

1

The Semantic Web

At the present time, the Web is primarily designed for human consumption and not for computer consumption. This may seem like an unusual state of affairs, given that the web is vast and mature computerized information resource. However, we must recognize that the computer is presently used as the carrier of this information, and not as the consumer of the information. As a result, a great deal of the potential of the Web has yet to be realized.

This book explores the challenges of *automatic* computer-based processing of information on the Web. In effect, we want to enable computers to use Web-based information in much the same way as humans presently do. Our motivation is that computers have a brute-force advantage over humans. Where we can gather and process information from a handful of Web-based sources, a computer may download and compare thousands of such sources in a matter of seconds. Nonetheless, despite the apparent simplicity of this task, there are a great many issues that must be addressed if we are to make effective use of this information. As a result, the automated processing of Web-based information is still in its infancy. In this book, we show how many different techniques can be used together to address this task.

The automated processing of information on the Web is principally an issue of *scale*. There are many existing techniques in Computer Science and Artificial Intelligence (AI) that may be appropriate to the task. However, there are significant issues that must be addressed if we are to scale up these techniques for use on the Web. Therefore, we present a detailed overview of the current state of the art, with a particular emphasis on practical solutions. The methods and technologies that we present in this book are of importance to all computer practitioners, as they will shape the future evolution of the Web.

1.1 Information and knowledge

To appreciate the challenges of computer-based consumption of Web-based information, we consider the following scenario. Suppose we are searching the Web for information on a specific ailment. In response to our query, we find a page that contains the requested keywords, but the information is beyond our comprehension, say a paper from a medical journal. We can read the paper in detail, and look up any unfamiliar words in a dictionary. We can also go further, and examine the references presented in the paper, and even contact the author. However, it is likely that we will have little more understanding of the content of the paper upon completion than we did at the start. What we are lacking is the background knowledge necessary to understand the paper, which can only be obtained through many years of study.

The scenario that we have described is very similar to the situation we face when attempting to process information on a web page automatically, by a computer program. Our program can readily count the keywords in the page, download the images, and follow the relevant hyperlinks. However, our program will have no real understanding of the content of the page, other than statistical data. We will not be able to do anything with this information beyond what can be achieved by purely statistical means. This statistical data can be used to good effect, as shown by the current generation of Web search technology. However, the limitations of this approach to Web search are often all too apparent.

We can illustrate the limitations of Web search when attempting to perform a search that goes beyond what can be accomplished by keywords alone. For example, suppose we wish to find the best recipe for making a chocolate cake. We perform a search with the keywords ‘chocolate cake recipe’, and are faced with over half a million matches. We can attempt to narrow the search by including additional keywords such as ‘best’ or ‘good’, but this does little to reduce the number of results. Ultimately, it will be up to us to examine the results, and decide on an appropriate recipe, though it is highly unlikely that we could examine all of the candidates. It may appear that our chocolate cake example is unrealistic, as the definition of the ‘best’ cake relies on personal preference. However, we have presented this exaggerated example to illustrate a problem, which results from the fact that a computer has no notion of what a ‘chocolate cake’ is.

A more realistic demonstration of the limitations of the Web can be obtained by considering an example that could technically be computed. For example, determining the best method for travelling to a particular destination. In this case, the relevant information can be found on the Web, and we can define ‘best’ mathematically, e.g. as the cheapest route. However, unless a suitable application is already available on the Web, the

computer will be unable to provide us with the desired result. A regular keyword search will be of no use, other than finding a large list of candidate sites containing the relevant information. The computer cannot calculate the desired result, as it has no understanding of the information that it perceives on the Web. It is because of this lack of understanding that the computer is unable to perform any kind of automatic inference with this information. In principle, if the computer was aware of the meaning of the information, then the required calculation could be performed automatically.

The fundamental issue that we are experiencing is related to the classical problem of *information* versus *knowledge*. What exists on the Web at the present time is information, essentially a large collection of facts. To make use of this information, we need to appreciate these facts in the wider context of knowledge. By this, we mean that the information must be interpreted in light of the *concepts*, such as truths, beliefs, perspectives, judgements, methodologies, and know-how. At present, the Web relies entirely on the human user to supplement the information with knowledge to make use of the information. We later show that a computer can make many of the same kinds of inference as a human if the information on the Web is supplemented with *semantic* knowledge about this information.

The provision of semantic knowledge associated with the information on the Web will open the door to the construction of a whole new class of intelligent Web applications. A number of authors have outlined their vision of a *Semantic Web*, and the applications that this will enable. These accounts appear in the Suggested Reading section at the end of the chapter. Examples of applications include intelligent searches, automated data mining, e-science experiments, e-learning systems, personalized newspapers and journals, intelligent devices, and so on. These motivating examples provide a flavour of the power and flexibility of associating semantic knowledge with the existing information of the Web. The remainder of this book is devoted to the challenges of realizing these goals. There are essentially two main questions that we answer:

1. How do we represent the knowledge in such a way that it can be understood and processed automatically by a computer?
2. Once we have this information, how can we use it effectively in real applications?

