Ontologies and How to Build Them

Xiaowei Yang

March 5, 2001

Abstract

This paper is the product of an area-exam study. It intends to explain the concept of ontology in the context of knowledge engineering research, which is a sub-area of artificial intelligence research. It introduces the state of the art on methodologies and tools for building ontologies. It also tries to point out some possible future directions for ontology research.

1 Introduction

This paper is the product of an area-exam study. It intends to explain the concept of ontology in the context of knowledge engineering research, which is a sub-area of artificial intelligence research. It also introduces the state of the art on methodologies and tools for building ontologies. It also tries to point out some possible future directions for ontology research. Some background knowledge of artificial intelligence and knowledge engineering is presented in Appendix A. Readers that are unfamiliar with the field of knowledge engineering are strongly recommended to read it first. I discuss the confusing definition of "ontology" in Section 2. A simple example is used to explain the concept and at the end of Section 2, I summarize the definition of an ontology. In the next two sections (Section 3 and 4), I present the methodologies and tools shown in the research literature for building ontologies. At last, I suggest some future research directions. To complete the research, in Section 6, I briefly introduce how ontologies are used in fields other than knowledge engineering.

2 What is an Ontology?

Ontology research is a branch of knowledge engineering. Some useful background knowledge is presented in Appendix A, and should be consulted first if the reader is unfamiliar with the concepts "knowledge", "knowledge level", and "knowledge representation".

In philosophy, "ontology" refers to the branch that systematically studies what actually exists. In the context of knowledge engineering, the definition of ontology is rather confusing. In his paper [25] "Understanding, Building, And Using Ontologies", Guarino lists eight different definitions of ontologies taken from the research literature. In my opinion, those definitions attempt to explain what an ontology is from three different aspects: the content of an ontology, the form of an ontology and the purpose of an ontology. The three aspects of an ontology are described as follows.

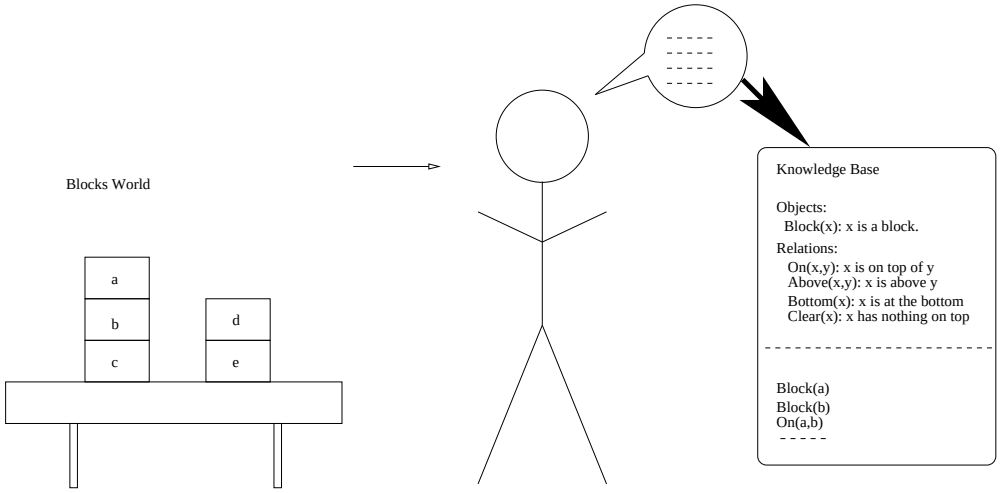

Figure 1: The Blocks World, the conceptualization and the ontology

2.1 What is the Content of an Ontology?

Gruber [18] defined an ontology as a specification of a conceptualization. Gruber's definition explains the content of an ontology but is confusing because it does not explain what a conceptualization is.

According to Genesereth [15], the formal representation of our knowledge about the world starts from a conceptualization. A conceptualization includes the objects and their relations which an agent presumes to exist in the world. The set of objects is called a *universe of discourse*. Here the word "object" has a general meaning. It refers to any concept that we want to talk about. It can be a concrete concept, such as a book, an abstract concept, such as right or wrong, an individual object, or an aggregate object. It can even be fictional, such as a unicorn. In my opinion, a conceptualization is a mental image. The process of a conceptualization is the process of mapping an object or a relation in the world to a representation in our mind.

Suppose we would like to build a knowledge-based system that reasons about the relative positions of blocks shown in Figure 1 [15, 35]. The first step is to formally represent our knowledge about the Blocks world. Our knowledge is expressed according to our conceptualization about the Blocks world.

In the universe of discourse, i.e., the Blocks world, there are five perceivable instances of the object type Block. Some blocks are placed at the bottom[1] and some are placed on top of another block. Some blocks have no block on top. An intelligent agent forms some concepts about this domain, such as the existence of an object type and that all five instances belong to the same object type. Furthermore, the intelligent agent can recognize that the objects have some spatial relations. Notice those concepts do not belong to a particular state of affairs. Even if the number of instances, or the spatial placement of the blocks are changed, such concepts still remain the same. Then the agent chooses some symbols such as "Block", "Bottom", "Clear", "Above", "On" to represent the concepts and relations. He chooses to use the symbols "a", "b", "c", "d", "e" to represent different blocks.

This set of symbols specifies the agent's conceptualization of the domain. He can write

[1]The table is not in the universe of discourse

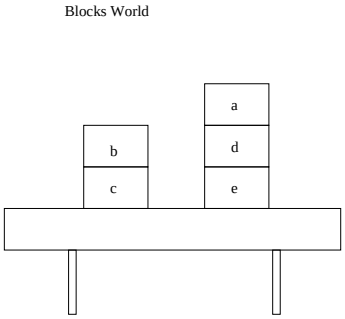

Figure 2: Another Configuration of the Blocks World

down his knowledge about the Blocks World in this set of symbols.

Object :

;;Five objects of the same type Block. Tag them with "a", "b", "c", "d", "e".
Block(a), Block(b), Block(c), Block(d), Block(e)

Relations :

;;Spatial relationships of Objects.
Bottom(c), Bottom(e)
Clear(a), Clear(d)
On(b,c), On(a,b), On(d,e)
Above(a,b), Above(b,c), Above(a,c), Above(d,e)

Genesereth defines the above concepts by their extensional entities [37], which are the set of objects the concepts denote. For example, the relation Clear is defined as the set $\{a, d\}$. This seems to associate the conceptualization with a specific configuration of the world. When the configuration is changed, as shown in Figure 2, is the conceptualization changed or not? According to Guarino [24], though the extension of the relation "Clear" has changed (Clear is denoted by $\{a, b\}$ in Figure 2, our intensional concept of "Clear" remains the same. We would like to explicitly point out that a conceptualization accounts for our intensional explanation of the world. In another word, the meanings of the symbols we use to represent our concepts remain the same even though the configuration of the world changes. The conceptualization about which an ontology specifies refers to the partial mental image that captures the relative constant concepts about the world.

