Chapter 3

A Logical Foundation for the Semantic Web

In this chapter, we will develop a framework for reasoning about the Semantic Web. We will start with a basic logic approach and gradually refine it to deal with the problems of representing knowledge on the Web.

3.1 An Initial Approach

A requirement for the Semantic Web is the ability to associate explicit meaning with the content of resources. We will do this by embedding a logical language in the resources and providing a denotational semantics for it. Many of the knowledge representation languages and structures discussed in Chapter 2, such as semantic networks, frame systems, and datalog, can all be formulated in first-order logic. For this reason, and because first-order logic is well-understood, we will use it as our basis. In order to use this framework with systems that cannot be described in first-order logic (e.g., probabilistic logics, temporal logics, higher-order logic), one must reformulate what follows to correspond to the desired logic.

First we must define our domain of discourse. The main objects of interest are internet resources and entities that are described by them. An internet resource is anything that provides information via the Internet, such as a web page, newsgroup, or e-mail message. We will use R to refer to the set of these resources. The domain of discourse, on the other hand, is the collection of things that are described or mentioned by internet resources, including potentially internet resources themselves. We will use D to refer to this set.

We will assume that we have a first-order language $\mathcal{L}$ with a set of non-logical symbols S. The predicate symbols of S are $S_P \subset S$, the variable symbols are $S_X \subset S$, and the constant symbols are $S_C \subset S$. For simplicity, we will not discuss function symbols, since an n-ary function symbol can be represented by a n+1-ary predicate. The well-formed formulas of $\mathcal{L}$ are defined in the usual recursive way. We will use W to refer to the infinite set of well-formed formulas that can be constructed in $\mathcal{L}$.

Let $K : R \rightarrow 2^W$ be a function that maps each resource into a set of well-formed formulas. We call K the *knowledge function* because it extracts the knowledge contained in a resource and provides an axiomatization for it.

We will define an interpretation $\mathcal{I}$ in the standard way. It consists of the domain of discourse D (as defined above), a function $\mathcal{I}_C : S_C \rightarrow D$ that maps constant symbols to elements of the domain, and a set of functions $\mathcal{I}_{P_n} : S_P \rightarrow D^n$ that map n-ary predicate symbols to sets of n-tuples formed from the domain. If a formula ϕ is true with respect to an interpretation $\mathcal{I}$, then we write $\mathcal{I} \models \phi$ and say that $\mathcal{I}$ satisfies ϕ or that $\mathcal{I}$ is a model of ϕ. Given a set of sentences Γ, if an interpretation $\mathcal{I}$ satisfies every $\phi \in \Gamma$ then we write $\mathcal{I} \models \Gamma$.

One way to consider the Semantic Web is to think of each resource as specifying an independent theory, that is, there is no interaction between the theories. In this approach, each resource must specify the complete theory that is needed to reason about it. For example, a genealogy web page that contains information on one's ancestors should include the following axioms that provide basic semantics for the $ancestorOf$ predicate:

$$parentOf(x, y) \rightarrow ancestorOf(x, y)$$
$$ancestorOf(x, y) \wedge ancestorOf(y, z) \rightarrow ancestorOf(x, z)$$

However, a disadvantage of the independent theory approach is that all other genealogy pages must replicate the axioms that define the basic genealogy predicates. Furthermore, if one resource contained the fact $ancestorOf(alice, bill)$ and another contained the fact $ancestorOf(bill, carol)$, then we would be unable to conclude $ancestorOf(alice, carol)$, because we cannot combine the theories.

In order to prevent the Semantic Web from becoming a billion unrelated islands, there needs to be a way to combine the information contained in the resources. We will state this as a fundamental principle of the Semantic Web:

Principle 3.1 *The Semantic Web must provide the ability to combine information from multiple resources.*

Given this proposition, let us consider an approach to combining the resources. We define a naive integrated theory NIT as the union of the well-formed formulas generated by the set of resources.

Definition 3.2 *Given a set of resources R, a naive integrated theory is:*

$$NIT(R) = \bigcup_{r \in R} K(r)$$

At first glance, this seems to be a sufficient approach. Since the formulas of all resources are combined, the axiomatization of any domain needs only to be expressed in a single resource, and it is possible to deduce things that require premises from distinct resources. However, upon closer inspection, problems with this approach begin to emerge.

Recall that the Web is a decentralized system and its resources are autonomous. As a result, different content providers are free to assign their own meanings to each nonlogical symbol, thus it is likely that multiple meanings will be assigned to many symbols. Different axiomatizations for the same symbols may result from the polysemy of certain words, poor modeling, or even malicious attempts to break the logic.

To resolve the problem of accidental name conflicts, we will assume that the constants S_C of $\mathcal{L}$ are URIs. Since URIs provide hierarchical namespaces (as described in in Section 2.1.2), they can be used to guarantee that constants created by different parties will be distinct.

Other problems are more complex. As pointed out by Guha [46, Section 2.6], a theory usually has an implied background theory that consists of its assumptions. To combine a set of theories accurately, we need to make some of these assumptions explicit. For example, if one theory about stock prices assumed that the day was yesterday and another assumed that the day was today, an integrated theory should attach the date to each assertion. Guha calls this process *relative decontextualization*. Relative decontextualization may also involve mapping synonymous terms, or in the more complex case, different representational structures that have the same meaning. Note that in order to perform relative decontextualization, one must first know the contexts associated with the two theories. We will distinguish between simple combination, in which no relative decontextualization is performed, and integration, in which it is performed.

Principle 3.3 *Semantic web resources can only be integrated after they have undergone relative decontextualization.*

One way to avoid the need for relative decontextualization is to create a standardized vocabulary with official definitions for each symbol, However, to handle all expressions that might appear on the Web, the vocabulary would have to be enormous, making it nearly impossible to standardize, comprehend, and later change as necessary.

3.2 An Ontology-Based Approach

Recall from Section 2.2.3 that an ontology provides a common vocabulary to support the sharing and reuse of knowledge. When two parties agree to use the same ontology, they agree on the meanings for all terms from that ontology and their information can be combined easily. Unfortunately, there is no widely accepted formal definition of an ontology. In this section and the next two, we will formally define ontologies that are applicable to the Semantic Web.

3.2.1 Ontology Definitions

Let us think of an ontology as simply a set of symbols and a set of formal definitions, along the lines of Farquhar, Fikes, and Rice [30]. We will assume that the formal definitions are written in the language $\mathcal{L}$. We can now define an ontology:

Definition 3.4 *Given a logical language $\mathcal{L}$, an ontology is a tuple $\langle V, A \rangle$, where the vocabulary $V \subset S_P$ is some subset of the predicate symbols of $\mathcal{L}$ and the axioms $A \subset W$ are a subset of the well-formed formulas of $\mathcal{L}$.*

As a result of this definition, an ontology defines a logical language that is a subset of the language $\mathcal{L}$, and defines a core set of axioms for this language. Since the ontology defines a language, we we can talk about well-formed formulas with respect to an ontology.

Definition 3.5 *A formula ϕ is well-formed with respect to an ontology $O = \langle V, A \rangle$, iff ϕ is a well-formed formula of a language $\mathcal{L}'$, where the constant symbols are S_C and the variable symbols are S_X, but the predicate symbols are V.*

We can also define what it means for an ontology to be well-formed.

Definition 3.6 *An ontology $O = \langle V, A \rangle$ is well-formed if every formula in A is well-formed with respect to O.*

We will now provide meaning for ontologies by defining interpretations and models for them. First, we define a pre-interpretation that maps each constant symbol of the language to the domain of discourse.

Definition 3.7 *A pre-interpretation of $\mathcal{L}$ is a structure that consists of a domain D and a function that maps every constant symbol in S_C to a member of D.*

Every $\mathcal{L}$-ontology uses the same pre-interpretation. Since the symbols from S_C are URIs, their intended interpretation is fixed by the URI scheme or by their owners. For this reason, it is assumed that these interpretations are universal. An interpretation of an ontology consists of the pre-interpretation and a mapping of the predicate symbols of $\mathcal{L}$ to relations on the domain.

Definition 3.8 *An interpretation of an ontology consists of a pre-interpretation and a function that maps every n-ary predicate symbol in S_P to an n-ary relation on D.*

We can now define a model of an ontology.

Definition 3.9 *A model of an ontology O=$\langle V, A \rangle$ is an interpretation that satisfies every axiom in A.*

Thus an ontology attempts to describe a set of possibilities by using axioms to limit its models. Some subset of these models are those intended by the ontology, and are called the intended models of the ontology. Note that unlike a first-order logic theory, an ontology can have many intended models because it can be used to describe many different states of affairs.

Note that we chose to have the interpretation of an ontology assign relations to every predicate symbol in the language $\mathcal{L}$, not just those in the ontology. This makes it possible to compare the models of different ontologies that may have separate vocabularies. Since we are treating ontologies as disjoint, this is not significant now. However, it will become important when we begin to discuss ontologies that can extend other ontologies and reuse their vocabulary. Also note that the intended interpretations of an ontology will limit the relations that directly or indirectly correspond to predicates in its vocabulary, while allowing any of the possibilities for predicate symbols in other domains.

3.2.2 Resource Definitions

Now we need to associate an ontology with each resource. If we let $\mathcal{O}$ be the set of ontologies, then we can create a function $C : R \to \mathcal{O}$, which maps resources to ontologies. We call this the *commitment function* because it returns the ontology that a particular resource commits to. When a resource commits to an ontology, it agrees to the meanings ascribed to the symbols by that ontology. Because C is a function, a resource can only commit to a single ontology, but in Section 3.3.3 we will show how a single ontology can combine multiple ontologies, thus overcoming this limitation.

The vocabulary that a resource may use is limited by the ontology to which it commits.

Definition 3.10 *A resource r is well-formed if $C(r) = O$ and $K(r)$ is well-formed with respect to O.*

That is, a resource is well-formed if the theory given by the knowledge function is well-formed with respect to the ontology given by the commitment function.

We now wish to define the semantics of a resource. When a resource commits to an ontology, it has agreed to the terminology and definitions of the ontology. Thus every interpretation of an ontology is an interpretation of the resources that commit to it, and an interpretation that also satisfies the formulas of a resource is a model of that resource.

Definition 3.11 *A model of a resource r, where $C(r) = O$, is a model of O that also satisfies every formula in $K(r)$.*

3.2.3 Simple Ontology Perspectives

Using the definitions above, we could once again create a separate theory for each resource, knowing that the ontologies provide reusable sets of axioms that do not need to be repeated for each resource in the same domain. However, this approach would still prevent us from combining information from different resources, and thus be in conflict with Principle 3.1. Instead, we will consider a way to create larger theories that combine resources which share ontologies. We will attempt to divide the Semantic Web into sets of resources that share a context, and thus can be combined without relative decontextualization. We will call these divisions perspectives, because they provide different views of the Semantic Web.