1.1.1 Knowledge representation

In our discussion, we have identified knowledge as the key to unlocking the potential of the information held on the Web. With a suitable *representation* of this knowledge, we can perform *inference* about the information, and thereby obtain new insights. However, before we perform any kind

of inference, we must consider how this knowledge can be represented, so that it can be automatically processed and shared. It is necessary to adopt appropriate conventions that can be consistently interpreted by both the producers and the consumers of the knowledge.

Before we discuss these conventions, it is important to consider the big picture of what we are trying to represent. We previously talked about knowledge in terms of concepts such as beliefs. These concepts, together with the facts on which they are defined, and the relationships between them form the basis of our representation. What we actually represent is a *conceptualization*, which is a simplified view of a part of the world.

At this point it is helpful to consider an example of a conceptualization. Figure 1.1 presents a specification that can be used to classify cameras into different categories. The boxes in the example contain the concepts, and the arrows define the relationships between them. The root of the hierarchy is a class called *Thing*, which represents the universe of discourse, i.e. everything that we could have in our conceptualization. *Thing* has two direct subclasses: *Tangible Thing*, which have a real physical existence, and *Intangible Thing*, which do not. This is a useful distinction as it enables us to separate real physical objects, such as the Camera itself, from properties such as Autofocus. This is a common distinction in the design of conceptual

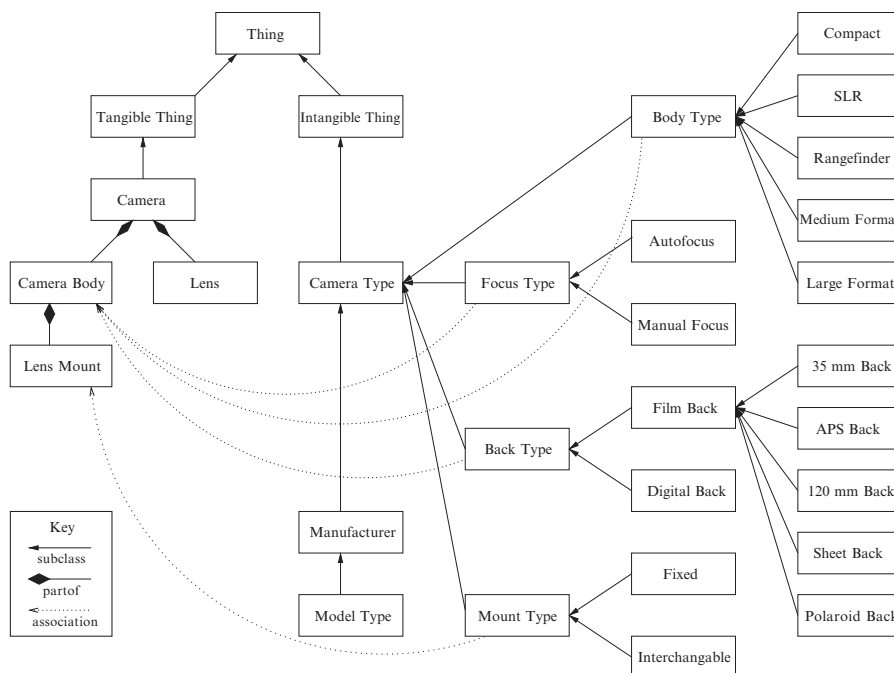


Figure 1.1 Camera ontology.

hierarchies. The remainder of the hierarchy identifies the key concepts that are important to photographers in categorizing different types of camera.

The style of definition that we have used should be familiar to anyone who has encountered a class hierarchy in an object-oriented programming language such as Java, though there are some important differences. The most apparent difference is that we define two kinds of relationship, in addition to the usual *subclass* relation. The *partof* relation is used in relation to Tangible things, and indicates that a thing is a physical part of another thing. The *association* relationship relates intangible things to tangible things. For example, Body Type is a property of a Camera Body. Our use of this relationship in our example indicates that we do not have a strict tree-like hierarchy, rather we have defined a graph, or network, of relationships.

To illustrate the use of our example hierarchy, we present a list of the features for a range of popular cameras in Table 1.1. This is similar to a list of cameras that may appear on the Web, for example, on a shopping website. It should be clear that we can classify all of these cameras according to the conceptualization that we have defined. By performing this classification, we are representing knowledge about each camera and constructing a *knowledge base*.

