which is the destination of the move. (In a more complete blocks-world system capable of reasoning about sequences of actions, it would be necessary to have temporal concepts in the ontology, and an additional slot in the `Move` frame to represent the time when the move occurred.)

When structured concepts are combined with IS-A taxonomy, the *inheritance* rule of inference states that a sub-concept is implicitly defined to possess all of the slots of its parent, plus any new slots defined specifically for it[2]. Figure 3 shows part of the blocks-world taxonomy with internal concept structure added. The sub-categories of `Block` (`cube, pyramid`, etc.) have a `Color` property (by

[2] As in object-oriented programming, knowledge representation systems must decide whether to allow a sub-concept to over-ride inheritance by creating a new slot with the same name as a slot of its parent or how to resolve name conflicts if multiple inheritance is allowed (Fikes et al. 1997).

virtue of the inheritance rule), and in addition, cubes have a `Side-length` property, bricks have a `Length, Width`, and `Height` property, and so on.

2.2.3. Axioms

In addition to the IS-A hierarchy and internal structure of concepts, *axioms* are formal assertions that provide a way of representing more information about concepts, constraints on their internal structure, and their relations to one another. In the blocks world, we can use axioms to represent, for example, the following constraints:

a. Mutual constraints on the values of several properties of a single object, such as “The length of a brick is always greater than its height”

```
"X, Brick(X) => (Height(X) > Length(X))   (2.1)
```

b. Facts about the relations among objects, such as “Every block is supported by something”:

```
"X, Block(X) => $Y, Supported(X,Y)                    (2.2)
```

c. Constraints on property or role values for related objects, such as “No cube is supported by a smaller cube”:

```
"X"Y, (Cube (X) Ù Cube (Y) Ù Suppoted (X,Y))                   (2.3)
          => (Side-length(X) < Side-length(Y))
```

Logically there may be little difference between internal concept structure and axioms. One can define a concept using a frame formalism with roles and properties represented by slots of a frame. One can also express the same facts using axioms. For example, axiom 2.4 below states that all blocks must have a color property.

```
"X, Block(X) => $Y,(Color-value (Y) Ù Color(X) = Y)  (2.4)
```

Concept: Block
ISA: Object

Concept: Block
    ISA: Object
    Name: <String-value>
    Color: <Color-value>

Concept: Cube
    ISA: Block
    Side-length: <Number-value>

Concept: Brick
    ISA: Block
    Width: <Number-value>

Figure 3. Blocks World Ontology with Structured Concepts and Inheritance

An inheritance relation, too, can be represented using an axiom: Axiom 5 states that `Cube` is a sub-category of `Block`.

```
"X, Cube(X) => Block (X)                    (2.5)
```

The major advantages of using explicit taxonomy and structured concepts in ontology design are:

a. Focusing the attention of the inference or retrieval engine on what is most important. Taxonomic and slot-filler information is central to the identity of objects. It is helpful to identify this essential knowledge, rather than mixing it into a large database of axioms, most of which are rarely used.

b. Supporting special-purpose inference algorithms. Since taxonomic and slot-filler knowledge have a restricted logical form, special-purpose programs can use it efficiently, which is not possible if it is expressed in the more general formalism of logical axioms. The ability to specify default values for slot-fillers is a useful representation convention that is difficult to implement in a general logic-based inference system, since it requires non-monotonic reasoning (Reiter 1988)

c. Aiding human understanding of the ontology. The current state-of-the-art requires human participation in the construction of ontologies, although there is work aimed at automatically inducing taxonomies and other ontological data from large text corpuses (Riloff and Shepherd 1997). For the present, however, it is essential that researchers can view and understand ontologies that they build. Taxonomy and structured concepts, augmented by axioms, provide a more natural way for people to express and comprehend ontologies than a pure axiomatic approach.