We need some guidelines for determining how to construct the perspectives. Each perspective will be based on an ontology, hereafter called the basis ontology or base of the perspective. By providing a set of terms and a standard set of axioms, an ontology provides a shared context. Thus, resources that commit to the same ontology have implicitly agreed to share a context. To preserve the semantics intended by the author of each ontology and resource, we will require that the models of each perspective be a subset of the models of its basis ontology and of each resource included in the perspective. Thus, the perspective must contain the axioms of the ontology and the formulas of each resource that commits to it.

Definition 3.12 *Given a set of ontologies $\mathcal{O} = \{O_1, O_2, \ldots, O_n\}$ where $O_i = \langle V_i, A_i \rangle$, a simple ontology perspective based on ontology O_i is:*

$$SOP_i(R) = A_i \cup \bigcup_{\{r \in R | C(r) = O_i\}} K(r)$$

With this approach, we have a separate logical theory for each ontology. Each of these theories includes the axioms of the ontology that serves as its basis and the theories of each resource that commits to that ontology. Since each resource in the perspective agrees to the meanings ascribed to the symbols by the perspective's ontology, there will be no name conflicts between different resources in the perspective. Additionally, since only one ontology is used in each perspective, there is no possibility of axioms from another ontology having unintended side effects.

Although the simple ontology perspective approach solves many problems, it greatly restricts interoperability of resources. The only resources that can be integrated are those that commit to the same ontology. Obviously, this is too restrictive; some content providers may find that an existing ontology would be suitable if only a few minor additions were made. It would be unfortunate if a new, incompatible ontology had to be created for this purpose. Furthermore, if information concerning a domain is provided independently from different sources, then it is likely that the sources will want to use the terminology that is convenient for them. These needs can be summed up in a single principle.

Principle 3.13 *A semantic web ontology should be able to extend other ontologies with new terms and definitions.*

This principle requires that ontologies be able to reference other ontologies, and to provide axioms that relate new terms to the terms in these other ontologies.

3.3 Ontology Extension

Two prominent themes in ontology research are reusability and composability. It is recognized that ontology construction is a difficult and time-consuming task, so the ability to create standard modules for specific purposes is appealing. In theory, if standard ontologies existed for common tasks, much of the ontology design process could occur by assembling a number of existing modules, and simply modeling the unique aspects of the domain as needed. A useful consequence of ontology reuse is that all ontologies that reuse a given module will use the same vocabulary and axioms to model similar concepts. In the previous section, we described how different resources can reuse ontologies, but now we will consider ontologies that reuse other ontologies.

In most existing ontology work, reuse is handled by providing a mechanism that allows an ontology to extend another ontology (see Section 2.2.3). Essentially, when an ontology extends another, it includes the axioms and vocabulary of that ontology and any ontology extended by the second ontology. Using this framework, it is possible to design taxonomies of ontologies, with high-level generic ontologies at the top, and more specific ontologies at the bottom. Thus it is possible to have a top-level ontology that defines common concepts such as *Person* and *Organization*, which is extended by industry specific ontologies, that are in turn extended by corporation specific ontologies, and so on.

Unlike work in schema integration, ontology extension integrates ontologies at schema (ontology) design time. When a new ontology is created, its relationships to other ontologies are specified. This process greatly reduces the semantic heterogeneity of the ontologies, while accommodating differences where necessary. When an existing term is needed, it is simply borrowed from another ontology; when the terms from other ontologies are unsuitable, a new term can be created and axioms can be used to describe its relationship to existing terms. Section 3.5 discusses the practical problems of such an approach.

3.3.1 Ontology Extension Definitions

We can formally define ontology extension as follows:

Definition 3.14 *Given ontologies O_1 and O_2, O_1 is said to extend O_2 iff all models of O_1 are also models of O_2.*

Note that this definition depends on the ability to compare the models of two different ontologies. Recall from Section 3.2.1 that the predicate interpretation function of an ontology is defined for all predicate symbols of $\mathcal{L}$, not just those in the ontology's vocabulary. Thus since every interpretation of every ontology has a set of tuples associated with each predicate symbols, it is possible to compare the tuples for each predicate symbol and determine whether one interpretation is a subset of another.

Now let us add the concept of ontology extension to our formalism. We will refine our definition of an ontology to include the set of ontologies extended by it. Definition 3.4 can be thought of as a special case of this definition, where $O = \langle V, A, \emptyset \rangle$.

Definition 3.15 *Given a logic $\mathcal{L}$, an ontology is a three-tuple $\langle V, A, E \rangle$, where the vocabulary $V \subset S_P$ is some subset of the predicate symbols, the axioms $A \subset W$ are a subset of the well-formed formulas, and $E \subset \mathcal{O}$ is the set of ontologies extended by O.*

This new ontology definition requires us to reconsider the definitions from Section 3.2. Many of the definitions are unchanged, but well-formedness with respect to an ontology, the well-formedness of ontologies, and the models of an ontology need to be redefined. However, before we can discuss the new definitions, we need to consider the uniqueness of the vocabulary symbols used by ontologies and define the concept of ancestor ontologies.

When different ontologies are used, they may assign different meanings to the same symbol. In the previous section, we ignored this fact because each ontology was used to form a separate theory. However, when ontologies include other ontologies, reasoners that assume that a symbol means the same thing when used in different contexts risk incorrect inferences. Therefore we will assume that unless otherwise stated, identical symbols in different ontologies represent distinct concepts, as advocated by Wiederhold [92]. In our framework, we will achieve this by prefixing each predicate symbol with its source ontology and a colon, for example *ont*:*symbol*. When we refer to symbols, we will use either qualified names or or unqualified names, where a qualified name includes the prefix, while an unqualified name does not. To prevent ambiguity, all unqualified names have exactly one corresponding qualified name, which is the name formed by adding the prefix of the ontology in which the name appears, or in the case of resources, the ontology committed to by the resource. The function $Q : name \rightarrow qname$ performs this mapping, where $qname \subset name \subset S_P$ and $\forall x, x \in qname \leftrightarrow Q(x) = x$. For convenience, we will write the set of names resulting from applying Q to each member of a set N as $Q(N)$.

An ancestor of an ontology is an ontology extended either directly or indirectly by it. If O_2 is an ancestor of O_1, we write $O_2 \in anc(O_1)$. In this case, we may also say that O_1 is a descendant of O_2. The formal definition of an ancestor is:

Definition 3.16 *Given ontologies $O_1 = \langle V_1, A_1, E_1 \rangle$ and $O_2 = \langle V_2, A_2, E_2 \rangle$, $O_2 \in anc(O_1)$ iff $O_2 \in E_1$ or there exists an $O_i = \langle V_i, A_i, E_i \rangle$ such that $O_i \in E_1$ and $O_2 \in anc(O_i)$.*

An ontology should have access to all symbols defined in its ancestors, and likewise a formula of that ontology should still be well-formed if it uses symbols from the ancestor ontologies. Thus, we need to redefine what it means to be well-formed with respect to an ontology. First, we must identify the vocabulary accessible to an ontology. This is the union of its own vocabulary and that of all of its ancestors. The vocabulary of an ontology is a set of unqualified names, but the extended vocabulary can be a mixture of qualified and unqualified names. This is because we must use the qualified names of the ancestor ontologies to guarantee that there are no name conflicts.

Definition 3.17 *The extended vocabulary V_i^* of an ontology $O_i = \langle V_i, A_i, E_i \rangle$ is $V_i \cup \bigcup_{\{j|O_j \in anc(O)\}} Q(V_j)$.*

Now we modify Definition 3.5 to say that a formula is well-formed with respect to an ontology if it is well-formed with respect to a language where the predicate symbols are given by the extended vocabulary of the ontology.

Definition 3.18 *A formula ϕ is well-formed with respect to an ontology $O = \langle V, A, E \rangle$ iff ϕ is a well-formed formula of a language $\mathcal{L}'$, where the constant symbols are S_C and the variable symbols are S_X, but the predicate symbols are V^*.*

We also need to modify the definition of a well-formed ontology. In addition to using the new definition of well-formedness with respect to an ontology, we must consider aspects of the extended ontologies. Besides requiring that they be well-formed, we also must prevent cycles of ontology extension.

Definition 3.19 *An ontology $O = \langle V, A, E \rangle$ is well-formed iff A is well-formed with respect to O, all ancestors of O are well-formed, and O is not an ancestor of O.*

Finally, let us redefine a model of an ontology. In particular, all models of an ontology should also be models of every ontology extended by it.

Definition 3.20 *Given an ontology $O = \langle V, A, E \rangle$, if $E = \emptyset$ then a model of O is an interpretation that satisfies every formula in A, otherwise a model of O is a model of every ontology in E that also satisfies every formula in A.*

3.3.2 Example of Ontology Extension

In Figure 3.1, we demonstrate how ontology extension can be used to relate the vocabularies of different domains, thus promoting interoperability. When two ontologies need to refer to a common concept, they should both extend an ontology in which that concept is defined. In this way, consistent definitions can be assigned to each concept, while still allowing communities to customize ontologies to include definitions and rules of their own for specialized areas of knowledge.

The problems of synonymy and polysemy can be handled by the extension mechanism and use of axioms. An axiom of the form $P_1(x_1, \ldots, x_n) \leftrightarrow P_2(x_1, \ldots, x_n)$ can be used to state that two predicates are equivalent. With this idiom, ontologies can create aliases for terms, so that domain-specific vocabularies can be used. For example, in Figure 3.1, the term *DeptHead* in O_{U2} means the same thing as *Chair* in O_U due to an axiom in O_{U2}. Although this solves the problem of synonymy of terms, the same terms can still be used with different meanings in different ontologies. This is not undesirable, a term should not be restricted for use in one domain simply because it was first used in a particular ontology. As shown in the figure, different ontologies may also use the same term to define a different concept. Here, the term *Chair* means different things in O_U and O_F because different axioms are used to define it.

Figure 3.1 is easier to understand when shown graphically as in Figure 3.2. In this figure, we have assigned meaningful names to each ontology and used arcs to indicate two common types of axioms: *renames* is used for axioms that state two predicates are equivalent and *isa* is used for axioms of the form $C(x) \rightarrow P(x)$, to go along with the intuition that this means that all members of a class C are also members of a class P. We will say more on the usage of idioms in a semantic web language in Chapter 4.

$$\begin{aligned}
O_G = \quad & \langle\{Thing, Person, Object\}, \\
& \{Person(x) \rightarrow Thing(x), \\
& Object(x) \rightarrow Thing(x)\}, \\
& \emptyset\rangle \\
O_U = \quad & \langle\{Chair\}, \\
& \{Chair(x) \rightarrow O_G : Person(x)\}, \\
& \{O_G\}\rangle \\
O_F = \quad & \langle\{Chair\}, \\
& \{Chair(x) \rightarrow O_G : Object(x)\}, \\
& \{O_G\}\rangle \\
O_{U2} = \quad & \langle\{DeptHead\}, \\
& \{DeptHead(x) \leftrightarrow O_U : Chair(x)\}, \\
& \{O_U\}\rangle \\
O_{F2} = \quad & \langle\{Seat\}, \\
& \{Seat(x) \leftrightarrow O_F : Chair(x)\}, \\
& \{O_F\}\rangle
\end{aligned}$$

Figure 3.1: Example of ontology extension.