The advantage of the classification is that we can perform inference on the information. This inference can provide additional knowledge that is not readily apparent from the initial list of features. We can infer facts from our examples such as: all SLR cameras have an interchangeable lens, all compact cameras have a fixed lens, and all digital cameras are autofocus. Our confidence in these inferences would clearly be enhanced if our list of examples were larger. We can also answer questions about a specific camera, for example, what type of body does it have? Comparisons of features between the cameras can also be performed, and we can ask for all the cameras of a particular type. Further inference can be performed in relation to additional knowledge. For example, if we had a list of

Table 1.1 *Camera features.*

Camera	Features
Olympus MD3	Compact, Autofocus, APS, Fixed Lens
Canon Ixus 500	Compact, Autofocus, Digital Back, Fixed Lens
Pentax K1000	SLR, Manual Focus, 35 mm, Interchangeable Lens
Nikon D70	SLR, Autofocus, Digital Back, Interchangeable Lens
Leica M2	Rangefinder, Manual Focus, 35 mm, Interchangeable Lens
Hasselblad H1	Medium Format, Autofocus, 120 mm, Interchangeable Lens

prices available, we could infer the cheapest camera with a specific set of features.

Representing knowledge in the form of a conceptualization is central to the automatic processing of information on the Web. As we have shown, this kind of representation can be used to make inferences. In particular, we can infer facts that would be difficult or impossible to determine otherwise. Nonetheless, there are a further two important considerations that must be addressed:

1. We need a suitable conceptualization model for the information that we wish to classify.
2. We need to actually perform the classification of the information, and this is itself a difficult task.

1.1.2 Ontologies and knowledge lifecycles

We have illustrated how a body of knowledge can be represented as a conceptualization, which is an abstract view of the world. More formally, we can define an *ontology* that is a specification of a conceptualization. This term is borrowed from philosophy, where an ontology refers to the study of existence, and the fundamental categories of being. For our purposes, we define existence as that which can be represented. A formal ontology defines a set of objects, and the relationships among them. The ontology may also define axioms that constrain the interpretation and usage of the objects. More precisely, an ontology is the statement of a logical theory.

A formal ontology is usually defined as a knowledge vocabulary, rather than in a graphical notation. There are a number of different languages that can be used to define an ontology. For the Web, RDF, RDFS, and the OWL family of languages are the most relevant. These languages use XML-syntax, and have varying degrees of expressivity. The underlying semantics in these languages is provided by graph theory and description logics.

Designing a formal ontology is a difficult task. Even a simple ontology, such as our Camera example, can take a lot of effort to produce. This difficulty has long been recognized, and many kinds of methodology and tools have been produced to assist in the construction of formal ontologies. More recent thinking, principally in light of the Web, has highlighted the dynamic nature of knowledge. A formal ontology should not be considered in isolation. Instead, ontologies should be linked together, parts of ontologies should be derived from others, and ontologies should change over time in response to trends.

The dynamic view of knowledge has resulted in the concept of a knowledge *lifecycle*, as illustrated in Figure 1.2. A lifecycle expresses the craft

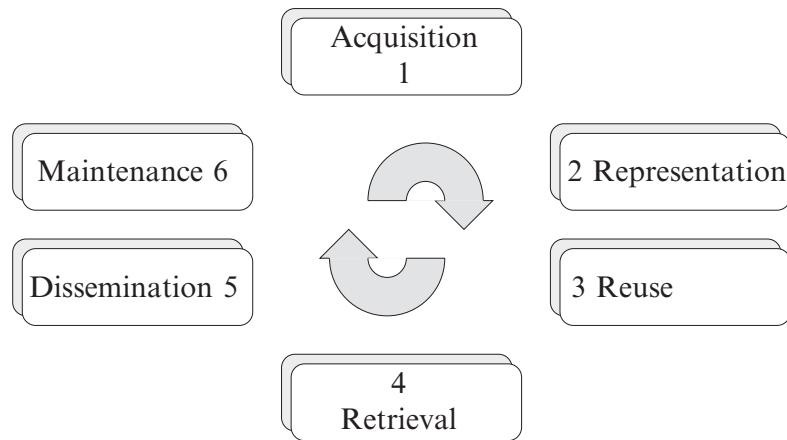


Figure 1.2 *Knowledge lifecycle.*

of knowledge management through a set of distinct tasks. The lifecycle model illustrated contains six possible tasks that express the management of Web-based knowledge. The application of these tasks can be performed in a variety of different orderings, over a range of time periods, and tasks can be omitted. The results of this application can be viewed as a flow of knowledge over time.

The first task in the lifecycle model is the acquisition of the information that we wish to represent in the ontology. This information can come from a variety of different sources such as human experts, though we are primarily interested in Web-based sources here. As we have previously stated, the Web contains a vast body of information. Therefore, this task is concerned with sifting potential sources of information to find the relevant pieces, identifying gaps in the knowledge, and integrating knowledge from different sources.

