KNOWLEDGE REPRESENTATION

João Pavão Martins
Instituto Superior Técnico
Universidade Técnica de Lisboa

João Pavão Martins
Grupo de Inteligência Artificial
Departamento de Engeharia Mecânica
Instituto Superior Técnico
Av. Rovisco Pais
1096 Lisboa CODEX
Portugal

Contents

Preface

This book corresponds to a partial compilation of the notes that have been used in the course on Knowledge Representation in the Technical University of Lisbon (Portugal). It should be considered as a rough draft to a forthcoming book on Knowledge Representation. Comments and suggestions about its contents will be most welcome.

The book is organized into four parts: Part 1 (Chapter 1) provides an overall overview on the field of Knowledge Representation; Part 2 (Chapters 2, 3, 4, and 5) provide the logical foundations to understand and evaluate different Knowledge representation formalisms); Part 3 (Chapters 6, 7, 8, and 9) presents the characteristics of the main representation formalisms, discusses, in detail, one formalism of each type, and presents evaluation criteria for a knowledge representation system); Part 4 (Chapters 10 and 11) discuss issues on knowledge sharing and ontologies.

Chapter 1

Basic Ideas

1. The notion of knowledge
2. Cognitive acting
3. Underlying hypotheses
4. Levels of approach
5. Notational and inferential aspects
6. Declarative vs. procedural controversy
7. Aspects to consider
8. Summary
9. Recommended reading
10. Exercises

In this chapter we discuss the principles and the basic assumptions underlying the field of Knowledge Representation. We discuss, in general, the importance of Knowledge Representation in Artificial Intelligence and the problems it addresses.

Knowledge comes, but wisdom lingers.
Lord Alfred Tennyson, *Locksley Hall*

Artificial Intelligence (AI) is the branch of Computer Science that is engaged with the development of programs that present intelligent behavior (the kind of behavior found in Humans). Since this behavior is largely dictated by the knowledge that the agents that exhibits it contains, it is just natural that one important area of AI addresses the problem of how to represent and use knowledge. The is no doubt that the actual development of AI is largely due to the concept of computing system capable of using a portion of explicitly represented knowledge to reason about its goals, about its environment, about other agents and even about itself.[1]

There are several reasons to justify the importance of knowledge representation in AI. First, when we consider the way humans perform tasks such as playing chess, planning activities or solving mathematical equations, we usually bear in mind what they know in order to perform that task. In other works, knowledge is associated with intelligent behavior. Second, for several years now, it has been recognized that in order to obtain intelligent behavior in a program, it is more important to have large amounts of knowledge that to have sophisticated algorithms. Despite the fact that there are problems whose solution does not require large amounts of knowledge, such as certain games, whose solution can be obtained by searching through the situations that can be obtained from a given situation, the solution for most of the problems that we face in our daily life depends, almost exclusively, from our knowledge about the application domain. For example the success of medical diagnosis depends on the knowledge that the doctor has about the different illnesses and their symptoms, the types of medication to use, and their side effects, and even about the knowledge that the doctor has about the psychology of the patient.

The idea behind Knowledge Representation is very simple, it consists in writing, using a specific language, sentences or figures that correspond to the description of a world or the state of a world. When a computer manipulates this representation it reaches conclusions or predictions about the world or its state (Figure 1.1[2]).

A *knowledge representation formalism* can be defined as a collection of data structures and procedures that manipulate these data structures. When the procedures and the data structures are used in a proper way, the program that uses them reflects intelligent behavior. We should note that data structures are knowledge in the same sense that an encyclopedia is knowledge: it is a source of knowledge, but, without a reader, it is only ink over paper. Knowledge

[1] This conclusion was reached after the unsuccessful efforts of developing general mechanisms for intelligent behavior, for example, the GPS system [Ernst and Newell 69].

[2] Reprinted with kind permission of John Rickwood.

Figure 1.1: Knowledge Representation in AI.

Representation is closely related with reasoning since it needs to make explicit things that are implicit.