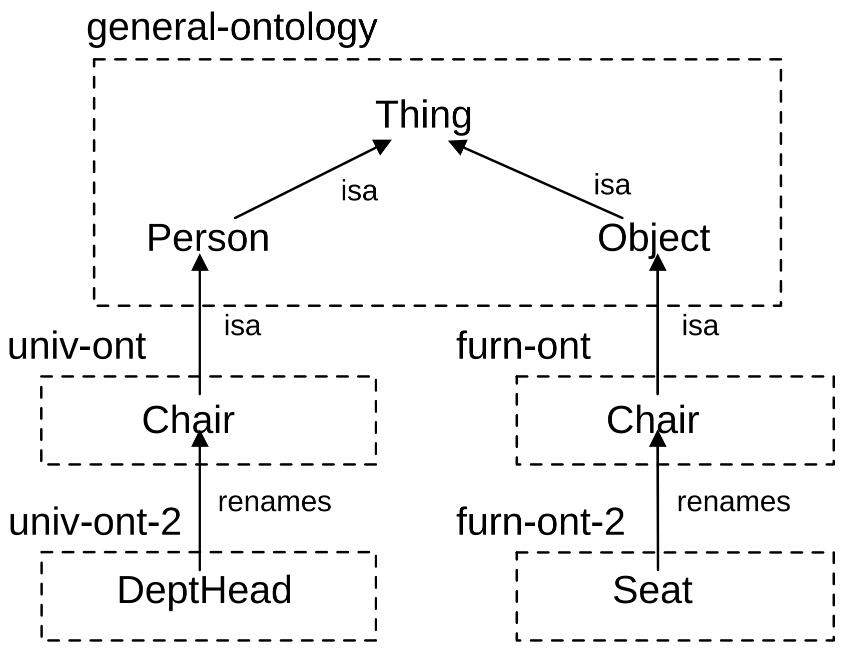

Figure 3.2: Graphical depiction of Figure 3.1.

3.3.3 Extended Ontology Perspectives

If we include ontology extension in our semantic web language, then how does that affect interoperability? Often, traditional ontology work has assumed that reuse was a mechanism to ease the construction of a single unified ontology. It had not considered that a number of related ontologies might be used to structure different data sets. For example, if two resources commit to different ontologies, where one ontology is an ancestor of the other, then it should be possible to integrate data from these sources.

However, since different ontologies can be provided by different sources, it is important that new ontologies do not automatically require a reinterpretation of existing data. Thus an extending ontology should provide the ability to reason about existing resources in new ways, but should not supersede the ontology that it extends. Otherwise, accidental or malicious ontologies could have serious side effects on existing portions of the Web. This point is important enough to deserve a principle.

Principle 3.21 *Each ontology should provide a different perspective on a set of resources, and no ontology should change the perspective of another.*

Given this principle and our new definition of an ontology, how can we define a perspective that maximizes integration? We will assume that an ontology which includes another does not attempt to change the intended meaning of the ontology and will contain any axioms necessary for decontextualization with respect to it. Thus, we can refine our ontology perspectives from Section 3.2 to include resources that commit to any ontologies that are ancestors of the perspective's basis ontology. We wish for our perspectives to be the intersection of the models of the ontology and all included resources. In the case of an ontology, its models are determined by its axioms and those of its ancestors, while the models of a resource are determined by its knowledge function and the ontology to which it commits. Thus, a new kind of perspective an be defined as follows:

Definition 3.22 *Given a set of ontologies $\mathcal{O} = \{O_1, O_2, \ldots, O_n\}$ where $O_i = \langle V_i, A_i, E_i \rangle$, then an extended ontology perspective based on ontology O_i is:*

$$EOP_i(R) = A_i \cup \bigcup_{\{j|O_j \in anc(O_i)\}} A_j \cup \bigcup_{\{r \in R|C(r)=O_i \,\vee\, C(r) \in anc(O_i)\}} K(r)$$

With extended ontology perspectives, there is a separate theory for each ontology, but some theories may have overlapping axioms and ground atoms, depending on how the basis ontologies are related. A perspective contains the axioms of its basis ontology, the axioms of all of its ancestor ontologies, and the formulas from all resources that commit to the defining ontology or one of its ancestors. Thus, two perspectives that are based on ontologies with a common ancestor will have that ancestor's axioms in common, and will have the formulas of all resources that commit to that ancestor in common as well.

A desirable property of these perspectives is that a perspective based on an ontology entails all of the sentences that are entailed by any perspectives based on ancestors of the ontologies. This ensures that any conclusions sanctioned by a resource are valid conclusions in any perspective that includes the resource. We will show that extended ontology perspectives satisfy this property.

Theorem 3.23 *Given two ontologies O_1 and O_2 such that $O_2 \in anc(O_1)$, if $EOP_{O_2}(R) \models \phi$, then $EOP_{O_1}(R) \models \phi$.*

Proof Let $O_1 = \langle V_1, A_1, E_1 \rangle$ and $O_2 = \langle V_2, A_2, E_2 \rangle$. We will prove the theorem by showing that $EOP_{O_1}(R)$ is a superset of each of the parts of $EOP_{O_2}(R)$. First, because $O_2 \in anc(O_1)$, $A_2 \in EOP_{O_1}(R)$. Second, due to Definition 3.16, $anc(O_1) \supset anc(O_2)$, thus the A_i of all ontologies $O_i \in anc(O_2)$ are also $\in EOP_{O_1}(R)$. Finally, since $O_2 \in anc(O_1)$ and $anc(O_1) \supset anc(O_2)$, all $K(r)$ such that $C(r) = O_2$ or $C(r) \in anc(O_2)$ are in $EOP_{O_1}(R)$. Therefore, $EOP_{O_1}(R) \supseteq EOP_{O_2}(R)$. Since FOL is monotonic, if $EOP_{O_2}(R) \models \phi$, then $EOP_{O_1}(R) \models \phi$.

As it turns out, simple ontology perspectives are a special case of extended ontology perspectives. When no relevant ontologies extend any other ontologies, the two functions are equivalent.

Theorem 3.24 *Given a set of ontologies $\mathcal{O} = \{O_1, O_2, \ldots, O_n\}$, where $\forall O_i \in \mathcal{O}, O_i = \langle V_i, A_i, \emptyset \rangle$, then $EOP_i(R) = SOP_i(R)$.*

Proof If we substitute $\emptyset$ for E_i in Definition 3.22, then $anc(O_i) = \emptyset$ and the set of j such that $O_j \in anc(O_i)$ is empty. Therefore, the corresponding union is $\emptyset$. Additionally, since C is a total function, the set of r such that $C(r) \in anc(O_i)$ is empty, which reduces the union condition to $\{r \in R | C(r) = O_i\}$. Thus the definition reduces to:

$$\begin{aligned}
EOP_i(R) &= A_i \cup \bigcup_{\{j | O_j \in anc(O_i)\}} A_j \cup \bigcup_{\{r \in R | C(r) = O_i \vee C(r) \in anc(O_i)\}} K(r) \\
&= A_i \cup \emptyset \cup \bigcup_{\{r \in R | C(r) = O_i\}} K(r) \\
&= A_i \cup \bigcup_{\{r \in R | C(r) = O_i\}} K(r) \\
&= OPT_i(R)
\end{aligned}$$

Extension is useful in overcoming one of the limitations imposed by the commitment function. This function only allows each resource to commit to a single ontology, however with extension, a virtual ontology can be created that represents multiple ontologies committed to by a single resource. For example, given two ontologies $O_1 = \langle V_1, A_1, E_1 \rangle$ and $O_2 = \langle V_2, A_2, E_2 \rangle$, the ontology $O_{union} = \langle \emptyset, \emptyset, \{O_1, O_2\} \rangle$ is equivalent to their union. A resource that needed to commit to O_1 and O_2 could instead commit to O_{union}.

3.4 Ontology Evolution

The Web is a dynamic place, where anyone can instantaneously publish and update information. It is important that this ability is not lost when we provide more structure for the information. People must be able to publish semantic web ontologies as easily as other documents, and they must be allowed to revise these ontologies as well. While good design may prevent many ontological errors, some errors will not be realized until the ontology is put to use. Furthermore, pressing information needs may limit the time that can be applied to design particular ontologies, resulting in the need to improve the ontologies later. More philosophical arguments concerning the need for ontology revision are made by Foo [34].

Most ontology systems do not manage the problem of ontology change. Often this is because these systems are prototypes used for research purposes, and thus any dependencies are insignificant. For centralized systems, a change to the ontology can be synchronized with the corresponding changes to any dependent information, making change management unnecessary.

The developers of Ontolingua, a language used in the distributed development of ontologies, made the decision to ignore prior versions of an ontology. Farquhar, Fikes, and Rice [30] state that the inclusion feature in Ontolingua does not result in a "cut and paste" of the contents because "this interpretation would result in unfortunate version dependencies." However, this ignores the problem that the included ontology could change in a way that would make all of its including ontologies invalid.

One area where the problem of ontology change has been examined are medical terminology systems. Medical terminologies often consist of hierarchies of concepts, and sometimes include synonyms and properties. A number of different systems are used for different purposes, and the terminologies are frequently merged or mapped to each other, so that information from different systems can be combined. However, due to a number of factors, such as new medical knowledge and corrections of errors, the individual terminologies will continue to evolve. Since these terminologies are used in real systems, management of ontology change is a critical issue. Oliver et al. [77] discuss the kinds of changes that occur in medical ontologies and propose the CONCORDIA concept model to cope with these changes. The main aspects of CONCORDIA are that all concepts have a permanent unique identifier, concepts are given a *retired* status instead of being physically deleted, and special links are maintained to track the retired parents and children of each concept. However, this approach is insufficient for managing change on the Semantic Web. In the next sections, we will discuss the kinds of changes that might occur, and present a revised ontology definition that can describe these changes.