Representing the acquired information in a form suitable for problem-solving is the second task of the lifecycle. This is the point at which an ontology is constructed so that the acquired knowledge can be classified. Design decisions about the ontology must be made so that the ontology will be suitable for inference and flexible enough for future needs. In particular, the ontology must be representationally adequate for the needs of the domain in question.

Constructing an ontology and the associated knowledge base entirely from scratch is not typically a profitable exercise, particularly if existing ontologies are available in the domain. Therefore, the third task in the lifecycle is the reuse of existing knowledge sources. Typically, the representation of knowledge is designed specifically for a particular kind of

problem-solving. Understanding and adapting the knowledge already at hand can result in a more general-purpose representation, and increase the utility of existing knowledge.

The fourth task in the lifecycle is the retrieval of knowledge from our representation. When the quantity of knowledge that we have available gets very large, finding the correct bit of knowledge can be a challenge in itself. It is necessary to be able to find the correct knowledge relevant to a particular problem. This task is significantly harder if we have knowledge that changes rapidly over time, such as news headlines.

Once we have constructed our ontology and knowledge base, we would like to make it available for others to use. This is addressed in the fifth task of the lifecycle. Our knowledge will be used by different users for different purposes. Some users may want to update the knowledge, while others may wish to visualize the data in ways that we did not originally envisage. Certain knowledge may be time critical, and must be made available at the right time. We may also wish to restrict the knowledge to certain kinds of users, or present only a subset of the knowledge.

The final task in the lifecycle is the maintenance of our knowledge to preserve its usefulness. This task involves the update of content to reflect changes, and the removal of obsolete content. This may involve a deep analysis of the knowledge in question. Additions or changes to the knowledge base may involve updating the underlying ontology. The verification of the accuracy of the content is also an important maintenance issue.

The lifecycle model does not prescribe a rigid methodology for the management of ontological knowledge. Instead, it provides a summary of many different kinds of operations that we would like to perform on our Web-based knowledge, from the initial acquisition to long-term maintenance. These tasks are intended to serve as a guide for the kinds of issues that we should consider in the construction of a knowledge base.

1.2 Agency and reasoning

The use of ontologies addresses the question of how to represent knowledge on the Web such that it can be understood by a computer. This is clearly a necessary requirement in automated processing of information on the Web. We now turn our attention to the second question, and consider how we can use this knowledge effectively. In answering this question, we move beyond representational issues, and consider the technologies and applications that will be used to realize the benefits of the Semantic Web.

The Semantic Web vision promotes the notion of *agents* as the primary consumers of knowledge. These agents are programs that will collect Web

content from diverse sources, process the information, and exchange the results with other agents. Agents are not just arbitrary programs, rather they are programs with *autonomous* and *rational* behaviours that interact with each other. These behaviours are required for the dynamic kind of systems that we want to construct. In particular, we want to define agents that can go out onto the Web and perform tasks on our behalf, without our direct intervention. The construction of programs with this kind of behaviour continues to be an active AI research area.

There is a surprising lack of consensus over how the term *agent* should actually be defined. In surveying the literature, it quickly becomes clear that there are many different definitions. The basic principles of autonomy and rationality are present in most definitions, but there is little further agreement. This can be a considerable source of confusion in attempting to apply agent techniques to the Web. The reason for this lack of consensus is because the term is used generically to refer to a heterogeneous body of research rather than a specific notion of agency. It can be argued that a precise definition of agency for the Semantic Web is unimportant, provided that the agents can cooperate in meaningful ways. In fact, it is probably unrealistic to insist on a single definition. In this book, we adapt and apply a variety of different definitions for our purposes.

The first challenge that we address is how to construct agents that are capable of autonomous and rational behaviour. In other words, how can we design a program that can decide on its own what needs to be done, and to do it, without explicitly being told what to do. This is rather different from a typical style of programming, where the computer performs only exactly what we instruct it to do. The inspiration for this style of programming comes from the study of *human reasoning* in philosophy. In essence, this is the reasoning directed towards actions and the process of determining what to do to achieve these actions. This differs from purely *logical reasoning*, e.g. all water is wet; rain is water; therefore rain is wet.

Human reasoning can be considered to comprise two activities:

1. We decide what state of affairs we want to achieve.
2. We decide how to achieve this state of affairs.

The first of these activities is *deliberation*, and the result is a list of intentions. The second is *means–ends reasoning*, and the result is a plan of action. We can illustrate this kind of reasoning by returning to our chocolate cake example. Through some deliberation on our current state of being, we decide to make a chocolate cake, and this becomes our intention. We then undertake means–ends reasoning on how this can be achieved. In our first stage of reasoning, we decide that we need to obtain a recipe, then obtain the ingredients for the recipe, and then follow the recipe to make

the cake. These decisions become our new intentions and we embark on further reasoning, for example, we decide to obtain a book of recipes, and this in turn requires money to purchase the book, and so on. Once the reasoning process is complete, we will have a plan of action that we can use to bake our chocolate cake.