2.2 What is the Form of an Ontology?

According to Neches [33], an ontology is a shared set of explicitly defined terminologies and relationships of a domain of interest, which comprises the vocabulary of the domain. This definition explains what an ontology usually looks like.

Unlike our toy knowledge-based system, the process of building a knowledge-based system (KBS) usually includes acquiring knowledge from human experts, formalizing the knowledge and representing it into computer data structures, and developing computer programs to utilize the knowledge to solve problems. Building a knowledge-based system from scratch is a very time-consuming task. A good portion of expertise knowledge consists of heuristic knowledge, which is not explicitly documented in textbooks and which may not be

clear to the experts themselves [11]. For example, a good Chinese chef may not be able to articulate the knowledge she implicitly uses when she cooks. Hence, knowledge acquisition is a slow process. After knowledge acquisition, it also takes a fair amount of time to formalize the knowledge and to encode it into a computer language. This has limited the size of many knowledge-based systems. To build truly large knowledge-based systems, knowledge reuse and sharing is necessary. This is very similar to software sharing and reuse efforts. Standard libraries for different applications are created to facilitate software development.

During their research, knowledge engineers have observed two phenomena [33]:

1. When building knowledge-based systems for different applications in the same domain, significant portions of the domain, such as object types, relations and constraints on combining them, must be repetitively represented.

2. To reuse knowledge-based built systems by others, a shared set of terminologies and an agreed interpretation about their meanings are necessary. For example, if two medical diagnose systems use different terms for the same symptom, it is difficult to combine the knowledge of the two systems for use in a wider variety of disease diagnoses. In fact, for two persons that speak different languages, it is almost impossible for them to share their knowledge, not to mention computer programs.

Therefore, knowledge engineers have proposed ontologies as one method of sharing and reusing knowledge. In our example of Blocks World, the meanings of Block, Clear, etc are implicitly represented. We understand them because we understand English. For more complicated concepts, such shared understandings may not be possible. One way to specify the meanings of the terms is to explicitly define them. Therefore, in order to share and reuse knowledge, we should explicitly define the terminologies in which we talk about the domain. Application knowledge bases should be organized into layers, where the top level is a shared ontology that defines the vocabulary for talking about a domain, the middle level is a customized ontology using the shared vocabulary to define a system-specific domain model and the bottom level specifies the run-time and state dependent knowledge [33].

From the knowledge sharing and reuse aspect, an ontology is therefore defined as a set of definitions of concepts, objects and their relationships in the domain of interest. This definition specifies the form of an ontology. The way to specify a conceptualization of a domain is to explicitly define the terms used to describe the domain.

2.3 What is the Purpose of an Ontology?

As mentioned before, an ontology is created to facilitate knowledge sharing and reuse. It works by explicitly defining terms, usually in a formal language, so that intelligent agents can agree with the meanings of the terms and use it in a way consistent with their meanings. An agreement to use a vocabulary consistent with the meanings that are defined in an ontology is called the ontological commitment.

Of course, the ultimate ontological commitment is between humans. We commit to an ontology and write programs to use the vocabulary consistent with the ontology. Thus, two agents or an agent and a human, committed to the same ontology can exchange their knowledge in a meaningful way.

2.4 A Summary

To summarize, an ontology is an explicit specification of a conceptualization. The conceptualization accounts for the intensional meaning of objects and their relations in a domain of interest. One way to specify an ontology is to define a set of terminologies, whose meanings are associated with concepts in the conceptualization. In the context of knowledge engineering, the set of terminologies serves as the vocabulary describing domain knowledge. One purpose of building ontologies is to reach an agreement about a conceptualization in order to facilitate knowledge reuse and sharing.

I notice, however, that although the same conceptualization can be specified differently, there is no explicit term to denote the conceptualization specified by an ontology . For example, the same conceptualization can be either described by a natural language or a formal language. I suggest the use of the phrase "knowledge level ontology" to denote the conceptualization of objects and their relations; and to use the word "ontology" to denote the specification of a knowledge level ontology.

2.5 General Ontologies and Domain Ontologies

Ontologies are divided loosely into two categories: general ontologies and domain ontologies. The domain of interest for a general ontology is the whole world. The concepts in a general ontology are often organized in a taxonomy, with the top concept being everything that can exist. Examples of general ontologies include the Upper CYC Ontology [28, 26, 4], Unified Medical Language System (UMLS) [3], Generalized Upper Model [27], Sowa's Ontology [37]. To get a feeling of what an ontology implementation looks like, below are some lines of code excerpted from the Upper CYC Ontology [4].

```
;;; #AFewDaysDuration
(#$isa #$AFewDaysDuration #$OrderOfMagnitudeInterval)
(#$isa #$AFewDaysDuration #$Time-Quantity)
(#$comment #$AFewDaysDuration ``Duration of 2 to 10 days'')
```

This piece of code is a definition of the term "AFewDaysDuration". In CYC's taxonomy, "AFewDaysDuration" is an element of the collection "OrderOfMagnitudeInterval" and "Time-Quantity". The comment says the concept means a "Duration of 2 to 10 days".

A domain ontology is the specification of a particular domain conceptualization. Examples of domain ontologies include TOVE [14, 21, 13, 2], Chemistry Ontology [29], Eng-Math [20]. Compared to general ontologies, domain ontologies have a variety of representation forms and may not have an explicit taxonomy.

2.6 Knowledge Representation Mechanisms

Before I proceed to discuss how ontologies are built, I introduce some knowledge representation mechanisms that are relevant to the discussion of ontologies. Readers with AI background SHOULD skip this section.