Many AI systems represent knowledge in a separate component, called the *knowledge base*. The knowledge base is independent from the program that manipulates it, which is called the *inference engine*. The behavior of these systems is dictated by the by the use of the knowledge, in the knowledge base, by the inference engine, and they are called *knowledge-based systems*. However, not every program that reflects intelligent behavior is a knowledge-based system. For example, a chess player program, decides its moves, not based on a symbol structure were knowledge is represented but rather is a search procedure based on heuristic functions.

Knowledge Representation has links to many other scientific areas besides AI: psychology (in what respects the study of psychological models of memory and reasoning); neurology (in what respects the study of physiological models of human's nerve system); linguistics (in what respects the study of the representation of the form, semantics and pragmatics of language); philosophy (in what respects the study of the nature of concepts and propositions); logic (in what respects the study of deductive reasoning); and computer science (in what respects the study of information systems and programming languages).

When we address the area of Knowledge Representation there are several questions that pop up and whose answer we address in this first chapter: What is knowledge? How does knowledge influence intelligent behavior? Are there assumptions that support the field of Knowledge Representation? At what level should we consider the analysis of knowledge? What are the main concerns of Knowledge Representation?

1.1 The notion of knowledge

The attempt to define what knowledge is, is very old. Both Plato (427-347 b.c.) and Aristotle (384-322 b.c.) have addressed this question, and there is a long philosophic tradition on discussions on knowledge-related questions, what is knowledge, what is its nature and origin, how can we classify it, what connections does it have with the notion of truth.

When we try do define the meaning of "having knowledge" we should analyze, among others, the meanings of "knowledge" and of the verb "to know". If we look in a dictionary [Hawkins and Allen 81] we obtain:

know *v.& n.* $-$*v.* **1** *tr.* (often foll. by *that, how, what,* etc.) **a** have in the mind; have learn; be able to recall (*knows a lot about cars; knows what to do*). **b** (also *absol.*) be aware of (a fact) (*he knows I am waiting; I think he knows*). **c** have a good command of (a subject or language) (*knew German; knows his tables*). **2** *tr.* be acquainted or friendly with (a person or thing). **3** *tr.* **a** recognize; identify (*I knew him at once; knew him for an American*). **b** (foll. by *to* + infin.) be aware of (a person or thing) as being or doing what is specified (*knew them to be rogues*). **c** (foll. by *from*) be able to distinguish (one from another) (*did not know him from Adam*). **4** *tr.* be subject to (*her joy knew no bounds*). **5** *tr.* have personal experience of (fear, etc.). **6** *tr.* (as **known** *adj.*) **a** publicly acknowledge (*a known thief; a known fact*). **b** *Math.* (of a quantity etc.) having a value that can be stated. **7** *intr.* have understanding or knowledge. **8** *tr. archaic* have sexual intercourse with. $-n.$ (in phr. **in the know**) *colloq.* well-informed; having special knowledge.

knowledge *n.* **1 a** (usu. foll. by *of*) awareness or familiarity gained by experience (of a person, a fact, or thing) (*have no knowledge of that*). **b** a person's range of information (*is not within his knowledge*). **2 a** (usu. foll. by *of*) a theoretical or practical understanding of a subject, language, etc. (*has a good knowledge of Greek*). **b** the sum of what is known (*every branch of knowledge*). **3** *Philos.* true, justified belief; certain understanding, as opp. to opinion. **4** = *carnal knowledge.*

As can be seen, "knowing" and "knowledge" may have many different meanings. In order to understand the differences between these different meanings, let us consider the following sentences (adapted from [Lehrer 80]):

1. I know how to go from Buffalo to Toronto.
2. I know how to type.
3. I know the expansion of π with four decimal places.
4. I know Bill.
5. I know Hamlet.
6. I know Denmark.
7. I know that the area of a rectangle is given by the product of its length by its height.
8. I know that you are telling the truth.