Practical human reasoning can be expressed computationally by a number of different logic-based systems. The most popular of these systems is the Belief–Desire–Intention (BDI) model of Michael Bratman. The BDI model is said to take an *intentional stance* in determining the behaviour of the agent. This is a theory from philosophy, which makes predictions about the behaviour of an entity by treating it as a rational agent whose behaviour is governed by intentional states. Intentions are the driving force of the BDI model as they determine the actions of the agent. However, it is important to appreciate that intentions may have unexpected side effects. For example, I may intend to get to a meeting by the quickest route, but this does not mean that I want to swim across a river on the way.

In the BDI model, reasoning is defined by three mental states. *Beliefs* correspond to knowledge that the agent has about the domain, *desires* represent the state of affairs that the agent would (ideally) like to bring about, and *intentions* are desires that the agent has committed to achieving. In addition to these three states, we also have *goals*, which are a consistent subset of the desires as some desires may conflict, and *actions* which are derived from the intentions. Figure 1.3 illustrates the various components of the model.

The BDI model can be used to define and implement agents with rational behaviours. However, it is important to note that this is just one particular technique for defining rational agency. There are many alternative techniques that we can adopt, which may be more appropriate depending on the kinds of agents that we want to build, e.g. reactive agents, hybrid agents, planning agents, and theorem-proving agents.

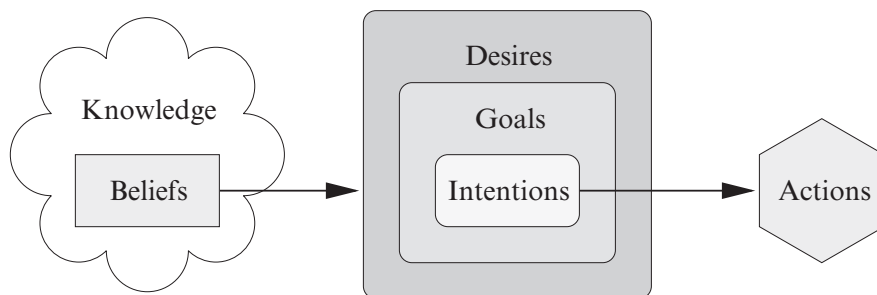


Figure 1.3 The BDI model.

1.2.1 Communication and societies

An individual agent is a useful entity that we can define to perform tasks on our behalf, given suitable reasoning and enactment mechanisms. For example, we can define an agent with the intention to obtain some useful piece of information, e.g. a list of banks in our area. The agent can then be let loose onto the Web and will attempt to find this information for us by utilizing a variety of knowledge sources, and performing inference on this knowledge.

The Semantic Web vision goes beyond the notion of agents acting in isolation, and views the agents acting together as a *society*. In a society, the individual agents interact closely and assume group behaviours such as cooperating, coordinating, negotiating, and so on. This is a common view of agency and is based on the idea of agents interacting in much the same way as humans interact on a daily basis. The term *Multiagent System* (MAS) is used to describe a system of agents that interact closely with a range of behaviours.

Embracing a societal view of agency introduces a range of new challenges that we must address. One of the most important is simply how to get the agents to talk together, since the agents cannot assume group behaviours if they cannot communicate. The issue goes beyond standards for communication, although such standards are a crucial first step. In addition, we need to communicate the meaning of our communication in such a way that it can be understood, e.g. to express our beliefs to another agent. The inspiration for this style of communication comes from the study of *human dialogue* in philosophy.

A popular basis for the definition of agent interaction is the theory of *speech acts* proposed by the philosopher John Austin and extended by John Searle. This theory recognizes that certain natural language utterances have the characteristics of actions. That is, they change the state of the world in a similar way to physical actions. For example, the act of moving an object changes the state of the world, as does the speech act of ‘declaring war’, to use a popular example. The theory identifies a class of *performative verbs*, which correspond to different types of speech acts. These verbs can be classified into five groups. *Assertives* commit the speaker to the truth of what is asserted, e.g. inform. *Commissives* commit the speaker to a course of action, e.g. promise. *Declaratives* effect some change on the state of affairs, e.g. declare war. *Directives* attempt to get the listener to do something, e.g. propose. Finally, *Expressives* express a mental state, e.g. prefer.

Speech acts are a popular basis for defining communication between agents in MASs. In this approach, the inter-agent communication is performed by exchanging messages containing performatives. Each message

has an associated performative that expresses the intended meaning of the message. To ensure compatibility between different agents, a number of standard Agent Communication Languages (ACLs) have been defined. These languages define a common set of performatives, and their precise meanings. The two dominant ACLs in MASs are the Knowledge Query and Manipulation Language (KQML), and more recently the Foundation for Intelligent Physical Agents-Agent Communication Language (FIPA-ACL). KQML defines a set of forty-one performatives, and FIPA-ACL defines a set of twenty-two performatives. Figure 1.4 illustrates an example FIPA-ACL message. The example is a message from `agent1` informing `agent2` that the price of `item` is 150. The message also specifies the content language of the message, in this case the FIPA Semantic Language (`s1`). Finally, the message specifies the ontology relevant to the communication.