1. Logic [15] [40]. Logic is a formal language to represent our knowledge. There are two types of symbols in logic: variables and constants. Constants contain object constants, function constants and relations constants. An object constant represents a specific object in the world; a function constant maps objects to some other object; a relation constant represents a relation between its arguments. For example,

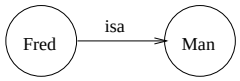

Figure 3: An Example of A Semantic Net Representation

in logic, the knowledge "Fred is a man" can be represented by the atomic logical sentence "Man(Fred)", where "Man" is a relation constant and "Fred" is an object constant. Logic sentences can be connected by logical operators $\vee$(disjunction, or), $\wedge$(conjunction, and), $\neg$(negation, not) and $\Rightarrow$(implication) to make more complex statements. A logical sentence can also contain two types of quantifiers: the universal quantifier ($\forall$), which provide a way to say something about all objects in the world and the existential quantifier ($\exists$), which provides a way to say something about an individual object without explicitly naming it.

Here are some examples of what we can say in logic.

(a) All birds have feathers.

$$\forall x \; Bird(x) \Rightarrow Feather(x)$$

(b) Some birds can fly.

$$\exists x \; Bird(x) \Rightarrow Fly(x)$$

Inference is the process of generating new expressions from old ones. Logical inference is sanctioned by inference rules. The characteristics of logic inference is truth preserving. One example is Modus Ponens , which says if $A \Rightarrow B$ is true and A is true, then we can infer B is true.

First-Order logic is a variant of logic where variables can only represent objects and can not represent relations. It is the most commonly used logic language. Prolog is a widely used first-order logic implementation language.

When knowledge is represented by logic, we view knowledge as a set of logical sentences and reasoning is done by thinking about what is true and what follows from it.

2. Semantic nets [40]. A semantic net is a directed graph, with nodes denoting objects, links denoting relations between objects, and link labels denoting the particular relations. Figure 3 shows an example of a semantic net, representing the same knowledge "Fred is a man".

   The motivation of the semantic nets representation is to model human memory and its use for language processing.

3. Frames. The language of frames is an object-centered language. A sentence in this language is in the form of a frame. A frame represents a prototype. A frame can contain several slots, which denotes functions or relations. Slots are assigned values. The representation of frames is inspired by how a human understands and reasons about the world. Figure 4 shows an example of a frame, which says something about Fred: "Fred is a man and likes movies".

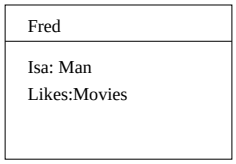

Figure 4: An Example of A Frame Representation

3 Methodologies for Building Ontologies

Like any other computer system design and implementation, building ontologies is an art, rather than engineering. There is no standard methodology for building an ontology. Individual researchers have taken different approaches. Several researchers have summarized some methodologies from their own experience of building ontologies. These include the methodologies described by Uschold [39], "METHONTOLOGY" [29, 12] developed by Gomez-Perez and a formal approach by Gruninger [23]. All these methodologies are for building relatively small domain ontologies from scratch.

3.1 A Common Methodology

Uschold's methodology and Gomez-Perez's METHONTOLOGY have a lot in common, while Gruninger's methodology is more formal and different.

According to Uschold and Gomez-Perez, during an ontology development, there are nine major activities happening. The activities do not have to happen sequentially. Some may happen in parallel, and some may happen in a loop. Below is a list and discussion of these nine activities.

1. Planning. Before building an ontology, a plan should be made about what should be done, who will do them, and what resources are needed.

   This is a common step for starting any project, though some authors did not mention it in their methodology descriptions.

2. Specification. In this step, the scope and purpose of an ontology should be specified. Why is the ontology being built? What is its intended usage? What is its scope? It is recommended that a specification document that answers the question be written. Gomez-Perez also suggests that the specification include the set of domain terminologies in the ontology. In her example of building a chemistry ontology, the terminologies are well-documented in the handbook of chemistry. I think in most cases, this is not true. The set of terminologies to represent objects, relations and constraints are not well documented. Knowledge acquisition and conceptualization are needed to identify the set of terms. Thus, I think it is not necessary to specify the domain terminologies at this step, if it is not easy to do so. The major goal of this activity is to identify the scope and purpose of the ontology.

3. Knowledge Acquisition. Usually, one does not necessarily have enough domain knowledge to know all terms and their meanings in the domain. Thus, knowledge acquisition is unavoidable. In this step, one should be able to locate a set of knowledge sources, such as domain experts, textbooks, and other publications. One should use knowledge

acquisition techniques to capture relevant domain knowledge. For example, if we are going to build an ontology for medical terms, we need to read medical books, talk to doctors etc.

4. Conceptualization. This activity requires a lot of intellectual thought to structure the domain knowledge. What are the concepts existing in the domain? What are the relations between two concepts? How should concepts and relations be defined? What are general concepts and what are special ones? What are the attributes of a concept that help us identify the concept? The result of this mental exercise is a conceptualization of the domain. The final ontology is an explicit specification of the conceptualization.

   Both Uschold and Gomez-Perez propose representing the conceptualization in some intermediate representation (IR). Uschold proposes using natural language as the intermediate representation to define the terminologies and their relations, producing a specification for the implementation of the ontology. Gomez-Perez suggests using a set of tables such as concept tables, binary relation tables, and attribute tables as the intermediate representation. Table contents are specified in a more restricted syntax than the natural language. Definitions in IR can be automatically translated into the language Ontolingua by the tool Ontology Design Environment (ODE).

   I notice that the set of IR definitions may serve two purposes. First, it serves as the specification document or an intermediate development result for ontology developers. Second, it may serve as part of the ontology documentation to explain the conceptualization to non-technical users, or to domain experts to assist knowledge acquisition. Therefore, turning the conceptualization into an intermediate representation is useful. The set of IRs should be maximized in the fidelity of representing conceptualizations. In the mean time, it should also facilitate the implementation of an ontology into a computer language. These two purposes are not always compatible with each other. For example, the concept of "car" may be best illustrated by a picture of a real car to a human reader. However, this does not help with formalizing a definition of a car by a computer language.

   A knowledge representation reflects some of our particular viewpoints about the world [8]. Therefore, a particular representation scheme, formal or informal, may not fit all conceptualizations. Hence, I propose the use of a variety of representation schemes as the intermediate representation. Since the intermediate representation is not for computer processing, it can take advantage of the expressive power of natural languages and graphics representations, with the restriction that it should provide enough hints for being translated into a symbol level encoding. Therefore, not only natural languages, binary tables, but also other representations such as semantic nets and frames, are all candidates for intermediate representations. Pictures may also serve as an auxiliary representation for illustrating a terminology.