One of the meanings of "knowing" (illustrated by sentences 1, 2 and 3) corresponds to having a certain kind of competence. Under this meaning, we state that we know how to go from one place to another we are conveying competence about being able to move ourselves or to explain to someone how to do it. When we state that we know how to use a device, we are stating our competence about its use. When we state that we know something (by heart) we are stating that we have the competence to recall and convey that information.

Another meaning of "knowing" (illustrated by sentences 4, 5 and 6), corresponds to being acquainted with someone or with something. The meaning of sentence 6 is harder to precise since it may mean that I have been to Denmark or that I am able to more around geographically or socially. This sentence

Figure 1.2: Acting cycle of a cognitive agent.

may correspond to two different sense of the verb "to know", competence and acquaintance.

A third meaning of "knowing" (represented by sentences 7 and 8) corresponds to recognizing something as information. This is the meaning that is more interesting to us. Usually knowing something, according to one of the other meanings, also entails having knowledge according to this meaning of the verb. In this book we will mainly consider knowledge under this interpretation since its is the most closely connected with cognition.

In philosophy, there is a distinction between the words "knowledge" and "belief". In AI we follow a more permissive approach and we associate the word "knowledge" with both these senses. So, we roughly can say that, in AI, knowledge is a collection of facts and relations among them such that, when these are interpreted in an adequate way, they give rise to intelligent or to rational behavior. The amount of knowledge that an agent (either natural or artificial) possesses can be measured by the variety of situations in which the agent is able to achieve successful results.

1.2 Cognitive acting

If we observe the behavior of a cognitive agent (agent, either natural or artificial, whose behavior is based on knowledge), we can explain its behavior in the following way:[3] (1) the agent has knowledge about the surrounding environment and about the way this environment is affected by its actions; (2) the agent has a certain number of goals, that also correspond to knowledge; (3) the actions performed by the agent are based on its knowledge and are executed in order to achieve some of its goals.

Under this perspective, the activity of a cognitive agent mainly consists in perceiving, trough its sensing devices, the world that surrounds it, in reasoning about that world, and in making decisions about what actions to execute. These actions are dictated by the knowledge of the agent, by its goals, and by the current perception of the outside world. This means that a cognitive agent is involved in an infinite loop of perceiving, reasoning, and acting (Figure 1.2).

This acting cycle, requires the agent to acquire an internal representation that corresponds to a model of the external world, representation that should be kept up-to-date with the current state of the world. An avoidable characteristic of this model of the world is that it is, in the more optimistic case, an incomplete model. It may also contain errors due to false assumptions and invalid reasoning.

[3] This characterization was proposed by [Newell 80].

Figure 1.3: Model of the real world.

According to [Woods 87], the differences between the model and the real world may be due to:

1. Changes that happened in the world since the agent recorded information about it (the world is not static);

2. The lack of ability, from the agent, to learn, within a reasonable amount of time, the things that it should know (there are too many things to learn);

3. The limitations of the Knowledge Representation system that may not allow the conceptualization of certain aspects of the world (the world is richer than we can imagine).

This description applies both to computing systems and to human beings.

It is widely recognized that the real world is far too complex and detailed to be full perceived or understood by a cognitive agent. This fact leads to the creation of models of the world (through the use of abstraction), where simplification removes irrelevant aspects. This perspective is used by engineers with approximate models that disregard unnecessary aspects of the world (points, masses, frictionless surfaces, and so on). The partial agreement of these models with the real world is balanced by the ease and efficiency of its use. Their usefulness corresponds to their capability of predicting the behavior of the world. In other words, using these models, we can find out what the state of the real world will be (in respect to the aspects existing in the model) after the execution of a certain number of actions, without the need to perform those actions in the real world.

In a knowledge-based system, the model of the world is represented in a *knowledge base*, that is manipulated by general-purpose procedures, the *inference engine*. The knowledge base corresponds to a representation of a model of the real world (where the aspects irrelevant to the problem at hand are left out (Figure 1.3). In this sense, the knowledge base corresponds to an internal representation of the outside world. This internal representation is updated with the results of perception, and is used (resorting to reasoning) to compute the action to be taken and to predict the effect that the action will have on the external world. In case the action is carried out in the world, the predicted results are compared with the actual results.