3.4.1 Ontology Evolution Examples

When we decide to change an ontology, then we must consider that in a distributed ontology framework such as the one needed by the Semantic Web, there will often be dependencies on it. We will illustrate the issues with examples of ontology change within our framework. In Figure 3.3, we demonstrate what happens when a new term is added to the ontology. In the example, O_U, r_1, and r_2 represent a simple university ontology and two resources that commit to it. Recall that an ontology is three-tuple $\langle V, A, E \rangle$ where V is its vocabulary, A is its set of axioms, and E is the set of ontologies extended by it. Also recall that K is the knowledge function that maps resources to formulas. Thus, O_U consists of a single term *Faculty*, while r_1 and r_2 are resources that use the *Faculty* predicate. At some later point in time, O'_U, r'_1, r'_2, and r'_3 represent the state of relevant web objects. Here, the ontology O'_U represents a new version O_U which includes terms that represent subclasses of *Faculty*. When an ontology designer adds terms in this way, it is likely that he will add axioms, such as $Professor(x) \rightarrow Faculty(x)$ to help define the terms. Note that r'_1 and r'_2 are r_1 and r_2, respectively at the later point in time. Because $K(r'_1) = K(r_1)$ and $K(r'_2) = K(r_2)$, these resources have not changed. Since the vocabulary V' of O'_U is a superset of V, r_1 and r_2 are still well-formed with respect O'_U. Once O_U has been revised to O'_U, we can create resources that use the new terms in O'_U; r'_3 is an example of such a resource that contains an assertion about *drjones*.

$$\begin{array}{ll}
O_U = & \langle\{Faculty\}, \\
 & \emptyset, \\
 & \emptyset\rangle \\
K(r_1) = & \{Faculty(drdoe)\} \\
K(r_2) = & \{Faculty(drsmith)\} \\
 & \\
O'_U = & \langle\{Faculty, AssistProf, AssocProf, Professor\}, \\
 & \{AssistProf(x) \rightarrow Faculty(x), \\
 & AssocProf(x) \rightarrow Faculty(x), \\
 & Professor(x) \rightarrow Faculty(x)\} \\
 & \emptyset\rangle \\
K(r'_1) = & \{Faculty(drdoe)\} \\
K(r'_2) = & \{Faculty(drsmith)\} \\
K(r'_3) = & \{AssocProf(drjones)\}
\end{array}$$

Figure 3.3: Adding terms to an ontology.

$$\begin{array}{ll}
O_M = & \langle\{Cd, Tape\}, \\
 & \emptyset, \\
 & \emptyset\rangle \\
K(r_1) = & \{Cd(whiteAlbum)\} \\
K(r_2) = & \{Tape(darkSide)\} \\
 & \\
O'_M = & \langle\{Cd\}, \\
 & \emptyset, \\
 & \emptyset\rangle \\
K(r'_1) = & \{Cd(whiteAlbum)\} \\
K(r'_2) = & \{Tape(darkSide)\}
\end{array}$$

Figure 3.4: Deleting a term from an ontology.

All of the resources can be integrated with an extended ontology perspective. For example, if $R = \{r'_1, r'_2, r'_3\}$, then in $EOP_{O'_U}(R)$, *Faculty(drdoe)*, *Faculty(drsmith)*, and *Faculty(drjones)* are all true. Since we are assuming a monotonic logic, if we add terms and axioms to an ontology, then we know that the logical consequences of a perspective based on it will either be unchanged or increased.

However, if a term is deleted from the ontology then existing resources may become ill-formed. An example of this is presented in Figure 3.4. Here, we have a simple music store ontology that defines the classes *Cd* and *Tape*. Resource r_1 makes an assertion about an instance that is a *Cd*, while resource r_2 makes an assertion about an instance that is a *Tape*. Assume that at some point in the future, tapes become obsolete, and the decision is made to remove *Tape* from the ontology. If resource r_2 is not changed, then it becomes ill-formed because the ontology it commits to no longer includes the predicate used in its assertion. Since the resource may be not be owned by the ontology designer (for example, if it is a specific record store that is reusing the ontology), it is impossible to ensure that it will be updated when the ontology changes.

$$\begin{array}{ll}
O_U = & \langle\{Class\}, \\
 & \emptyset, \\
 & \emptyset\rangle \\
K(r_1) = & \{Class(ai)\} \\
K(r_2) = & \{Class(databases)\} \\
 & \\
O'_U = & \langle\{Class, Course\}, \\
 & \emptyset, \\
 & \emptyset\rangle \\
K(r'_1) = & \{Class(ai)\} \\
K(r'_2) = & \{Class(databases)\} \\
K(r'_3) = & \{Class(algFall2001)\} \\
K(r'_4) = & \{Course(algorithms)\}
\end{array}$$

Figure 3.5: Changing the meaning of a term from an ontology.

This leads us to another principle:

Principle 3.25 *The revision of an ontology should not change the well-formedness of resources that commit to an earlier version of the ontology.*

In practice, strict deletion will probably occur rarely. It is more likely that a term will be removed because it can be merged with another term, or a different name is preferred. If the meaning of the term is changed, then a significant problem can arise. For example, consider Figure 3.5, where *Class* used to mean “a subject matter of instruction,” but was changed to mean “a particular offering of a course,” so that *Course* could be used for the old meaning. In this case, old resources such as r'_1 and r'_2 that used the term would be using it incorrectly in the context of the new ontology. However, since they would still be well-formed, there is no way to automatically detect the problem. As a result, false conclusions may be drawn from the information.

One possible solution to the ontology evolution problem is to require that revisions of ontologies be distinct ontologies in themselves. Then each resource can commit to a particular version of an ontology. For example, if in Figure 3.4, $C(r'_1) = O_U$ and $C(r'_2) = O_U$, then both resources commit to the version of the ontology that still has the term *Tape*. Since the ontology committed to by the resources does not physically change, they cannot become ill-formed unless the ontology changes.

Although treating each ontology version as a separate ontology solves the problems with deleting terms, it creates problems for integrating data from resources that commit to different versions of the ontology. Consider the example in Figure 3.3. Here, $C(r'_1) = O_U$, $C(r'_2) = O_U$ and $C(r'_3) = O'_U$. Because O_U and O'_U are different ontologies, and neither extends the other, then any resources that commit to them would be partitioned in separate theories. That is, the vocabularies are treated as distinct even though in fact they are just different formalizations of the same concept. We will formulate the need to integrate resources that commit to different versions of an ontology as a principle.

Principle 3.26 *Resources that commit to a revised ontology can be integrated with resources that commit to compatible prior versions of the ontology.*

In this principle, we need to define what is meant by “compatible.” In cases such as the one examined in Figure 3.3, the newer version of an ontology is usually a better formalization of the domain than a previous version, (i.e., it is a closer approximation of the intended models). Thus, it would be useful if we could use the new perspective to reason about resources that committed to the older version. However, to do this, we need some way to indicate when the ontology revision is simply a refinement of the original ontology. In the next section, we augment our definition of ontology for this purpose.

3.4.2 Ontology Revision Definitions

We introduce the notion of backwards-compatibility to describe revisions that include all terms defined in the previous version and have the same intended meanings for them, although the axiomatizations may be different. This indicates that reasoners can safely assume that descriptions that commit to the old version also commit to the revision.

Definition 3.27 *An ontology O_2 is backwards-compatible with an ontology O_1 iff every intended model of O_1 is an intended model of O_2 and $V_1 \subseteq V_2$.*

Since the definition of backwards-compatible depends on knowledge of the intended models of an ontology, it cannot be computed automatically, instead it must be specified by an ontology’s author. This is driven by the fact that ontologies only specify a theory partially, and that the intended meaning of a term may change even though the ontology’s theory remains the same. Since the ontology can only restrict unintended models, there is no way to formally describe the intended models of an ontology. For example, if an ontology with a rather sparse axiomatization changed the term *Chair* to mean something you sit on as opposed to the head of a department, then if no relations or rules needed to be changed, any reasoning agent would be unaware that the term means different things in different versions. Thus backwards-compatibility must be indicated in an ontology definition. However, syntactic compatibility, such as whether $V_1 \subseteq V_2$, can be checked automatically, and when backward compatibility is specified, syntactic compatibility should be verified.

We will refine Definition 3.15 to include the concepts of an ontology revising another ontology and for an ontology to be backwards-compatible with older versions.

Definition 3.28 *Given a logic $\mathcal{L}$, an ontology is a five-tuple $\langle V, A, E, P, B \rangle$, where the vocabulary $V \subset S_P$ is some subset of the predicate symbols, the axioms $A \subset W$ are a subset of the well-formed formulas, $E \subset \mathcal{O}$ is the set of ontologies extended by O, $P \subset \mathcal{O}$ is the set of prior versions of the ontology, and $B \subset P$ is the set of ontologies that O is backwards compatible with.*

Definition 3.15 is a special case of this definition, where $P = \emptyset$ and $B = \emptyset$. All of the definitions from Section 3.3 still hold, although the five-tuple structure should be substituted for the three-tuple one where necessary.

We will also name two special cases of ontologies.

Definition 3.29 *A top-level ontology is an ontology $O = \langle V, A, E, P, B \rangle$, where $E = \emptyset$.*

Definition 3.30 *A basic ontology is an ontology $O = \langle V, A, E, P, B \rangle$, where $E = P = B = \emptyset$.*

Thus top-level ontologies are ontologies that have no ancestors; they are at the top of the ontology hierarchy. Every ontology must have at least one top-level ontology as an ancestor. Basic ontologies are top-level ontologies that have no prior versions.

Note that backwards-compatibility does not require that the revision contains a superset of the axioms specified by the original version. This allows axioms to be moved to a more general included ontology if needed.

3.4.3 Compatible Ontology Perspectives

Given the new definition of ontology, we can define a method of integration that incorporates backward-compatibility.

Definition 3.31 *Given a set of ontologies $\mathcal{O} = \{O_1, O_2, \ldots, O_n\}$ where $O_i = \langle V_i, A_i, E_i, P_i, B_i \rangle$, then a compatible ontology perspective based on ontology O_i is:*

$$\begin{aligned} COP_i(R) \;=\; & A_i \cup \bigcup_{\{j \mid O_j \in anc(O_i)\}} A_j \cup \bigcup_{\{r \in R \mid C(r) = O_i \,\vee\, C(r) \in anc(O_i)\}} K(r) \\ & \cup \bigcup_{\{r \in R \mid C(r) \in B_i\}} K(r) \cup \bigcup_{\{r \in R \mid \exists j, O_j \in anc(O_i) \,\wedge\, C(r) \in B_j\}} K(r) \end{aligned}$$

Like extended ontology perspectives, this method creates perspectives based upon different ontologies. Each perspective contains the axioms of its basis ontology, the axioms of its ancestors, and the assertions of all resources that commit to the basis ontology or one of its ancestors. However, these perspectives also include the assertions of resources that commit to any ontologies with which the basis ontology is backwards-compatible, and those of any resources that commit to ontologies that the base's ancestor ontologies are backwards-compatible with.