   The two activities, knowledge acquisition and conceptualization are highly interleaved. Uschold, in his paper, uses the phrase "ontology capturing" to encompass both activities. Conceptualization requires domain knowledge and it is during the process of knowledge acquisition that some part of the conceptualization is forming.

   According to Uschold [39], during the process of ontology capturing, the first step is to produce the glossary of terms. Those terms can then be loosely grouped into working areas. For example, noun terms and verb terms can be grouped into different

areas. The third step is to define terms in each work area in turn and start with the one that overlaps most with other work areas. The most overlapping area is the area that has the most relations with other work areas.

To produce definitions, Uschold recommends taking the middle-out approach, which means defining the most fundamental terms before moving to more general or abstract terms. The other two approaches, bottom-up (starting from the most specific terms), and top-down (starting from the most general terms), both have some disadvantages. There is some psycholinguistic evidence supporting this argument. According to some psycho-linguists [31], nouns (in a natural language) are organized in hierarchies in our lexical memory. For those hierarchies of concepts, there is a level somewhere in the middle, where most of the distinguishing features are attached. Below this level, only a few features are added which help distinguish object types; above this level, descriptions are brief and perhaps too general. For example, if we are going to build an ontology for animals, we may want to start with the concept "dog", instead of the concept "mammal" or concepts of special species of dogs.

A top-down approach controls the level of detail very well. We may choose to stop at any level. However, because the descriptions of top level concepts are brief, this approach often results in an arbitrary top level division. For example, none of the three well-developed general domain ontologies UMLS, the Upper CYC Ontology, the Generalized Upper Model agree at the top level.

The bottom-up approach has less control of the level of detail. Working from the lowest level leads to a high degree of detail and makes it difficult to find common features between related concepts. Therefore, a middle-out approach can balance the level of detail. Moreover, by starting from the level of most distinguished features, it is easier to spot common features and classify higher level concepts.

All three approaches are widely adopted in real practice. Again, I think there is no single approach that fits all domains. If the developers have a very good high level understanding of the domain, it is better to start from the top-down, as the top level concepts can then be used to define lower level concepts, producing more concise definitions.

5. Integration. Existing ontologies that may help to develop the current ontology should be integrated as much as possible, to avoid duplicating efforts. For example, in her Chemical ontology, Gomez-Perez reuses Gruber's EngMath ontology for the definition of physical quantity.

6. Implementation. This step involves choosing a computer language and implementing the ontology. In Gomez-Perez's METHONTOLOGY [12], this activity is divided into two activities: formalization and implementation. The computer language, as a knowledge representation, has its own meta-ontological commitment. For example, the language KIF [32] has its built-in real-number and set-theory ontologies and it is a first-order logic based language. The ontological commitment of the language should be consistent with our conceptualization, whether characterized by first-order logic, semantic nets or frames. Deciding a representation formalism is called "formalization" in METHONTOLOGY. Based on this formalization, an implementation language is chosen to implement the ontology.

7. Evaluation. Unlike a KBS, whose evaluation can be done by comparing its performance with an expert's performance, there are no specific methods to evaluate ontologies. Gruber [18] has proposed a set of design criteria for ontology development. The EngMath ontology was evaluated against the criteria and was put under several revisions. I feel this set of criteria may offer some primitive guidelines for evaluating ontologies as well as for ontology development.

   (a) Clarity: Definitions in an ontology should be clear and documented with the natural language. Logical axioms should be used to state definitions whenever possible.

   (b) Coherence: Both the formal definitions such as logical axioms and the informal definitions should be consistent.

   (c) Extensibility: An ontology should be designed to be a conceptual framework for a range of anticipated tasks. One should be able to define new terms based on existing vocabularies monotonically. For example, in the engineering math ontology developed by Gruber, if the set of units of measure is fixed, then the ontology is not good for extensibility, as there are many standards of units of measure. Thus, the ontology should not be limited to one type of measurement units. Instead, a way to define a standard set of units should be provided.

   (d) Minimal encoding bias: Encoding bias refers to a representation choice that depends on the symbol level encoding. For example, in the EngMath ontology, if we specify the magnitude of a physical quantity to be "double-float", we introduce an encoding bias. The precision is an implementation level rather than a knowledge level specification. A better way to design the ontology is to specify it as a "real-number".

   (e) Minimal ontological commitment: In a simple wording, an ontology should make as few claims as possible. An ontology must be consistent, but it does not have to be complete. The more vocabulary and constraints an ontology has, the more likely that some definitions will be incompatible with future representation needs.

   Two techniques for achieving extensibility and minimal ontological commitment are presented by Gruber. The first one is modular design. A monolithic ontology is decomposed into a set of loosely coupled ontologies. The EngMath ontology is decomposed into several sub-ontologies, each of which comprises a set of definitions. The definitions in EngMath ontology is the union of them. In this design, one can only commit to a subset of definitions and axioms of the entire ontology, achieving the goal of minimal ontology commitment. One can also extend a subset of definitions without affecting the rest of the ontology. Therefore, a base ontology can be extended to different specific ontologies.

   The other technique is parameterization of conventions. This technique is similar to defining templates in C++. For example, the standard units for different systems may be different and this affects the vocabulary for a particular domain model. In EngMath, instead of stating them as global constants, which limits extensibility and enlarges ontological commitments, the standard units are specified as parameters.

   I feel there is one missing criterion – Conceptual coverage. An ontology should not include over-specific concepts to hinder its extensibility, but it should cover the concepts that it intends to cover.

8. Documentation. An ontology should be documented to facilitate sharing and reuse. This is also a common activity for software development. As Gruber suggests, the implemented ontology should include natural language definitions for every definition and every axiom implemented in the ontology.

9. Maintenance. As with any software product, this is also a necessary activity. There may be missing concepts or unclear definitions detected later by ontology users, or new concepts may be introduced into the domain. The task of maintenance is to keep the ontology up to date.

Most of the activities are common to the development of a software production. The unique activities are the knowledge acquisition and conceptualization. I imagine they also entail the most difficult work.

3.2 A Formal Methodology

So far, we have described the most common activities in an ontology design process. Gruninger proposes a methodology to develop a formal ontology from his experience of developing the enterprise ontology TOVE. I have not seen any other ontology developed in this way. His methodology includes the following stages.

1. Motivating Scenarios. What are the motivations for developing a new ontology or extending an existing ontology? The motivating scenarios often arise in applications where there are story problems or examples that are not handled appropriately by existing ontologies.