2. 3 Two large-scale ontologies

2.3.1 The Cyc project

Fourteen years ago a comprehensive effort was commenced to create a general ontology for common sense knowledge: the Cyc project (http://www.cyc.com/cyc-2-1/cover.html ; Lenat 1990; Lenat and Guha 1990; Guha and Lenat 1994; Lenat 1995). Cyc contains more than 10,000 concept types used in the rules and facts encoded in the knowledge base. It includes all three of the elements discussed in the previous section: a large general taxonomy, structured concept definitions, and axioms.

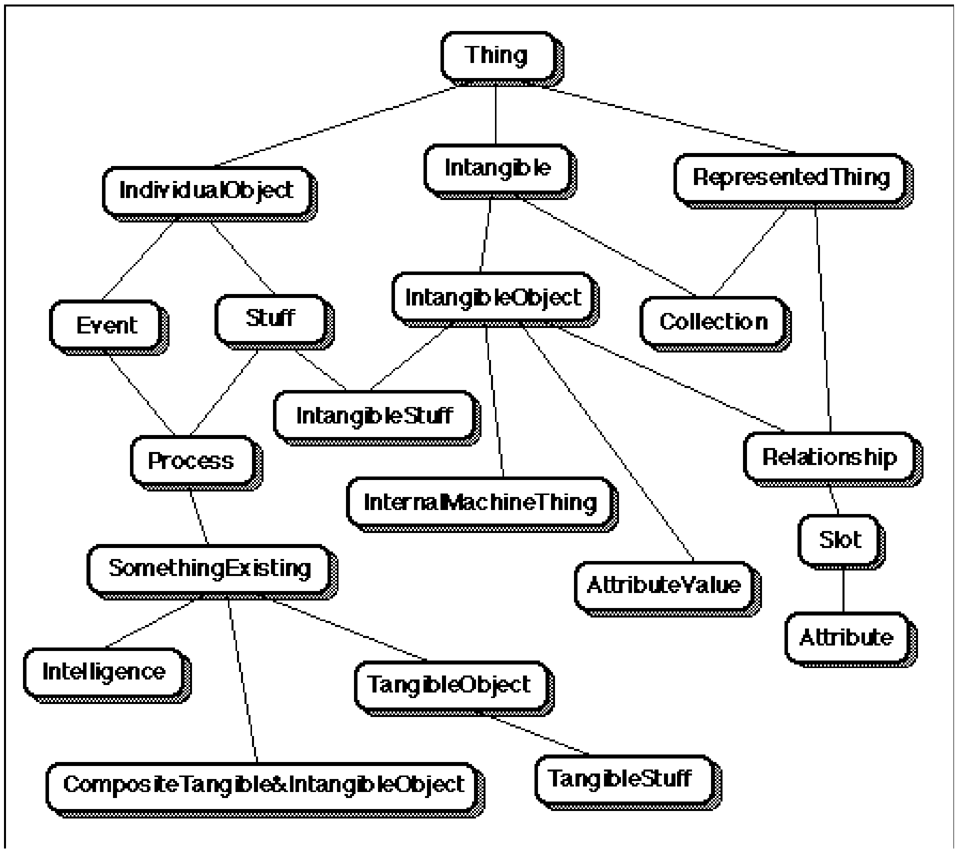


Figure 4. Cyc: Top-level categories (adapted from (Lenat and Guha 1990))

The upper level of the Cyc hierarchy is presented in Figure 4. At the top of the hierarchy is the *Thing* concept which does not have any properties of its own. The hierarchy under *Thing* is quite tangled. Not all the subcategories are exclusive. In general, *Thing* is partitioned in three ways:

- *RepresentedThing* versus *InternalMachineThing*. Every Cyc category must be an instance of one and only one of these sets. *InternalMachineThing* is anything that is local to the platform Cyc is running on (strings, numbers and so on). *RepresentedThing* is everything else.

- *IndividualObject* versus *Collection*. This is another total partition of *Things*. *Collections* include all the categories mentioned in Cyc. Hence, *Collection* doesn't have mass and is imperceptible.