It should be mentioned that this method does not ensure that the perspective is logically consistent. The word compatible is used here in the sense of backward-compatibility, as defined in Section 3.4.2. The problem of inconsistency is discussed in Section 3.6.

As with extended ontology perspectives, a desirable property of compatible ontology perspectives is that a perspective based on an ontology entails all of the sentences that are entailed by any perspectives based on ancestors of the ontologies. We will show that compatible ontology perspectives satisfy this property.

Theorem 3.32 *Given two ontologies O_1 and O_2 such that $O_2 \in anc(O_1)$, if $COP_{O_2}(R) \models \phi$, then $COP_{O_1}(R) \models \phi$.*

Proof Let $O_1 = \langle V_1, A_1, E_1 \rangle$ and $O_2 = \langle V_2, A_2, E_2 \rangle$. We will prove the theorem by showing that $COP_{O_1}(R)$ is a superset of each of the parts of $COP_{O_2}(R)$. Since compatible ontology perspectives build on extended ontology perspectives, the proofs for the first three sets are identical. Since $O_2 \in anc(O_1)$, then every r such that $C(r) \in B_2$ is also in the fifth set of COP_{O_1}. Finally, since $anc(O_1) \supset anc(O_2)$, then for each r such that $O_j \in anc(O_2) \wedge C(r) \in B_j$, then also $O_j \in anc(O_1) \wedge C(r) \in B_j$. Therefore, the fifth set of COP_{O_1} subsumes that of COP_{O_2}. Since all sets that form COP_{O_2} are subsets of the sets that form COP_{O_1}, $COP_{O_1}(R) \supseteq COP_{O_2}(R)$. Since FOL is monotonic, if $COP_{O_2}(R) \models \phi$, then $COP_{O_1}(R) \models \phi$.

Also, if no ontologies revise any other ontologies, then compatible ontology perspectives are equivalent to extended ontology perspectives.

Theorem 3.33 *Given a set of ontologies* $\mathcal{O} = \{O_1, O_2, \ldots, O_n\}$*, where* $\forall O_i \in \mathcal{O}, O_i = \langle V_i, A_i, E_i, \emptyset, \emptyset\rangle$*, then* $COP_i(R) = EOP_i(R)$*.*

Proof If we substitute $\emptyset$ for B_i in Definition 3.31, then the set of $r \in R$ such that $C(r) \in B_i$ is empty because there are no r such that $C(r) \in \emptyset$. Therefore, the corresponding union is $\emptyset$. Likewise, the set of $r \in R$ such that $O_j \in anc(O_i) \wedge C(r) \in B_j$ must be empty, and the corresponding union is $\emptyset$. Thus the definition reduces to:

$$\begin{aligned}
COP_i(R) &= A_i \cup \bigcup_{\{j\,|\,O_j \in anc(O_i)\}} A_j \cup \bigcup_{\{r \in R | C(r) = O_i \,\vee\, C(r) \in anc(O_i)\}} K(r) \cup \emptyset \cup \emptyset \\
&= A_i \cup \bigcup_{\{j\,|\,O_j \in anc(O_i)\}} A_j \cup \bigcup_{\{r \in R | C(r) = O_i \,\vee\, C(r) \in anc(O_i)\}} K(r) \\
&= EOP_i(R)
\end{aligned}$$

Technically, the Semantic Web should not allow ontologies to arbitrarily revise other ontologies. Unlike, ontology extension, revision implies that a change has been authorized by the ontology's owner. Potential mechanisms for ensuring this include requiring older versions to point to their revisions, requiring revisions to be placed in the same directory of the same server as the ontology being revised, or to be signed by the same entity.

3.5 Ontology Divergence

As discussed earlier, an important aspect of this framework is that interoperability is achieved through ontology reuse. That is, the preferred method of ontology development is to extend existing ontologies and create new definitions only when existing definitions are unsuitable. In this way, all concepts are automatically integrated. However, when there is concurrent development of ontologies in a large, distributed environment such as the Web, it is inevitable that new concepts will be defined when existing ones could be used. Even when ontology authors have the best intentions, they may be unaware of similar efforts to describe the same domain, and their ontologies may be widely used by the time the problem is noticed. As a result there will be a tendency for the most specific ontologies to diverge and become less interoperable. In these situations, occasional manual integration of ontologies is needed.

This section discusses the types of semantic heterogenity that may occur in ontologies and presents a method for resolving ontology divergence within the framework presented earlier in this chapter. The ideas described here are a refinement of those presented in an earlier paper [52].

3.5.1 Domain Differences

The divergence of ontologies increases the semantic heterogeneity (see Section 2.4) of the Semantic Web. However, the use of first-order logic as our model results in a more restricted set of possible differences than those typically described by work in database schema integration. Wiederhold [91] describes four types of domain differences, which we paraphrase here:

context: a term in one domain has a completely different meaning in another

terminology: different names are used for the same concepts

scope: similar categories may not match exactly; their extensions intersect, but each may have instances that cannot be classified under the other

encoding: the valid values for a property can be different, even different scales could be used

Each of these differences can be resolved within our semantic web framework. Context differences are due to polysemous terms, and are handled by treating terms in each ontology as distinct. The other differences require the use of articulation axioms [19, 28], which are similar in purpose to lifting rules [46, Section 3.2]. An articulation axiom is simply an axiom that describes how to relate terms from two different ontologies. We will now demonstrate how to resolve the domain differences described above using axioms.

Terminological differences are synonyms, and as such can be handled using the equivalence idiom described in Section 3.3.2. For example, if it was determined that *Employee* in O_{kmart} meant the same thing as *StaffMember* in $O_{walmart}$, then the articulation axiom would be:

$$O_{kmart} : Employee(x) \leftrightarrow O_{walmart} : StaffMember(x)$$

Scope differences require mapping a category to the most specific category in the other domain that subsumes it. Thus, if we knew that every *FighterPilot* in O_{af} is a *JetPilot* in O_{faa}, then we would create the articulation axiom:

$$O_{af} : FighterPilot(x) \rightarrow O_{faa} : JetPilot(x)$$

Encoding difference are somewhat trickier. The problem is that different sets of values are used to describe the same data. These sets may have different cardinalities or may be infinite. An example of value sets with different cardinalities may be two rating schemes for movies. One scheme uses {*Poor*,*Fair*,*Excellent*} while the other uses integers 1-5. In this case, individual values could be mapped as in:

$$O_{siskel} : Rating(x, Excellent) \leftrightarrow O_{ebert} : Rating(x, 5)$$

Other differences may be due to different units, such as meters versus feet. Articulation axioms to resolve these sorts of encodings would require the use of arithmetic functions, as in:

$$O_{english} : Foot(x, l) \rightarrow O_{metric} : Meter(x, *(l, 0.3048))$$

Note that arithmetic functions are simply functions whose domains range over integers or real numbers, and thus do not require any special treatment in first-order theory. However, such functions can be problematic in reasoning algorithm implementation. For example, unit conversion may introduce inaccuracies due to floating point arithmetic and rounding. This can get compounded if ontologies have rules for translating both ways. For example, if a reasoner translated 3 feet to 0.914 meters, it better not then apply the opposite rule and get a length of 2.999 feet as well. Such a process could go on ad infinitum. An even more difficult encoding difference is due to different textual representations. Consider "Smith, John" versus "John Smith." An articulation axiom to establish name correspondences in general would require a function that can take the last-name-first form and convert it to the first-name-first form. Although this is easy in theory, in practice it requires a large list of pre-defined functions or a complex language for defining functions.

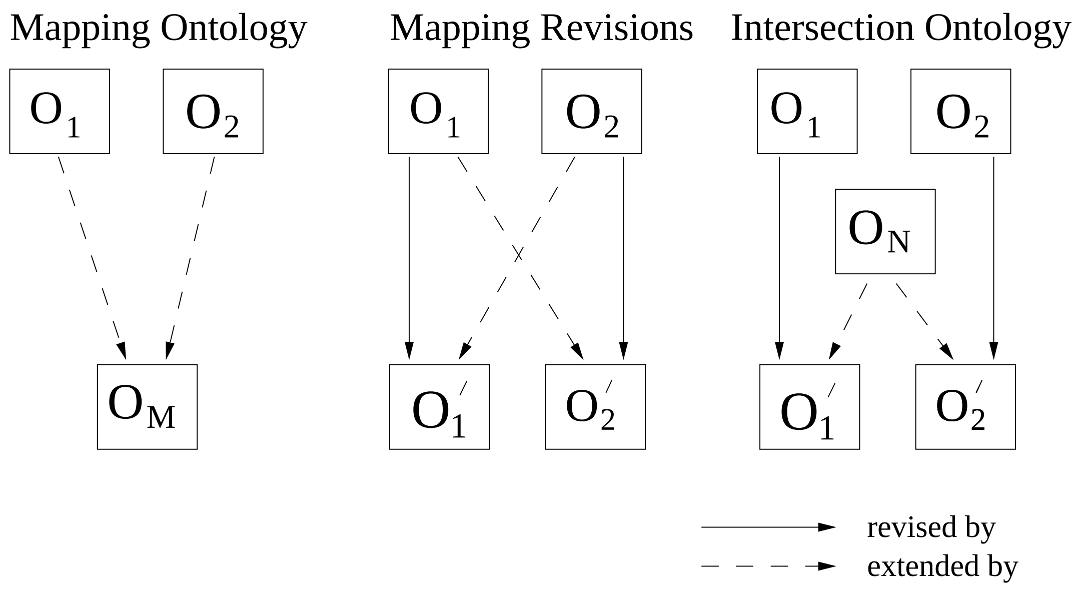

Figure 3.6: Methods for resolving ontology divergence.

3.5.2 Resolving Ontology Divergence

Ontology integration typically involves identifying the correspondences between two ontologies, determining the differences in definitions, and creating a new ontology that resolves these differences. The process for aligning ontologies can be performed either manually or semi-automatically. Chimaera [69] and PROMPT [75] are examples of tools that help users to align ontologies. However, it is important to note that simply creating a new integrated ontology does not solve the problem of integrating information on the Web. When the web community has synthesized the ontologies (that is, other web pages and ontologies come to depend on them), all of the dependent objects would have to be revised to reflect the new ontology. Since this would be an impossible task, we instead suggest three ways to incorporate the results of an ontology integration effort, each of which is shown in Figure 3.6. In this figure, we assume that O_1 and O_2 are two ontologies which have some domain overlap and need to be integrated.