2. Informal Competency Question [22]. Given the motivation scenarios, a set of questions should arise. These questions must be answerable by the ontology. These questions characterize the scope and purpose of the ontology. For example, in the development of the Enterprise Ontology [21], one competence question involves temporal projection – "given a set of actions that occur at different points in the future, what are the properties of resources and activities at arbitrary points in time?

3. Terminology in First-Order Logic. To formally describe the competence questions and their solutions, the terminology must be specified using some formalism, such as the first-order logic. For example, to answer the temporal projection question, the definitions of time, activities and resources must be specified [14].

4. Formal Competency Questions. At this step, the competence question is expressed in the formal terminology.

5. Axioms in First-Order Logic. Axioms in the ontology specify the meanings of the terms by restricting their interpretations. Axioms are added in order to answer the competence question. The set of axioms in the ontology plus object instances should be able to logically infer the solution of the competence question.

6. Completeness Theorems. The completeness theorems state that the axioms in the ontologies and object instances are necessary and sufficient to represent the competence questions and their solutions. With any single axiom removed, we are not able to prove the theorems. Only with the entire set of axioms, are we able to prove them.

In Gruninger's methodology, the competence questions play a dominating role. In the stages of motivating scenarios and informal competence questions, the competence questions specify the purpose and scope of the ontologies. The competence questions guide the development of terminology and axioms in the first-order logic. The terminology and axioms are specified in order to formalize the competence questions and provide a solution to them. At the last stage, the completeness theorems with respect to the competence questions are used to evaluate the ontology.

I feel that Gruninger's methodology actually develops a consistent and complete (with respect to the competence questions) logical theory. The evaluation process of ontologies developed under this methodology is relatively simple. It is done by proving the completeness theorems. However, an ontology is different from a knowledge-based system in that its main task is to define vocabularies instead of solving problems. The logical theory defined by an ontology should be consistent, but it does not have to be complete. Axioms or terminologies may be later added to the ontology for an individual specific application. Moreover, if the motivations for developing an ontology are not easily specified by a logical theorem, this methodology is also inappropriate. For example, the motivation for developing CYC is to represent all human knowledge. I see no easy way to come up with a set of provable logical theorems to describe the motivation. Thus, I do not think this is a general approach for developing ontologies.

3.3 Missing Methodologies

3.3.1 How to Build General Ontologies

I have not come across any literature that systematically discusses the methodology for building a general ontology. A general ontology may contain thousands of concepts. And the top level division of a general ontology often requires a fair amount of philosophical thought. Both the designers of the CYC ontology and Sowa have mentioned that philosophy plays an important role in deciding the top level divisions of their ontologies. However, exactly how the rest is to be developed is left unspecified.

3.3.2 How to Build Ontologies from Existing Ontologies

There is also little work which describes methodologies for building a new ontology from existing ones. In [38], Swartout et. al. describes their method of building a domain specific ontology (military air campaign planning) from a general ontology (SENSUS). The general ontology is organized in a taxonomy. The process starts with a small number (approximately 60) of "seed" terms that are specified by domain experts. First, the "seed" terms are linked to the general ontology. Second, all terms on the path from the "seed" terms to the root of the general ontology are included. If there are nodes that have a large number of paths through them, in some cases the entire subtree under the node is included. The resulting ontology has approximately 1600 concepts.

In my opinion, this method seems to be a good approach for developing a domain specific ontology. As most non-domain specific concepts can be reused, it takes substantially less time to build the ontology. Since the entire domain ontology exists as a sub-ontology of the general ontology, it can be easily extended by attaching more "seed" terms to the general ontology. I would like to see more research work devoted to exploring the power of this methodology.

4 Tools for Developing Ontologies

Developing tools for ontologies often contain an ontology editor as the front end and an ontology repository as the backend. The ontology server developed at Knowledge System Lab at Stanford is the most well-known system in this area [17, 19, 9, 10].

4.1 Ontolingua and KSL Ontology Server

The front end of the ontology server is a web browser. The backend ontology server generates dynamic HTML pages which render an ontology editing environment to the users. Users can browse ontologies stored at the server, create their own ontologies and store them at the server. A group of users can log into one session and collaboratively work on the same ontology. The server will notify each user of the changes other users have made. The server also provides users the ability to browse older versions of the ontology they create.

The ontology server allows a user to include other ontologies from its stored ontology library and automatically detects conflicts and resolves them. For example, a term with the same syntactic form may be defined in different ontologies. When a user includes both ontologies, there is a conflict. The server will automatically attach the string "@*ontology name*" to the term to resolve the conflict. The created ontology at the server can be translated by a system called Ontolingua to different implementation languages. "Ontolingua" is also the name of a language, which is an extension of KIF. Users can also download ontologies from the server and send queries to the server to get the definition of a term.

4.2 Ontology Editing Environment (OED)

OED is a tool developed by Gomez-Perez et. al [5, 16, 29]. It is an ontology editor that allows users to manage and create ontologies in tabular notation, which is an intermediate representation which they defined. Ontologies created in this fashion can be translated into different implementation languages. As the intermediate representation is in table format, the entire ontology is stored in a relational database. This allows other applications to retrieve the knowledge via SQL queries.

4.3 Ontosaurus Browser

Ontosaurus is another ontology server developed [38] at ISI. The front end is again a web browser. Ontosaurus has a slightly different interface than the KSL ontology server but similar functions. Users can include images for documentation purposes. Ontosaurus is Loom (a computer language) based. Translations from Loom to C++ are provided by the server.

The common features of the tools include: 1. a user interface to facilitate the ontology navigation; 2. A translation scheme that automatically generates ontology codes.

5 Areas Deserving Further Exploration

During my study, I found that there are some issues not clearly addressed by the research community.

5.1 The Making of a Definition

An ontology consists of a set of definitions. A definition of a symbol is supposed to explain the meanings of the symbol. However, a definition itself is composed of other symbols. Unless the meanings of those symbols are known, the definition remains vague. How are those symbols defined? By this argument, in any ontology, there remain some symbols whose meanings are not explicitly defined in terms of other symbols. "Every Knowledge base has a bottom" [6], and an ontology also has its bottom terms.