1.3 Underlying hypotheses

Before we address the representation of knowledge, we should look into the underlying assumptions of the field. These assumptions correspond to the two hypotheses discussed in this section.

1.3.1 Physical symbol system hypothesis

The first hypothesis related to Knowledge Representation, the physical symbol system hypothesis, was stated by [Newell and Simon 76] and is discussed in detail in [Newell 80].

A physical symbol system is composed by a set of entities, called *symbols*, that correspond to physical patterns. Symbols can be grouped into symbol structures or *expressions*. At any moment, the physical symbol system contains a collection of these symbol structures. Besides the symbol structures, the physical symbol system has a collection of procedures that operate on symbol structures, producing symbol structures (these procedures create, modify, reproduce, and destroy symbol structures).

A *physical symbol system* is a machine that, over time, produces a succession of these symbol structures. The system exists in a world containing other objects besides the symbol structures. We should note that this description is so generic that it can be applied to any computer program.

The *physical symbol system hypothesis* states that a physical symbol system is a necessary and sufficient condition for intelligent behavior. By *necessary* condition, is meant that any system that shows intelligent behavior can be viewed as a physical symbol system; by *sufficient* condition, is meant that any physical symbol system of adequate size can be organized so that it presents intelligent behavior; finally, by intelligent behavior is meant the kind of behavior found in Humans.

The importance of this hypothesis to AI stems from the fact that it states that it is possible to write a computer program (nothing more than a physical symbol system) that presents intelligent behavior. This hypothesis makes no commitment about what is the "knowledge" of this program nor to the way the knowledge is represented.

The physical symbol system hypothesis means that we will be interested in:

1. Finding out symbol patterns that correspond to important aspects of the domain being considered;

2. Finding out the operations that manipulate these symbol structures to produce potential solutions to our problems.

These aspects stress the need for Knowledge Representation and reasoning.

1.3.2 Knowledge representation hypothesis

Over the years, there has been many disputes between researchers in knowledge representation. There is, however, some agreement about fundamental questions. One of the points of agreement is called *knowledge representation hypothesis*, formulated by [Smith 82], which states that:

1. Any mechanical process that presents intelligent behavior is composed of structures that are assigned, by an external observer, to a propositional representation of the knowledge used by that process.

2. Independently of the meaning that external agents may assign to these propositional structures, they play a formal but causal and essential role in the behavior that manifests that knowledge.

The fact that these structures are propositional in nature and could be graphs, trees, arrays, records, or any other data structure, means that this hypotheses spans over a large scope of knowledge representation formalisms, thus the reason of agreement! These structures are stored in a component called the *knowledge base*.

The knowledge representation hypothesis complements the physical symbol system hypothesis. Whereas the physical symbol system hypothesis states that a physical symbol system is capable of generating intelligent behavior, the knowledge representation hypothesis assigns that behavior to the structures that represent the knowledge of the system.

In order to enable the existence of an explicit knowledge base, there should be a well defined language that will be used to represent the knowledge of the system, the *knowledge representation language*. The system must also have a set of procedures, the *inference engine* that enable the generation of information that is implicit in the knowledge base.

The knowledge about the application domain contains the necessary power to create intelligent behavior. This body of knowledge is explicitly represented in the knowledge base, using the knowledge representation language, and it is the responsible for intelligent behavior.

1.4 Levels of approach

Knowledge Representation can be approached at two different levels, known as the knowledge level and the symbol level. This distinction was introduced by [Newell 80] and is similar to the distinction between the definition of an abstract data type and the choice of representation for the elements of that type.

1.4.1 The knowledge level

The knowledge level looks at the knowledge base as an abstract data type, that can be accessed and manipulated through a small number of operations. These operations, include a constructor, called *tell*, that enables the addition of new knowledge to the knowledge base and a selector, called *ask*, that enables to query the knowledge base.

At this level, we are not concerned with the structures and mechanisms that will be used to maintain the evolving model of the domain under consideration.