In the first approach, we create a third ontology, called a mapping or articulation ontology, that can translate the terminologies of the two ontologies. In the example, the mapping ontology is O_M. In order to map the terminologies, O_M must extend both O_1 and O_2, and provide a set of articulation axioms T as described above. Note that O_M does not add any vocabulary terms, thus $O_M = \langle \emptyset, T, \{O_1, O_2\}, \emptyset, \emptyset \rangle$. The advantage of a mapping ontology is that the domain ontologies are unchanged; thus, it can be created without the approval of the owners of the original ontology. However, since it is like any other ontology, it can be made publicly available and used by anyone who would like to integrate the ontologies. The disadvantages are that the integration only occurs in the perspective that is based on the mapping ontology, if the source ontologies are revised then a new mapping ontology must be created, and a set of articulation axioms are needed for each additional ontology that covers the domain.

Another approach to implementing integration is to revise each ontology to include mappings to the other. First, we create a new version of each ontology, called a mapping revision. Each revision extends the original version of the other ontology and includes a set of articulation axioms, allowing it to translate the terms from that ontology. Since each revision leaves the original vocabulary unchanged, and (assuming the articulation axioms are correct) does not change the intended models, it is backward-compatible with the original version. Thus, in the example where O'_1 is the mapping revision of O_1 and O'_2 is the mapping revision of O_2, if T_1 is the set of articulation axioms from the vocabulary of O_2 to that of O_1 and T_2 is the set of articulation axiom from the vocabulary of O_1 to O_2, then $O'_1 = \langle V_1, A_1 \cup T_1, \{O_2\}, \{O_1\}, \{O_1\}\rangle$ and $O'_2 = \langle V_1, A_2 \cup T_2, \{O_1\}, \{O_2\}, \{O_2\}\rangle$. This ensures that perspectives based on O'_1 and O'_2 will integrate resources that commit to O_1 and O_2. The advantage of this approach is that the articulation axioms are inserted into the latest versions of the ontologies, ensuring that they will apply to later backward-compatible revisions. The main disadvantage is that due to the nature of revision (see page 38), it can only be performed by the owners of the original ontologies.

A common disadvantage of the mapping ontology and mapping revision approaches is that they ignore a fundamental problem: the overlapping concepts do not belong in either domain, but are more general. The fact that two domains share the concept may mean that other domains will use it as well. If this is so, then each new domain would need a set of articulation axioms to map it to the others. Obviously this can become unwieldy very quickly. A more natural approach is to merge the common items into a more general ontology, called an intersection ontology, which is then extended by revisions to the domain ontologies. First, we create a set of terms and axioms that standardize the commonalities between O_1 and O_2, referred to as V_N and A_N, respectively. We then create a new ontology O_N, where $O_N = \langle V_N, A_N, \emptyset, \emptyset, \emptyset\rangle$. Then we determine a set of articulation axioms T_1 and T_2 that translate from O_N to O_1 and O_2, respectively. When combined with A_N, these will allow us to conclude some formulas already in A_1 and A_2; we will refer to the sets of these formulas as D_1 and D_2. Formally, $\phi \in D_1$ iff $\phi \in A_1$ and $A_N \cup T_1 \models \phi$ (similarly for D_2). Now we can define the revised ontologies O'_1 and O'_2. $O'_1 = \langle V_1, A_1 - D_1, \{O_N\}, \{O_1\}, \{O_1\}\rangle$ and $O'_2 = \langle V_2, A_2 - D_2, \{O_N\}, \{O_2\}, \{O_2\}\rangle$. Note that as with the mapping revisions approach, the revised ontologies retain the vocabulary and do not change the intended models, so they can be backward-compatible.

3.6 Inconsistency

If the logical language used by the Semantic Web is rich enough to express inconsistency, then an inconsistency within a single resource or one that exists between a pair of resources will result in one or more perspectives that are inconsistent. For example, since first-order logic is monotonic, if $K(r_1) = \{A\}$ and $K(r_2) = \{\neg A\}$ then any perspective which contains both r_1 and r_2 is inconsistent. As is well known, such an inconsistency trivializes first-order logic, allowing anything to be proven. However, the distributed nature of the Web makes it impossible to guarantee that inconsistencies will not be stated. This results in another principle of the Semantic Web:

Principle 3.34 *An inconsistency due to a single resource or a pair of resources should not make an entire perspective inconsistent.*

Although perspectives solve some of the problems of handling distributed ontologies, logical inconsistency is still a danger. A perspective can become inconsistent in three ways: if the basis ontology is inconsistent with one of its ancestors, if a resource is inconsistent with the basis ontology (or an ancestor), or if two resources are inconsistent with each other. If an ontology is designed carefully, then inconsistency between it and one of its ancestors can be prevented. We will require that a valid ontology must be consistent with all of its ancestors. When a resource commits to an ontology, it implicitly agrees to the intended models of the ontology. If the resource includes an assertion that is inconsistent with the ontology, then this agreement is violated. Therefore, it is safe to assume that such resources are invalid and can be excluded from all perspectives. However, there is still the problem that two resources which commit to the same ontology could be inconsistent. If so, any perspective that included this ontology's resource would be inconsistent. Due to the dynamic nature of resources, a resource's validity should not depend on any other resource. However, given the distributed nature of the Web, it is impossible to prevent two resources from contradicting each other. Thus, perspectives created using the approaches described above will likely become inconsistent, and cannot be easily fixed. Indeed, this could even be the form of a new, insidious, and incredibly simple denial of service attack: publish an inconsistency and watch the Semantic Web grind to a halt. Clearly, there must be a way to prevent or resolve potential inconsistencies.

A common way to handle inconsistencies is through some form of nonmonotonicity. In nonmonotonic logics, certain statements are considered defaults, and are only true if it they are not inconsistent with more explicit information. Often, an implicit assumption in these theories is that the most recent information is correct and that it is prior beliefs that must change. On the Web, this assumption cannot be made; if anything, more recent information that conflicts with prior beliefs should be approached with skepticism. Additionally, inconsistencies on the Web will often be due to fundamental disagreements, and thus neither statement can be considered the "default."

The (primarily philosophical) field of belief revision [38] focuses on how to minimize the overall change to a set of beliefs in order to incorporate new inconsistent information. A representation of an agent's knowledge at some point in time is called an epistemic state, and a change of knowledge leads to a new epistemic state via an epistemic change. The three types of epistemic changes are expansions, revisions, and contractions. An expansion adds an assertion that is consistent with the existing epistemic state. A revision adds an assertion that is inconsistent with existing beliefs, and requires that some knowledge be retracted in order to determine the new epistemic state. Finally, a contraction removes an assertion, which may lead to the removal of other assertions that depend on it. Gardenförs [38] presents a series of postulates describing epistemic changes. An important criterion is that of minimal change, that is, the only changes made to the epistemic state are those required to accommodate the new information. In the case of revision, this may require choosing between equally adequate alternatives. In such cases, the relative epistemic entrenchment of the beliefs (which determines how important they are), may be used to choose an epistemic state. However, on the Semantic Web, it is unclear how the epistemic entrenchment of an assertion should be determined. Furthermore, it is unclear that maintaining a single consistent set of beliefs makes sense in a distributed knowledge system.

The chief problem with nonmonotonic logic and theories of belief revision is choosing which set of assertions should constitute the "beliefs" of the Semantic Web. Assumption-based truth maintenance systems (ATMSs) [21] present an alternative. In an ATMS, multiple contexts are maintained, where each context represents a set of consistent assumptions. Thus it is is possible to consider the truth of a formula with respect to a particular context, or to determine the set of contexts in which a formula is true. If we assume that each ontology and resource must be internally consistent, then there can be contexts that assume that each is individually true. More complex contexts can be formed from these by assuming that multiple ontologies or resources are true at the same time. However, if there are n ontologies and resources, then there could be as many as 2^n possible contexts. Although contradictions can automatically be propagated from simpler contexts to the complex contexts that contain them, management of contexts would be a serious problem on the Semantic Web. The Web already contains over a billion resources (web pages), and many more resources are added every day. Each new resource would have to be compared with every existing context to determine which new consistent contexts can be formed.

A different solution is to limit the language so that it is not possible to express logical inconsistencies. In first-order logic, this can be done by omitting negation.[1] Other logics, particularly description logics, include features such as cardinality constraints and the specification of disjoint classes, which can lead to inconsistency. The main argument against limiting the language to prevent logical inconsistency is that inconsistency can be a sign that two theories should not be combined. Still, the advantage of limiting the language is that it does not have the computational or philosophical problems associated with the other methods.

Unlike the previous sections, we do not suggest a solution to the problem of inconsistency here. We have discussed the relative benefits of various alternatives, but believe that only future research will determine the best choice for the Semantic Web. In Section 4.1, we will discuss the choice made for the SHOE language.

3.7 Scalability

Throughout this chapter, we have used first-order logic as the basis for our discussion of the Semantic Web. However, sound and complete reasoning in first-order logic is intractable, meaning that there is no polynomial-time algorithm that solves the problem. Thus, first-order logic systems will not scale to the quantity of formulas that would be expected on the Semantic Web. So then how can the problem of scalability be handled in a practical setting?

One approach is to use reasoning methods that are not sound and complete. Resource-bounded reasoning algorithms (that limit the computation time, number of steps, etc.) are quite common and would be applicable for many Semantic Web applications. In many cases, it is not necessary to know all of the answers on the Semantic Web, only a handful of correct ones will suffice. Given the extent of the Web, it is unlikely that any reasoner will have access to all of the assertions, so it is improbable that even one which used a sound and complete algorithm would be truly complete in the global sense.

Another approach to scalability is to reduce the expressivity of the language. This has been an important direction for the knowledge representation community which has tried to characterize the computational complexity of languages with various features. Starting with Brachman and Levesque [11], the complexity of different forms of description logics has been computed, and languages have been developed that attempt to maximize expressivity while minimizing complexity. Even so, subsumption is intractable in many description logics.

An alternative to description logics is to use Horn logic. It has been shown that although Horn-

[1]Here we mean only omission of the logical operator, and not of other logical connectives that can be rewritten by using negation, such as implication.

logic and the most common description logics can express things the other cannot, neither is more expressive than the other [8]. Thus the relative advantages of the two languages depend on the kinds of constructs viewed as most useful to the Semantic Web. In Section 3.6, we discussed the problems inherent in languages that include negation or other features that may lead to inconsistency; most description logics face these problems due to the presence of cardinality restrictions. Horn logic on the other hand can not be logically inconsistent. Furthermore, if we restrict the language to datalog, which is a minor variant of Horn logic, then polynomial reasoning algorithms such as the magic sets technique can be used.

As in Section 3.6, we do not provide a solution to scalability problem here. This is another difficult issue, and only future use of the Semantic Web will determine the right combination of language features and query methods. In Section 4.1 we will explain the choice made for the SHOE language, and in Section 5.2 will discuss the use of reasoning systems with different inferential capabilities.