In our example of the Blocks World, we can define "Clear" in terms of "On" by adding logical assertions.

$$\forall x\ Clear(x) \Leftrightarrow \forall y \neg On(y, x)$$

If we understand the meaning of "On", there is no problem for us to catch the concept of "Clear". Similarly, we can define "Above", "Bottom" in terms of "On".

$$\begin{aligned} \forall x\ Bottom(x) &\Leftrightarrow \forall y \neg On(x, y) \\ \forall x \forall y\ On(x, y) &\Rightarrow Above(x, y) \\ \forall x \forall y \forall z\ On(x, y) \wedge Above(y, z) &\Rightarrow Above(x, z) \end{aligned}$$

If we know the extension entity of "On", the above definitions uniquely define the extension entities of "Above", "Clear" and "Bottom". In the example of Figure 1, the extension entity of "On" is $\{< a, b >, < b, c >, < d, e >\}$. By the above definitions, "Clear" corresponds to $\{a, d\}$, "Bottom" corresponds to $\{c, e\}$, and "Above" corresponds to $\{< a, b >, < b, c >, < a, c >, < d, e >\}$.

How shall we define "On"? We can add more logical constraints to restrict the valid interpretation of "On". For example, we can add the following axioms into our definition:

$$\begin{aligned} \forall x\ \neg On(x, x) \\ \forall x \forall y On(x, y) &\Rightarrow \neg On(y, x) \\ \forall x \forall y \forall z On(x, y) \vee On(y, z) &\Rightarrow \neg On(x, z) \end{aligned}$$

The constraints describe the symbol "On" as representing a non-reflective, non-symmetric and non-transitive relation. However, for a non-English speaker, it is not enough for him to understand the meaning of "On". What else can we do? As we have said, an ontology is for making ontological commitments. The ultimate ontological commitments are made by humans, by those who write computer programs. Therefore, even we can not provide a logical definition for "On" or other symbols. We expect that humans will get the meanings of the symbols either from the symbols themselves or by reading our natural language definitions. In the excerpt of the CYC Upper Ontology, the number of days denoted by the symbol "AFewDaysDuration" is defined in the comments.

Here arise the following questions. When designing an ontology, how much do we have to depend on hints from natural languages to make a definition? That is, what are the "bottom terms" in an ontology that we can not rigorously define? When we have a choice between revealing the meanings via some hints from natural languages and logical axioms, which way shall we choose? How do we strike a balance between revealing meanings via natural language hints and spending extra effort to define them? In my opinion, when we build an ontology, we should try to limit the number of bottom terms, choose the bottom terms with simple and clear meanings and use logical axioms to constrain their interpretations.

The function of bottom terms is similar to the basis in a vector space. All other terms can be defined by them. I think how to choose the set of bottom terms is an interesting and challenging problem.

Moreover, I have not seen any research work that clearly articulates the methodologies to define a concept. A concept sometimes is defined by inheritance, where its meaning is partially inherited from the upper class, or by specifying what attributes it has, or by logical axioms.

It is worthy noting that explicitly defining a concept is not a requirement for building knowledge-based systems. If our knowledge is not going to be shared, we can simply keep the meanings "secret" to us. We are able to explain the new expressions produced by the system. It is only for the purpose of sharing that we need to let others know what entity a symbol represents. As the main output of an ontology is a set of definitions, I believe providing some guidelines on how to make a clear and concise definition will help with the design and implementation of ontologies.

5.2 The Use of Ontologies

It is a hypothesis that building a shared ontology can facilitate knowledge sharing and reuse. How are ontologies reused in practice? Do knowledge engineers find that ontologies are very useful for building their knowledge-based systems?

There are some doubts on how useful an ontology is. Depending on different tasks, a conceptualization of a domain might be different. In the example of Blocks World, if our task is to move the blocks, we may need to describe the weight of the block. To build a KBS for a particular application, we may find out that the existing ontology does not totally fit into our conceptualization. Which is easier for knowledge engineers, to extend or modify the existing ontologies or to start from scratch? What kind of ontologies are more useful to knowledge engineers than others? What kind of definitions are easier to understand? Different opinions exist on how to build an ontology. In Gruninger's method [23], an ontology contains enough information to deduce the solutions to the competency questions, while according to Gruber, an ontology is merely a vocabulary of terms defined in a human-readable or machine-readable text [19]). What type of ontologies do knowledge engineers prefer to have? As there is no objective method to evaluate an ontology, I hope the feedback from the knowledge engineer can give us some insight on it.

I think it will be beneficial to have more research work done on evaluating ontologies through the experience of building knowledge-based systems from ontologies.

5.3 Performance Improvement of Tools For Distributed Development

The existing editing tools KSL Ontology Server and Ontosaurus require users to interact with a single server through a web browser to create, edit and browse an ontology. Users have to tolerate the network delay and server response delay. A better architecture, I think, is to use the ontology server as an ontology repository, using Current Version Systems to allow developers to check out or check in ontologies. Developers can install the translator Ontolingua and the ontology editor locally. During development, developers work on their local copies of the ontologies, which will reduce the network delay and server delay. The changes each individual developer makes will be noticed by others when he commits his copy to the ontology server.

5.4 More Methodologies

As discussed in Section 3.3, most research work on methodologies focuses on the building of a small scale ontology from scratch. We would like to see more research on how to build a large-scale general ontology and how to reuse existing ontologies to build new ones.

5.5 Intermediate Representation

In the process of building an ontology, an intermediate representation can serve as both a specification to developers and as a documentation for communicating with domain experts and users. ODE is a tool for creating table format intermediate representations and translating the intermediate representation to implementation languages. I think there should be more tools for creating ontologies in other intermediate representations. For example, an ontology that has taxonomy structure may be easily specified by a directed graph. A tool may help translate the graph into some code that describes the class hierarchy. On the other hand, some developers may prefer writing code without creating intermediate representations. There should also be tools that can translate code into some easy-to-understand representations. For example, I have found the code for specifying the Upper CYC Model hard to follow. It would be nice if there were some translator that can draw a directed graph that could capture the class hierarchy in the ontology. One example is the Java to Universal Modeling Language (UML) translation. Code written in Java can be translated to the graphical language UML to facilitate understanding.

5.6 Miscellaneous

1. A Global Naming Space for Ontologies
   There seems to be no global naming space for ontologies. When two agents want to exchange knowledge, they may need to know what ontology the other one commits to. Therefore, there should be a way to uniquely name an ontology globally. I suggest that an ontology use the fully quantified domain name as part of its name to provide global uniqueness. For example, a chemistry ontology created at KSL may be named as chemistry.ontology.www-ksl-svc.stanford.edu, while a chemistry ontology created by the author can be named as chemistry.ontology.cordelia.lcs.mit.edu.