In the speech acts approach, an interaction between agents will consist of the exchange of performatives over a time period. This process is akin to a *dialogue* between humans. The sequence of performatives that an agent uses will be focused on achieving a particular intention of the agent. The sequence can be quite complex and varied over time, as the agent may also cooperate with the intentions of other agents. Nonetheless, a sequence will have a definite pattern, dependent on what kind of goal the agent is attempting to satisfy. This sequence is called a *protocol*, and it is useful to consider these protocols in their own right, rather than simply a side effect of agent interaction. Treating the protocol separately is particularly useful when we consider large numbers of interacting agents.

Protocols for agents can be expressed using a finite-state representation that defines the legal sequences of performatives in a dialogue. This approach has been adopted in a number of different agent frameworks. Figure 1.5 defines an example protocol for a simple interaction between a doctor and a patient. The protocol begins with both agents in the INITIAL state. A patient agent then sends a request message to a doctor agent indicated by `request(P, D)`, where P is a patient and D is a doctor. This message is intended to represent the patient making an appointment to see a doctor. The patient then enters the WAIT state until an `accept(D, P)` message is received from the doctor. At this point the agent enters the

```
(inform
  :sender    agent1
  :receiver  agent2
  :content   (price item 150)
  :language  s1
  :ontology  english_auction)
```

Figure 1.4 Example FIPA-ACL message.

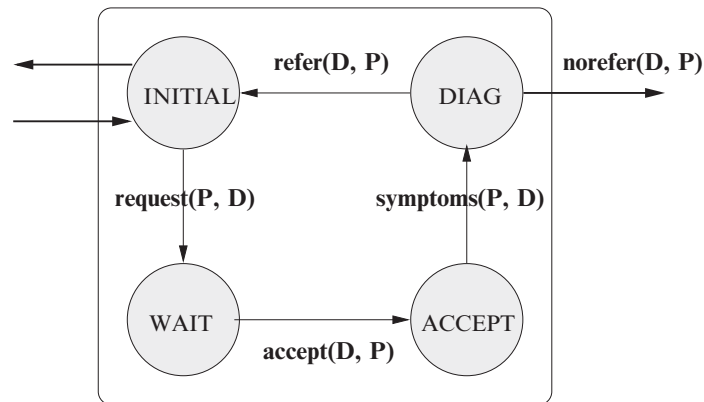


Figure 1.5 *Doctor and patient protocol.*

ACCEPT state and proceeds to send a message **symptoms(P, D)** to the doctor. The doctor then performs a diagnosis of the patient in the DIAG state and the result is that the agent is referred **refer(D, P)** for further diagnosis, or no-referral **norefer(D, P)** is made in which case the protocol terminates.

Agent protocols, or patterns of dialogue, also have a parallel in human dialogue. In any human interaction, there are always implicit rules or social norms at work. Some of these rules are more obvious than others. For example, in an auction house or in sport there are clearly defined rules. Other rules are more subtle and flexible, for example the interaction between a salesperson and a customer. Finally, in general-purpose conversation there are different ways in which we speak to business associates and close friends.

The different kinds of human dialogue have been classified into six main categories by the philosophers Douglas Walton and Erik Krabbe. Table 1.2 summarizes the different types of dialogue that they identify. The eristic type is essentially the breakdown of rational dialogue. The

Table 1.2 *Dialogue types.*

Dialogue type	Goal	Initial situation
Persuasion	Conflict resolution	Conflicting point of view
Negotiation	Making a deal	Conflict of interest
Deliberation	Reach a decision	Need for action
Information Seeking	Spreading knowledge	Personal ignorance
Enquiry	Growth of knowledge	General ignorance
Eristic	Humiliation	Antagonism

classification of dialogues into these categories can act as a useful guide in the constructing agent protocols. For example, we can readily define templates for the different dialogue types. However, this approach is a relatively recent development in agency, and a formal theory that relates dialogue types to speech acts is still being developed.

1.3 Knowledge services

We have now described the two main technologies that are at the heart of the Semantic Web vision: *ontologies* and *agents*. Ontologies are used to represent knowledge, and agents are used to reason about this knowledge. Both of these are founded on mature AI techniques that have been under development for many years. The difference in the Semantic Web is the use of these technologies together, on the Web, with a focus on the provision of specific applications. To unite these two approaches, a third technology has been adopted, called Knowledge Services or ‘Semantic Web’ services.