3.8 Semantic Web Queries

The design of a semantic web language requires consideration of how the language will be used. The Semantic Web can be used to locate documents for people or to answer specific questions based on the content of the Web. These uses represent the document retrieval and knowledge base views of the Web.

The knowledge base view uses the logical definition of queries: a query is a formula with existentially quantified variables, whose answers are a set of bindings for the variables that make the formula true with respect to the knowledge base. But what is a knowledge base in the context of the Semantic Web? In order to resolve a number of problems faced by the Semantic Web we have extensively discussed means of subdividing it. Theoretically, each of these perspectives represents a single model of the world, and could be considered a knowledge base. Thus, the answer to a semantic web query must be relative to a specific perspective.

Consider the set of ontologies and resources presented in Figure 3.7. There are three compatible ontology perspectives generated from this data: $COP_G(R)$, $COP_U(R)$, and $COP_F(R)$. Based on Definition 3.31, different ontologies and resources appear in each perspective. For example, $COP_G(R)$ includes the axioms from O_G and the knowledge from r_1 and r_2. It does not include r_3 or r_4 because these commit to other ontologies. $COP_U(R)$ includes the axioms from O_U and, because O_G is an ancestor of O_U, those of O_G. It also includes the resources r_1, r_2, and r_3, which commit to these ontologies. On the other hand, $COP_F(R)$ includes axioms from O_F and O_G, and the resources r_1, r_2, and r_4. As a result, the answer to any particular query depends on which perspective it is issued against. As shown in Figure 3.8, the answer to $Person(x)$ in $COP_G(R)$ is just $\{bob\}$ because from this perspective the axioms and resources of O_U are irrelevant. However, in $COP_U(R)$, the answer is $\{bob, kate\}$ because we have the axiom from O_U that tells us every *Chair* is a *Person*. Also note that in $COP_F(R)$, the answer is $\{bob\}$ because O_F includes O_G. When we ask a query such as $Chair(x)$, then the variety in answers is even greater. In $COP_U(R)$ the answer is $\{kate\}$ while in $COP_F(R)$ it is $\{recliner29\}$. This is because the perspectives decontextualize the term *Chair* differently. Also note that in $COP_G(R)$, the query is ill-formed with respect to the ontology that serves as the basis of the perspective (i.e., the ontology does not include *Chair* in its vocabulary), and thus there can be no answers.

$$\begin{array}{ll}
O_G = & \langle\{Thing, Person, Object\}, \\
& \{Person(x) \rightarrow Thing(x), \\
& Object(x) \rightarrow Thing(x)\}, \\
& \emptyset, \\
& \emptyset, \\
& \emptyset\rangle \\
O_U = & \langle\{Chair, Person, Object\}, \\
& \{Chair(x) \rightarrow O_G : Person(x), \\
& Person(x) \leftrightarrow O_G : Person(x), \\
& Object(x) \leftrightarrow O_G : Object(x)\} \\
& \{O_G\}, \\
& \emptyset, \\
& \emptyset\rangle \\
O_F = & \langle\{Chair, Person, Object\}, \\
& \{Chair(x) \rightarrow O_G : Object(x), \\
& Person(x) \leftrightarrow O_G : Person(x), \\
& Object(x) \leftrightarrow O_G : Object(x)\} \\
& \{O_G\}, \\
& \emptyset, \\
& \emptyset\rangle \\
C(r_1) = & O_G \\
K(r_1) = & \{Person(bob)\} \\
C(r_2) = & O_G \\
K(r_2) = & \{Object(sofa52)\} \\
C(r_3) = & O_U \\
K(r_3) = & \{Chair(kate)\} \\
C(r_4) = & O_F \\
K(r_4) = & \{Chair(recliner29)\}
\end{array}$$

Figure 3.7: Example ontologies and resources.

	Perspective		
Query	$COP_G(R)$	$COP_U(R)$	$COP_F(R)$
$Person(x)$	bob	$bob, kate$	bob
$Object(x)$	$sofa52$	$sofa52$	$sofa52, recliner29$
$Chair(x)$	n/a	$kate$	$recliner29$

Figure 3.8: Query answers for different perspectives.

From the point of view of most agents, the Semantic Web can be seen as read-only. This is because most web pages can only be updated by their owners via the file mechanisms on their servers. A file is either saved or unsaved, and only saved files are available via HTTP. Thus, the update of a file becomes a single transaction and many issues that are important to databases, such as concurrency and serializability are not significant. Although web pages usually have a single writer, there are many web pages that change frequently. If the agent reaccesses such pages in the middle of the query, it may be presented with a different set of assertions. For these reasons, it is recommended that for the duration of each query, reasoning systems cache the assertions of all resources used in the query.

Document retrieval queries can locate a document that represents a concept, which may or may not be partially defined, or locate a document that has specified metadata. Here, metadata is data about the document itself, such as its author or modification date. When the domain of the language includes resources, then the knowledge base view subsumes the the document retrieval view. That is, we can specify the relationship between a document and the concept that it represents, and we can describe both the document and the concept independently.

3.9 Summary

In this chapter, we have gradually developed a formal model for describing the Semantic Web. The foundation for this model is first-order logic, but we found that we needed ontologies to represent common background knowledge and provide reusable vocabularies. We then presented a method of partitioning ontologies and resources to ensure that only those that shared the same context were integrated. We extended this model with the ability to specify ontology inclusion, so that content providers could describe their own information needs while still reusing existing ontologies. This allows us to increase the integration of distributed resources, as is done with extended ontology perspectives. We further extended the model to deal the problem of ontology evolution, and discussed the issue of backward-compatibility. This resulted in compatible ontology perspectives which can integrate resources that commit to any ancestors of an ontology as well as resources that commit to forward-compatible prior versions of the ontology. We discussed how extension alone would be insufficient for integrating resources in a distributed ontology environment, and discussed the problem of ontology divergence. Another important issue in distributed environments is the inability to preserve integrity constraints and the likelihood of global inconsistency. The size of the Web makes scalability an important issue and possible approaches were also discussed. Finally, we talked about queries on the Semantic Web, and how they depend on perspectives.

BIBLIOGRAPHY

[1] G. Arocena, A. Mendelzon, and G. Mihaila. Applications of a web query language. In *Proceedings of ACM PODS Conference*, Tuscon, AZ, 1997.

[2] N. Ashish and C. Knoblock. Semi-automatic wrapper generation for internet information sources. In *Proceedings of the Second IFCIS Conference on Cooperative Information Systems (CoopIS)*, Charleston, SC, 1997.

[3] T. Berners-Lee and D. Connolly. *Hypertext Markup Language - 2.0*. IETF HTML Working Group, November 1995. At: http://www.ics.uci.edu/pub/ietf/html/rfc1866.txt.

[4] T. Berners-Lee, J. Hendler, and O. Lasilla. The Semantic Web. *Scientific American*, May 2001.

[5] Tim Berners-Lee. *Weaving the Web*. HarperCollins Publishers, New York, NY, 1999.

[6] P. Biron and A. Malhotra. *XML Schema Part 2: Datatypes*. W3C, May 2001. At: http://www.w3.org/TR/2001/REC-xmlschema-2-20010502.

[7] D. Bobrow and T. Winograd. An overview of KRL, a knowledge representation language. *Cognitive Science*, 1(1), 1977.

[8] Alexander Borgida. On the relationship between description logic and predicate logic. In *Proceedings of CIKM-94*, pages 219–225, 1994.

[9] A. Bouguettaya, B. Benatallah, and A. Elmagarmid. An overview of multidatabase systems: Past and present. In Elmagarmid et al. [25].

[10] R. Brachman. What IS-A is and isn't: An analysis of taxonomic links in semantic networks. *IEEE Computer*, 16(10):30–36, October 1983.

[11] R. Brachman and H. Levesque. The tractability of subsumption in frame-based description languages. In *Proc. of the National Conference on Artificial Intelligence (AAAI-1984)*, pages 34–37, Menlo Park, CA, 1984. AAAI/MIT Press.

[12] R. Brachman, D. McGuinness, P.F. Patel-Schneider, L. Resnick, and A. Borgida. Living with Classic when and how to use a KL-ONE-like language. In J. Sowa, editor, *Explorations in the Representation of Knowledge*. Morgan-Kaufmann, CA, 1991.

[13] R. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2), 1985.

[14] T. Bray, D. Hollander, and A. Layman. *Namespaces in XML*. W3C, January 1999. At: http://www.w3.org/TR/1999/REC-xml-names-19990114/.

[15] T. Bray, J. Paoli, and C. Sperberg-McQueen. *Extensible Markup Language (XML)*. W3C (World Wide Web Consortium), February 1988. At: http://www.w3.org/TR/1998/REC-xml-19980210.html.

[16] D. Brickley and R.V. Guha. *Resource Description Framework (RDF) Schema Specification (Candidate Recommendation)*. W3C (World-Wide Web Consortium), March 2000. At: http://www.w3.org/TR/2000/CR-rdf-schema-20000327.

[17] V. Chaudhri, A. Farquhar, R. Fikes, P. Karp, and J. Rice. OKBC: A programatic foundation for knowledge base interoperability. In *Proc. of the Fifteenth National Conference on Artificial Intelligence (AAAI-1998)*, pages 600–607, Menlo Park, CA, 1998. AAAI/MIT Press.

[18] J. Clark. *XSL Transformations (XSLT)*. W3C (World-Wide Web Consortium), November 1999. At: http://www.w3.org/TR/1999/REC-xslt-19991116.

[19] C. Collet, M. Huhns, and W. Shen. Resource integration using a large knowledge base in Carnot. *IEEE Computer*, 24(12), December 1991.

[20] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigram, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proc. of the Fifteenth National Conference on Artificial Intelligence (AAAI-1998)*, Menlo Park, CA, 1998. AAAI/MIT Press.

[21] J. de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28(2):127–162, 1986.

[22] S. Decker, D. Fensel, F. van Harmelen, I. Horrocks, S. Melnik, M. Klein, and J. Broekstra. Knowledge representation on the Web. In *Proceedings of the 2000 International Workshop on Description Logics (DL2000)*, Aachen, Germany, August 2000.

[23] Stefan Decker, Michael Erdmann, Dieter Fensel, and Rudi Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In *Semantic Issues in Multimedia Systems. Proceedings of DS-8*, pages 351–369, Boston, 1999. Kluwer Academic Publisher.

[24] S. Dobson and V. Burrill. Lightweight databases. *Computer Networks and ISDN Systems*, 27(6):1009–1015, 1995.

[25] A. Elmagarmid, M. Rusinkiewicz, and A. Sheth, editors. *Management of Heterogeneous and Autonomous Database Systems*. Morgan Kaufmann, San Francisco, 1999.