- *Intangible* versus *TangibleObject* versus *CompositeTangible&IntangibleObject.* Every unit in Cyc is an instance of exactly one of these three categories. Intangible is anything that has no mass (set of all people, number42, etc.), whereas *TangibleObject* is anything that does have mass-energy (a rock, a person's body). *CompositeTangible&-IntangibleObject* is something that has both a physical extent and intangible extent. For example, a particular person has a body (physical extent) and mind (intangible extent).

2.3.2 WordNet

One of the best-developed lexical ontologies is WordNet, a manually-constructed on-line lexical reference system (ftp://clarity.princeton.edu/pub/wordnet/ ; Miller 1990). Lexical objects in WordNet are organized semantically (with the basic distinction between nouns, verbs, adjectives and adverbs). The central object in WordNet is a *synset*, a set of synonyms. If a word has more than one sense, it will appear in more than one synset. There are 70,000 synsets. Synsets are organized in a hierarchy via super-class/sub-class relationship (referred to as hypernymy/hyponymy). Part of WordNet hierarchy of tangible things is presented in Figure 5.

WordNet is primarily a taxonomy; it does not include structured concepts or axioms, and it represents only a very few non-taxonomic relations among concepts. (Concepts in WordNet are not entirely atomic, however, since each concept is represented as a list of the words used to name it.) For noun-synsets, there is one non-taxonomic relation represented: each noun has a pointer to other nouns representing its parts. For example, parts for the *bird* concept are *beak* and *wings.*

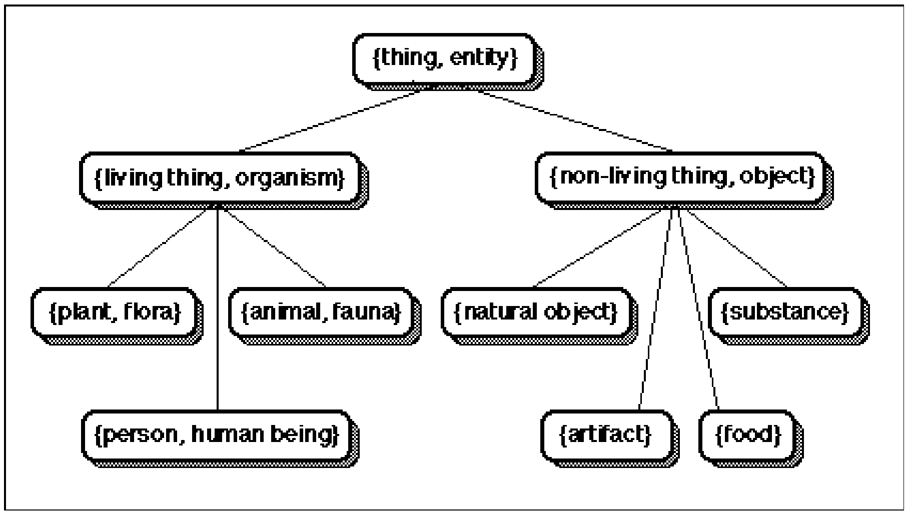


Figure 5. WORDNET: Representation of subclass relation among synsets denoting different kinds of tangible things (Miller 1990). Braces enclose concepts in the same synset.

Although WordNet uses a simple hierarchy for noun synsets, it employs a different organization of synsets for verbs and adjectives. Descriptive adjectives, are organized in bipolar clusters based on antonymy. For example, there is a bipolar cluster generated by *dry* and *wet* with synonyms of each of the adjectives at the corresponding side of the cluster. Verbs are divided into 15 clusters according to their meaning, with entailment being the primary relationship between the verbs in a cluster. Most of these clusters correspond to semantic domains: verbs of bodily care and functions, change, cognition, communication, competition, etc. Verbs such as *suffice, belong,* and *resemble* that do not belong to any of the semantic domains and refer to states, form a separate cluster.