Knowledge services are the means by which computation is performed on the Semantic Web. A *service* is a software component that can be invoked by an external entity to perform some task. There are essentially two different kinds of knowledge services: those that *provide* knowledge, and those that *transform* knowledge. The first kind of service is used to obtain knowledge, e.g. to access a resource such as a knowledge base. The second kind of service performs computation on the knowledge, e.g. to perform a particular kind of inference on a collection of knowledge. In essence, the knowledge providers are a wrapper around ontologies, and knowledge transformers a wrapper around agents.

A range of example knowledge services are shown in Figure 1.6. The most common kind of knowledge transformation is realized by an inference engine such as the BDI reasoner we outlined previously. Another kind of knowledge transformation can be achieved by combining together services into a plan according to a predefined workflow of tasks. This kind of service is appropriate where the full power of agency is not required. Finally, a knowledge transformation may be simply a predefined operation, e.g. a lexical analysis, performed by a computation engine. The most common kind of knowledge provider is a knowledge base, i.e. a database of knowledge. Alternatively, we may be interested only in obtaining the ontology, and therefore we can consult an ontology store. Thus, we may obtain our knowledge from an ordinary website containing semantically annotated information.

Knowledge services are typically implemented by encapsulating them in *web services*. Web services are a programming technique that standardizes

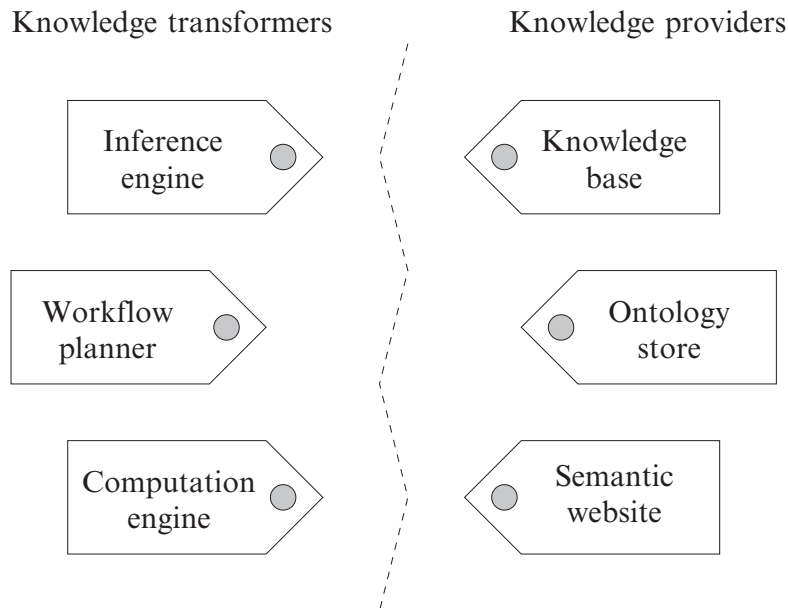


Figure 1.6 Knowledge services.

many aspects of distributed processing and communication on the Web. The appeal of web services over other interoperability standards, such as CORBA, is the simplicity and flexibility of the architecture. At the core of this technique there are just two XML-based standards that define the web services architecture: web services are specified in the Web Service Description Language (WSDL), and communication between web services is defined by the Simple Object Access Protocol (SOAP).

The use of a web services architecture is an important step towards the construction of a computer-processable Web. This architecture enables websites to be accessed through a standard mechanism, which is similar to procedure calls. Therefore, Web-based information can be retrieved in a suitable form for mechanized processing, rather than as an HTML-formatted document designed for human consumption. An increasingly large percentage of sites on the Web already allow their information to be accessed through a web service interface. Although this is still a long way from the full Semantic web vision, this alone is an important achievement as it provides a context for the Web-based information, and allows this information to be processed by external agents.

Semantic Web *applications* are constructed by composing together knowledge services. Consider our earlier example on the construction of a Semantic Web application to locate the cheapest camera with a specific

set of features. This application will require the composition of a variety of knowledge providers of pricing information, together with a number of additional knowledge transforming services that perform the comparison. These transformers may in turn require the composition of other services, e.g. to transform between different ontological representations of the knowledge, and to present the results in a suitable format for the end user.

The composition of knowledge services into Semantic Web applications is intended to be an on-the-fly process, which is performed dynamically by the agents performing the computation. If an agent needs a particular service to achieve one of its intentions, then this service will be composed by the agent into the application. However, this dynamic composition of services is a non-trivial task that is still being addressed by AI research. There are three main problems that must be solved, namely *discovery*, *invocation*, and *composition*.

Service discovery is the process of finding a suitable service for a particular task, and selecting from competing alternatives. We need to discover what the service does, and how to interact with the service. Service invocation is the process by which we execute a service. At this point we need to construct data of the appropriate form expected by the service, and interpret the results obtained from the service. Finally, service composition is the process by which we combine together services, where the task cannot be performed by a single service.