2. Ontology Integration
   A systematic methodology for integrating ontologies also appears to be lacking. How shall we choose an ontology to be integrated? How shall we deal with definitions inconsistent with our conceptualization? We hope to see more research work on methodologies of ontology integration.

3. The development of more applications which use ontologies
   Ontologies can be used in a variety of fields, such as information retrieval, multi-agent systems etc. In the emerging field of ubiquitous computing, an ontology of devices will enable semantic device discovery and inter-operation. We hope to see more applications of ontologies in the future.

6 Ontologies in Other Fields

The idea of an ontology is so powerful that it has been used in fields other than knowledge engineering. To complete the research, here I briefly introduce two ontology applications in fields other than the knowledge engineering.

1. Data Integration [36]
   In this age of information , there is an increasing need for accessing data from multiple information systems. However, various information systems are likely to differ in naming and format specifications of data. A three-step process can be used to resolve semantic differences. First, a shared ontology is specified. Second, information systems commit to the ontology by describing objects using the ontology. Third, the similarities and differences between objects are exposed by the ontology. For example, in a medical database DB1, the medical term A may be used to refer to the symptom fever. Meanwhile, in a medical database DB2, a difference term "B" is used to refer to fever. To integrate a patient's record from one database to another, both databases can commit to UMLS and describe the terms A and B via the vocabulary defined in UMLS. Thus, the semantic relation between the syntactically different term "A" and "B" is exposed and the patient's data can be correctly incorporated from one to another.

2. Semantic Web [1]

   The WWW has now become a rich information source. The WWW consortium has proposed to use the Resource Description Framework (RDF) to describe the semantics of the information. RDF schema is used to specify resources and their interrelations, creating an ontology for the resources in the domain. A web page that commits to the ontology uses the vocabulary in the RDF schema to describe the information presented in the page. For example, if a schema for book information is built, then a software agent that helps users to buy a particular book can retrieve information from multiple sites, comparing the prices at different sites.

7 Summary

In this paper, I have described what an ontology is in the context of knowledge engineering. I also presented the state of the art on tools and methodologies for building them. I pointed out some unanswered questions. Hopefully, future research will provide us a satisfying answer.

8 Acknowledgments

I would like to thank Professor Randall Davis for helping me decide the area for the exam, choose papers and answer my questions during the exam. It is a great intellectual adventure. I enjoyed it.

Appendix

A Background

Artificial Intelligence (AI) is the science and engineering of studying intelligent behaviors and making intelligent machines, especially intelligent computer programs [30]. Intelligence is characterized by the computational part of the ability of taking actions to achieve goals. An intelligent agent is viewed as consisting of a perceptual system, a memory system, a processing system, a motor system and so on [34]. As an intelligent agent seems to know their environment and the consequences of their actions, we account for this rational behavior by assuming they possess knowledge of the environment [15]. The Physical Symbol System Hypothesis [7] states that a physical symbol system, which consists of a set of symbols, a set of expressions, and a set of procedures that operate on expressions to produce other expressions, has the necessary and sufficient means for general intelligent action. This leads to the following assumption: by approximately representing what an agent knows by symbols, a physical symbol system, such as a computer, can become capable of intelligent behaviors. Expert systems are built in this fashion. Knowledge from experts are "cloned" to computer programs, so that computer programs can give advice, solve problems as an expert is capable of doing.

A.1 Knowledge and Knowledge Level

In AI, knowledge is defined in a "black box" fashion [34]. Instead of describing knowledge as a physical object with particular properties and relations, knowledge is described by the external behavior of the agent who possesses it. Knowledge is, " whatever can be ascribed to an agent, such that its behavior can be computed according to the principle of rationality". The principle of rationality is characterized by an intelligent agent's behavior: "if an agent has knowledge that one of its actions will lead to one of its goals, then the agent will select that action".

As characterized by Newell [34], a computer system consists of five distinct physical abstraction levels. Starting from the bottom, these are the device level, the circuit level, the logic level, the register-transfer level and then the symbolic level. Each level consists of a medium that is to be processed and a law of behavior to determine how the system behaves at that level. For instance, at the symbol level, the medium to be processed consists of symbols and expressions; the law of behavior is the sequential interpretation of those symbols. The Knowledge Level Hypothesis introduces another abstraction level into computer systems. The Hypothesis says "there exists a distinct computer system level, lying immediately above the symbol level, which is characterized by knowledge as the medium and the principle of rationality as the law of behavior". The Knowledge Level Hypothesis implies that despite different symbol level encodings, two systems can show the same intelligent behaviors at the knowledge level.

A.2 Knowledge Representation

Though we can characterize knowledge by the external behavior of an intelligent agent, to design systems capable of such behavior, we need to encapsulate knowledge at the physical levels of a computer system. Therefore, knowledge needs to be explicitly represented by symbols at the symbol level. We then write down a procedure to manipulate the symbols

in a way consistent with what the symbols are intended to represent. A computer program therefore can show intelligent behaviors. Researchers have studied how to represent knowledge by symbols.

As summarized by Davis et. al. [8], a knowledge representation plays five roles in intelligent systems.

1. A knowledge representation is a surrogate. Objects in the real world can not be physically stored internally by any intelligent system, be it a computer or a human being. To reason about them, they need to be represented by something that can be internally stored inside a computer, or a human's brain. Thus, an intelligent agent can reason about the world by manipulating the representations, instead of taking actions on the real objects. The correspondence between the representation and the entities they represent is called the semantics of the representation. All representations are approximations of the real things they represent. In computer systems, the representations are symbols at the symbol level.

2. A knowledge representation is a set of ontological commitments. Before we proceed to explain in detail what an ontology is and what ontological commitments are, we can regard ontology as the study for existence for now. Since all representations are unavoidably biased – a surrogate of an object must have ignored some parts of the object while representing other parts of the object–by selecting a representation, we have already made a decision about what exists from our perspective of the world. A knowledge representation reflects how we view the world. Different representations reveal quite different views about the world.

3. A knowledge representation is a fragmentary theory of intelligent reasoning. The way we represent the world is typically motivated by some insight on how we reason about the world. For example, the predicate logic representation is inspired by the phenomenon that we sometimes reason by logical inference. This dates back to Aristotle [37] who took note of this and invented the syllogism for logical reasoning. An example of a syllogism is:

   - If all men are good,
   - and Fred is a man,
   - then Fred is good

4. A knowledge representation is a medium for pragmatically efficient computation. Representations typically offer suggestions on how to organize information to facilitate computing on the knowledge they represent. For example, the frame representation organizes information in an object-centered manner, which facilitates the execution of reasoning by inheritance.