The capability of the knowledge base is measured in terms of the functions that use it: the statements that it is able to accept and the answers that it is capable to produce.

According to [Newell 80], one of the main characteristics of knowledge is the notion of *competence*. This notion is applicable to an agent whose behavior is dictated by the knowledge existing in a knowledge base and can be considered as an abstraction (our heuristic) to reason about the beliefs of that agent. This notion can be formalized in the following way: if the agent imagines the world in such a way that α is true, and if α implies β, then (disregarding whether or not, the agent is aware of it), the agent imagines the world in such a way that β is also true. In this way, the knowledge level analysis corresponds to the study of what is implicit in the beliefs of the agent.

This analysis requires the definition of the operations *tell* and *ask*. These operations may be defined as follows:

$$tell \ : \ knowledge \times statement \Rightarrow knowledge$$

$$ask \ : \ knowledge \times question \Rightarrow answer$$

Clearly, these two operations are not enough to a complete approach at the knowledge level, among others, we will need operations to create an empty knowledge base, an operation to simulate forgetting, an operation to remove contradictions, and so on. However, since *tell* and *ask* correspond to the main interaction between a program (or user) and a knowledge base, these correspond to the operations that have been discussed in deeper detail.

In order to define these operations in detail, we have to chose a language, say $\mathcal{L}$, to represent statements about the world and with which we can query the knowledge base. A possible language, used by [Levesque 84], corresponds to the language of first order logic. We will use $[\mathcal{L}]$ to denote the set of closed formulas (that is without free variables) of $\mathcal{L}$. Let KB be a knowledge base, a set of formulas of $[\mathcal{L}]$. The operations *tell ask* may be defined in the following way:

$$tell(\Delta, \alpha) = \Delta \cup \{\alpha\}$$

$$ask(\Delta, \beta) = \begin{cases} \text{yes} & if \ \Delta \vdash \beta \\ \text{no} & if \ \Delta \vdash \neg\beta \\ s & if \ \exists s \ : \ \Delta \vdash \beta \circ s \\ \text{unknown} & otherwise \end{cases}$$

In the previous description, $\Delta \subset KB$, $\alpha \in [\mathcal{L}]$, $\beta \in \mathcal{L}$ and s is a substitution.

One of the important decisions to take at this level is the choice of the language $\mathcal{L}$. In [Newell 80] it is argued that logic is an essential tool for the analysis at the knowledge level. [Levesque 84] looks into the choice of first order logic as the language for $\mathcal{L}$ and suggests an extension with higher expressive power, that enables the explicit distinction between believing and knowing a given proposition.

The analysis at the knowledge level enables to establish the theoretical foundations that enable an evaluation and comparison of the different knowledge representation formalisms. This is done because it concentrates on the services supplied by a knowledge representation system, independently how that service is realized.

1.4.2 The symbol level

In Computer Science, a good solution depends, most of the times, on a good representation. It is therefore important to have an adequate representation for the states of knowledge and reasoning of a cognitive agent. The researchers in AI soon recognized that the primitives offered by a knowledge representation system, together with the mechanisms available to manipulate this knowledge, strongly affect the capabilities of percepting, knowing, and acting of the system.

The symbol level looks at the task of knowledge representation under the perspective of a certain representational formalism. When we approach the representation at the symbol level, we are interested in defining symbol structures that enable the representation of knowledge and in defining procedures that perform inference based on those structures. Most of the work in knowledge representation has been done at this level and most of this book will be devoted to knowledge representation at the symbol level. In particular, we study the four main formalisms for knowledge representation, semantic networks, frames, hybrid systems and the representation based on logic.

1.5 Notational and inferential aspects

Any knowledge representation formalism presents two different aspects: the notational aspect and the inferential aspect.

The *notational aspect* is concerned with how knowledge is represented. In particular, it is concerned with the development of a precise formal language that specifies what basic structures are used to represent knowledge and how these basic structures can be combined.