A possible approach to addressing the three issues that we have highlighted is to equip each service itself with ontological knowledge. This knowledge classifies the service by *profile*, *process model*, and *grounding*. The profile defines what the service does in order to facilitate discovery. The process model defines how the service works so that we can determine the choreography of the service. Finally, the grounding defines how to access the service such that invocation of the service is possible. There are currently two standard ontologies for classifying services in this manner: OWL Service (OWL-S) and Web Service Modelling Ontology (WSMO).

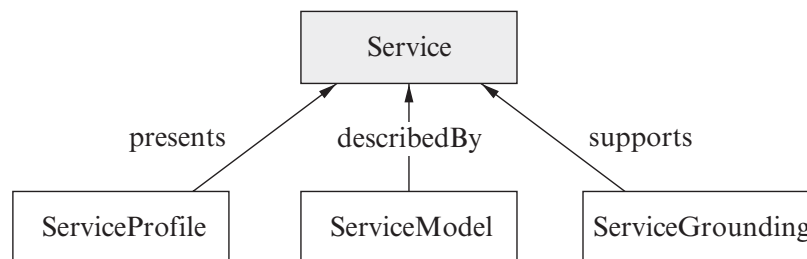


Figure 1.7 OWL-S top-level ontology.

Figure 1.7 illustrates graphically the top-level concepts and relationships in the OWL-S.

1.4 Book outline

This book is about the construction of the next generation of the Web, commonly called the Semantic Web. This Semantic Web will enable computers to automatically consume Web information, overcoming the human-centric focus of the Web as it stands at present. This in turn will expedite the construction of a whole new class of knowledge-based applications that will intelligently utilize Web content.

The construction of the Semantic Web will be a very difficult process. Nonetheless, there has been a decade of AI research into knowledge management and multiagent techniques that can be applied towards achieving this goal. We have presented a flavour of these techniques in this introductory chapter, and we expand on this material in the remainder of the book. Our presentation is structured naturally into four related themes, which correspond to the techniques that we have discussed:

1. Knowledge representation techniques for the Semantic Web: Chapter 2.
2. The construction of agents that can reason about knowledge: Chapters 3, 4, and 6.
3. Reasoning on the Semantic Web with agents: Chapter 5.
4. Knowledge Services for the Semantic Web: Chapter 7.

For each of these main themes, we present an overview of the state-of-the-art techniques, and the popular standards that have been defined. We are primarily interested in practical results that can be achieved using these technologies. For those who are interested in obtaining a deeper understanding, we also present the main theoretical results that underlie each of the technologies, and we summarize the main problems and research issues that remain. Our presentation is guided by the following two aims:

1. To educate the reader about the various Semantic Web techniques and technologies.
2. To give the reader a theoretical grounding in the field, so that they can perform research in the area.

We acknowledge that there will be some readers who are sceptical about the whole idea of the Semantic Web. As with many new Web technologies, there has been a lot of hype and many inflated claims on what can be achieved. It should be clear from our introduction that there are still many

significant issues that must be solved before the full Semantic Web vision can be achieved. Nonetheless, we claim that it is not necessary for all of the pieces to be in place before the benefits of the Semantic Web can be realized. For example, simply implementing a web service interface to a website enables a much greater degree of computer processing of Web information, and the construction of a single agent can perform many useful tasks, without requiring the existence of a community of agents. Our opinion is that the construction of the Semantic Web will happen in a piecemeal manner, and will be driven by the applications. This is evident as the most popular uses of ontologies on the Web at present are Really Simple Syndication (RSS) feeds, and Friend-of-a-Friend (FOAF) documents.

1.5 Suggested reading

1. G. Antoniou and F. van Harmelen. *A Semantic Web Primer*. MIT Press, 2004.
2. T. Berners-Lee and M. Fischetti. *Weaving the Web*. Harper, 1999.
3. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.
4. J. Davies. *Towards the Semantic Web: Ontology-Driven Knowledge Management*. Wiley, 2003.
5. D. Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag, 2001.
6. D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, 2003.
7. A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web*. Springer-Verlag, 2004.
8. T. R. Gruber. *Towards Principles for the Design of Ontologies Used for Knowledge Sharing*. Kluwer, 1993.
9. J. Hendler. Is There an Intelligent Agent in Your Future? *Nature—Web Matters*, (3), March 1999.
10. J. Hendler. Agents on the Web. *IEEE Intelligent Systems*, 16(2): 30–7, March 2001.
11. G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde, and B. Wielinga. *Knowledge Engineering and Management: The CommonKADS Methodology*. MIT Press, 1999.
12. K. Sycara and M. Paolucci. Ontologies in Agent Architectures. In *Handbook on Ontologies in Information Systems*. Springer-Verlag, 2004.
13. M. Uschold and M. Gruninger. Ontologies: Principles, Methods, and Applications. *Knowledge Engineering Review*, 11(2): 93–155, June 1996.