5. A knowledge representation is a medium of human expression. Knowledge representations are also means by which we express our knowledge to each other, or to machines. In that sense, natural language is a form of knowledge representation.

References

[1] Semantic web. http://www.w3.org/2000/01/sw/, March 2001.

[2] TOVE Manual. http://www.eil.utoronto.ca/tove/ontoTOC.html, March 2001.

[3] Unified medical language system. http://www.nlm.nih.gov/research/umls/umlsmain.html, March 2001.

[4] The upper cyc ontology. http://www.cyc.com/publications.html, March 2001.

[5] M. Blazquez, M. Fernandez, J.M. Garcia-Pinar, and A. Gomez-Perez. Building Ontologies at the Knowledge Level using the Ontology Design Environment. In *KAW'98: the 11th International Workshop on Knowledge Acquisition, Modeling and Mangement*, Banff, Canada, October 1998.

[6] Randall Davis. Addressed in Lecturing, Febuary 2001.

[7] Randall Davis. Leture 1. Lecture Notes for 6.871: Knowledge-based Systems, Febuary 2001.

[8] Randall Davis, Howard Shrobe, and Peter Szolovits. What is a knowledge representation. *AI Magazine*, 14(1):17–33, 1993.

[9] Adam Farquhar, Richard Fikes, Wanda Pratt, and James Rice. Collaborative ontology construction for integration. Technical report, Knowledge Systems Laboratory, Stanford University, August 1995.

[10] Adam Farquhar, Richard Fikes, and James Rice. The Ontolingua Server: A Tool for Collaborative Ontology Construction. Technical report, Knowledge Systems Laboratory, Stanford University, September 1996.

[11] Edward A. Feigenbaum. Knowledge Engineering. *The Applied Side of Artificial Intelligence, Annals New York Academy of Sciences*, (91-107), 1980.

[12] Mariano Fernandez, Asuncion Gomez-Perez, and Natalia Juristo. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In *the AAAI-97 Spring Symposium Series on Ontological Engineering*, pages 33–40, March 1997.

[13] Mark S. Fox, John F. Chionglo, and Fadi G. Fadel. A Common-Sense Model of the Enterprise. In *the Second Industrial Engineering Research Council*, pages 425–429, 1993.

[14] Mark S. Fox and Michael Gruninger. On Ontologies and Enterprise Modelling. In *The International Conference on Enterprise Integration Modelling Technology*, 1997.

[15] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., 1987.

[16] Asuncion Gomez-Perez, Mariano Fernandez, and Antonio J. de Vicente. Towards a Method to Conceptualize Domain Ontologies. In *the Workshop on Ontological Engineering, ECAI'96*, pages 41–51, 1996.

[17] Thomas R. Gruber. Ontolingua: A Mechanism to Support Portable Ontologies. Technical report, Knowledge Systems Laboratory, Stanford University, November 1992.

[18] Thomas R. Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Technical report, Knowledge Systems Laboratory, Stanford University, August 1993.

[19] Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. Technical report, Knowledge Systems Laboratory, Stanford University, 1993.

[20] Thomas R. Gruber and Gregory R. Olsen. An Ontology for Engineering Mathematics. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *KR'94: Principles of Knowledge Representation and Reasoning*, pages 258–269. Morgan Kaufmann, San Francisco, California, 1994.

[21] Michael Gruninger and Mark S. Fox. An Activity Ontology for Enterprise Modelling. In *the Third IEEE Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises*, Morgantown, WV, 1994.

[22] Michael Gruninger and Mark S. Fox. The Role of Competency Questions in Enterprise Engineering. In *the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice*, Trondheim, Norway, June 1994.

[23] Michael Gruninger and Mark S. Fox. Methodology for Design and Evaluation of Ontologies. In *the IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.

[24] N. Guarino. The Ontological Level. In R. Casati, B. Smith, and G. White, editors, Philosophy and the Cognitive Sciences. Holder-PichlerTempsky, Vienna, 1994.

[25] Nicola Guarino. Understanding, Building, and Using Ontologies. *International Journal of Human Computer Studies*, 46(2/3), 1997.

[26] R. V. Guha and Douglas B. Lenat. Cyc: A mid-term report. *AI Magazine*, 11(3):32–59, Fall 1990.

[27] Fabio Rinaldi John A. Bateman, Renate Henschel. The generalized upper model 2.0. http://www.darmstadt.gmd.de/publish/komet/gen-um/newUM.html, March 2001.

[28] Douglas B. Lenat. CYC: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[29] Mariano Fernandez Lopez, Asuncion Gomez-Perez, and Juan Pazos Sierra. Building a Chemical Ontology Using Methontology and the Ontology Design Environment. *IEEE Intelligent Systems*, 14(1):37–46, 1999.

[30] John McCarthy. What is artificial intelligence. http://www-formal.stanford.edu/jmc/whatisai/whatisai.html, March 2001.

[31] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Five Papers on Wordnet. Technical report, Cognitive Science Laboratory, Princeton University, 1993.

[32] Knowledge interchange format. http://logic.stanford.edu/kif/dpans.html. draft proposed American National Standard.

[33] Robert Neches, Richard Fikes, Tim Finin, Thomas Gruber, Rameshi Patil, Ted Senator, and William R. Swarton. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56, Fall 1991.

[34] Allen Newell. The Knowledge Level. *AI Magazine*, 18:1–20, 1982.

[35] Natalya Fridman Noy. *Knowledge Representation for Intelligent Information Retrieval in Experimental Sciences*. Ph.d. thesis, College of Computer Science, Northeastern University, Boston, MA, December 1997.

[36] Aris Ouksel. Ontologies are not the panacea in data integration. *Distributed and Parallel Databases*, 7(1), 1999.

[37] John F. Sowa. *Knowledge Representation: Logical Philosophical, and Computational Foundations*. Brooks/Cole, 2000.

[38] Bill Swartout, Ramesh Patil, Kevin Knight, and Tom Russ. Towards distributed use of large-scale ontologies. Technical report, Marina del Rey, California, 1996.

[39] Mike Uschold and Michael Gruninger. Ontologies: Principles, Methods and Applications. *The Knowledge Engineering Review*, 11(2):93–136, 1996.

[40] Patrick Henry Winston. *Artificial Intelligence*. Addison-Wesley Publishing Company, 1992.