[26] O. Etzioni, K. Golden, and D. Weld. Sound and efficient closed-world reasoning for planning. *Artificial Intelligence*, 89:113–148, 1997.

[27] M. Evett, W. Andersen, and J. Hendler. Providing computational effective knowledge representation via massive parallelism. In L. Kanal, V. Kumar, H. Kitano, and C. Suttner, editors, *Parallel Processing for Artificial Intelligence*. Elsevier Science, Amsterdam, 1993.

[28] A. Farquhar, A. Dappert, R. Fikes, and W. Pratt. Integrating information sources using context logic. In *Proceedings of AAAI Spring Symposium on Information Gathering from Distributed, Heterogeneous Environments*, 1995. Also available as KSL Technical Report KSL-95-12.

[29] A. Farquhar, R. Fikes, and J. Rice. The Ontolingua Server: A tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46(6):707–727, 1997.

[30] Adam Farquhar, Richard Fikes, and James Rice. Tools for assembling modular ontologies in Ontolingua. In *Proc. of Fourteenth American Association for Artificial Intelligence Conference (AAAI-97)*, pages 436–441, Menlo Park, CA, 1997. AAAI/MIT Press.

[31] D. Fensel, S. Decker, M. Erdmann, and R. Studer. Ontobroker: How to enable intelligent access to the WWW. In *AI and Information Integration, Papers from the 1998 Workshop,* Menlo Park, CA, 1998. AAAI Press. Technical Report WS-98-14.

[32] D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. Klein. OIL in a nutshell. In *Knowledge Acquisition, Modeling, and Management, Proceedings of the European Knowledge Acquisition Conference (EKAW-2000)*, pages 1–16, Berlin, 2000. Springer-Verlag.

[33] D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. Patel-Schneider. OIL: An ontology infrastructure for the Semantic Web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[34] N. Foo. Ontology revision. In *Conceptual Structures: Third International Conference*, pages 16–31, Berlin, 1995. Springer-Verlag.

[35] ISO (International Organization for Standardization). ISO 8879:1986(E). information processing – text and office systems – Standard Generalized Markup Language (SGML), 1986.

[36] D. Freitag. Information extraction from HTML: Application of a general machine learning approach. In *Proc. of Fifteenth American Association for Artificial Intelligence Conference (AAAI-98)*, pages 517–523, Menlo Park, CA, 1998. AAAI/MIT Press.

[37] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.

[38] Peter Gardenförs. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press, Cambridge, Mass., 1988.

[39] Michael Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA, 1987.

[40] Michael R. Genesereth, Arthur M. Keller, and Oliver Duschka. Infomaster: An information integration system. In *Proceedings of 1997 ACM SIGMOD Conference,* May 1997.

[41] M.R. Genesereth and R.E. Fikes. Knowledge Interchange Format, version 3.0 reference manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.

[42] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[43] M. Grüninger. Designing and evaluating generic ontologies. In *Proceedings of ECAI'96 Workshop on Ontological Engineering*, 1996.

[44] N. Guarino. Formal ontology and information systems. In *Proceedings of Formal Ontology and Information Systems*, Trento, Italy, June 1998. IOS Press.

[45] N. Guarino and P. Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. In N. Mars, editor, *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pages 25–32. IOS Press, Amsterdam, 1995.

[46] R. V. Guha. *Contexts: A Formalization and Some Applications*. PhD thesis, Stanford University, 1991.

[47] J. Heflin and J. Hendler. Searching the Web with SHOE. In *Artificial Intelligence for Web Search. Papers from the AAAI Workshop*, pages 35–40, Menlo Park, CA, 2000. AAAI Press. Technical Report WS-00-01.

[48] J. Heflin and J. Hendler. Semantic interoperability on the Web. In *Proc. of Extreme Markup Languages 2000*, pages 111–120, Alexandria, VA, 2000. Graphic Communications Association.

[49] J. Heflin and J. Hendler. A portrait of the Semantic Web in action. *IEEE Intelligent Systems*, 16(2):54–59, 2001.

[50] J. Heflin, J. Hendler, and S. Luke. Reading between the lines: Using SHOE to discover implicit knowledge from the Web. In *AI and Information Integration, Papers from the 1998 Workshop*, Menlo Park, CA, 1998. AAAI Press. Technical Report WS-98-14.

[51] J. Heflin, J. Hendler, and S. Luke. Applying ontology to the Web: A case study. In J. Mira and J. Sanchez-Andres, editors, *International Work-Conference on Artificial and Natural Neural Networks, IWANN'99, Proceeding, Vol. II*, pages 715–724, Berlin, 1999. Springer.

[52] Jeff Heflin and James Hendler. Dynamic ontologies on the Web. In *Proc. of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 443–449, Menlo Park, CA, 2000. AAAI/MIT Press.

[53] J. Hendler. Agents and the Semantic Web. *IEEE Intelligent Systems*, 16(2):30–37, 2001.

[54] James Hendler and Deborah L. McGuinness. The DARPA agent markup language. *IEEE Intelligent Systems*, 15(6):72–73, November/December 2000.

[55] Joint US/EU Ad Hoc Agent Markup Language Committee. *DAML+OIL*, 2001. At: http://www.daml.org/2001/03/daml+oil-index.

[56] V. Kashyap and A. Sheth. Semantic similarities between objects in multiple databases. In Elmagarmid et al. [25].

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

[58] Craig A. Knoblock, Steven Minton, Jose Luis Ambite, Naveen Ashish, Pragnesh Jay Modi, Ion Muslea, Andrew G. Philpot, and Sheila Tejada. Modeling web sources for information integration. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 211–218, Menlo Park, CA, 1998. AAAI/MIT Press.

[59] D. Konopnicki and O. Shemueli. W3QS: A query system for the World Wide Web. In *Proceedings of the 21st International Conference on Very Large Databases*, Zurich, Switzerland, 1995.

[60] N. Kushmerick, D. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *Proc. of Fifteenth International Joint Conference on Artificial Intelligence*, pages 729–735, San Francisco, 1997. Morgan Kaufmann.

[61] O. Lassila. Web metadata: A matter of semantics. *IEEE Internet Computing*, 2(4):30–37, 1998.

[62] O. Lassila and R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification*. World Wide Web Consortium (W3C), February 1999. At: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

[63] D. Lenat, R.V. Guha, K. Pittman, D. Pratt, and M. Shepherd. Cyc: Toward programs with common sense. *Communications of the ACM*, 33(8):30–49, 1990.

[64] Douglas Lenat and R.V. Guha. *Building Large Knowledge-Based Systems*. Addison-Wesley, Reading, Mass., 1990.

[65] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin, 1987.

[66] S. Luke, L. Spector, D. Rager, and J. Hendler. Ontology-based web agents. In *Proceedings of the First International Conference on Autonomous Agents*, pages 59–66, New York, NY, 1997. Association of Computing Machinery.

[67] R. MacGregor. The evolving technology of classification-based knowledge representation systems. In J. Sowa, editor, *Explorations in the Representation of Knowledge*. Morgan-Kaufmann, CA, 1991.

[68] J. McCarthy. Notes on formalizing context. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence (IJCAI-93)*, pages 555–560, Los Altos, 1993. Morgan Kaufmann.

[69] Deborah L. McGuinness, Richard Fikes, James Rice, and Steve Wilder. An environment for merging and testing large ontologies. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, Breckenridge, Colorado, April 2000.

[70] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A database management system for semistructured data. *SIGMOD Record*, 26(3):54–66, 1997.

[71] S. McIlraith, T. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

[72] J. Minker, editor. *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann, Los Altos, CA, 1988.

[73] M. Minsky. A framework for representing knowledge. In Patrick-Henry Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, New York, 1975.

[74] N. Noy and C. Hafner. The state of the art in ontology design. *AI Magazine*, 18(3):53–74, 1997.

[75] N. F. Noy and M. A. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, 2000.

[76] N. F. Noy, M. Sintek, S. Decker, M. Crubézy, R. W. Fergerson, and M. A. Musen. Creating semantic web contents with Protégé-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.

[77] D. E. Oliver, Y. Shahar, M. A. Musen, and E. H. Shortliffe. Representation of change in controlled medical terminologies. *Artificial Intelligence in Medicine*, 15(1):53–76, 1999.

[78] Y. Papakonstantinou, H. Garcia-Molina, and J. Ullman. A query translation scheme for rapid implementation of wrappers. In *Proceedings of the Conference on Deductive and Object-Oriented Databases(DOOD)*, Singapore, 1995.

[79] M. R. Quillian. Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science*, 12:410–430, 1967.

[80] D. Ragget. *HTML 3.2 Reference Specification*. W3C (World Wide Web Consortium), January 1997. At: http://www.w3.org/TR/REC-html32.

[81] R. Ramakrishnan and J. D. Ullman. A survey of research on deductive database systems. *Journal of Logic Programming*, 23(2):126–149, May 1995.

[82] M. Roth and P. Schwarz. Don't scrap it, wrap it! A wrapper architecture for legacy data sources. In *Proceedings of 23rd International Conference on Very Large Data Bases*, 1997.

[83] K. Sagonas, T. Swift, and D. S. Warren. XSB as an efficient deductive database engine. In R. T. Snodgrass and M. Winslett, editors, *Proc. of the 1994 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'94)*, pages 442–453, 1994.

[84] K. Stoffel, M. Taylor, and J. Hendler. Efficient management of very large ontologies. In *Proc. of Fourteenth American Association for Artificial Intelligence Conference (AAAI-97)*, Menlo Park, CA, 1997. AAAI/MIT Press.

[85] H. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. *XML Schema Part 1: Structures*. W3C, May 2001. At: http://www.w3.org/TR/2001/REC-xmlschema-1-20010502.

[86] J. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1. Computer Science Press, Rockville, MD, 1988.

[87] J. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 2. Computer Science Press, Rockville, MD, 1988.

[88] J. Vega, A. Gomez-Perez, A. Tello, and Helena Pinto. How to find suitable ontologies using an ontology-based WWW broker. In J. Mira and J. Sanchez-Andres, editors, *International Work-Conference on Artificial and Natural Neural Networks, IWANN'99, Proceeding, Vol. II*, pages 725–739, Berlin, 1999. Springer.

[89] W3C. *XHTML 1.0: The Extensible HyperText Markup Language*, January 2000. At: http://www.w3.org/TR/2000/REC-xhtml1-20000126.

[90] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3), 1992.

[91] G. Wiederhold. An algebra for ontology composition. In *Proceedings of the 1994 Monterey Workshop on Formal Methods*, pages 56–62. U.S. Naval Postgraduate School, 1994.

[92] G. Wiederhold. Interoperation, mediation, and ontologies. In *Workshop on Heterogeneous Cooperative Knowledge-Bases*, pages 33–48, Tokyo, Japan, 1994.