The choice of a formalism, according with the notational aspect, will depend on the ease of the representation of important aspects to be modeled by the language, with the expressive power of the language, and with the possibility of supplying an interpretation for the sentences of the language.

The *inferential aspect* is concerned with how to produce inferences from the static structures of the language, that is how to make explicit the knowledge that is implicit. It is thus concerned with the choice of inference rules to be included in the representation and the choice of what kind of control to perform on the inference.

1.6 Declarative vs. procedural controversy

There are two distinct ways of representing knowledge: using declarations (the declarative representation) or using procedures (the procedural representation). For many years, researchers have argued which of these ways should be adopted to represent knowledge. This dispute is known as the "declarative vs. procedural controversy" [Winograd 75]. The dispute corresponds to the philosophic distinction between *knowing what* and *knowing how.*

The procedural position was that knowing is "knowing how" and therefore, knowledge should be represented as a set of procedures, each one representing a fragment of knowledge. Between the researchers that defended this position, were [Winograd 72] and [Hewitt, Bishop, and Steiger 73].

The declarative position was that knowing is "knowing what" and therefor, knowledge should be represented as a set of statements, describing a particular domain. These statements will be manipulated by a general-purpose procedure that will be responsible for the inference of new statements. Between the researchers that defended this position, were [McCarthy 68] and [McCarthy and Hayes 69].

From the formal point of view, there is no distinction between these two positions. We know that programs may be looked at as data, and, according to this perspective, everything is declarative. On the other hand, we may look at declarations as statements for a program, under this perspective, everything is procedural.

The controversy is not concerned with the discussion of whether a given piece of knowledge corresponds to a statement or to a program but rather in finding out what do we gain if we look at it in one way or another. Clearly, each one of these types of representation has advantages and disadvantages. Typically, the advantages in one representation are disadvantages in the other and vice-versa.

1.6.1 Advantages of the declarative representation

The first advantage that can be pointed out to the declarative position corresponds to the *flexibility* and *economy* of the representation. The procedural position requires that when we specify knowledge we have to state how it is going to be used. Normally, there is more than one way of using that knowledge and it is not reasonable that every possible way of using that knowledge should be specified beforehand.

Consider, for example, the statement that "all lawyers from Chicago are rich". Using a language for procedural representation, PLANNER [Hewitt 69], this statement would be represented as:

```
(DEFTHEOREM THEOREM1
   (THCONSE (X) (#RICH $?X)
                (THAND (THGOAL (#PROFESSION $?X :LAWYER))
                       (THGOAL (#LIVESIN $?X :CHICAGO))
                )))
```

This representation should be read in the following way: if we want to find out whether `X` is rich (the wealth of `X` is a consequence that we want to prove, in PLANNER, `THCONSE`) then, we will have to simultaneously verify (in PLANNER, `THAND`) that `X` is a Lawyer and that `X` lives in Chicago.

Using a declarative representation, for example, first order logic, this piece of knowledge could be represented by the formula:

$$\forall(x)[Profession(x, Lawyer) \wedge LivesIn(x, Chicago) \rightarrow Rich(x)].$$

This statement could be used, for example, to:

1. To prove that David is rich, by showing that he is a Lawyer that he lives in Chicago;

2. To prove that Stuart does not live in Chicago, knowing that he is a lawyer and that he is not rich.

In a procedural representation this fragment of knowledge had to be represented in different ways in order to enable each one of these uses. Under this perspective, the declarative representation is more flexible and economical than the procedural representation, since the same "representation" can be used in different ways.

The second advantage that can be pointed out to the declarative representation concerns the *understanding* of the representation and the possibility of *learning*. In fact, the declarative representation is preferable in what respects the possibility of understanding what is represented and to the possibility of modifying that knowledge.

Since the declarative representation means that knowledge is represented as a set of statements, the addition, replacement or removal of knowledge is obtained through the addition, replacement or removal of statements. For this reason, learning is eased in a declarative representation. In a procedural representation a piece of knowledge may be embedded in a complex procedure, therefore, the addition, replacement or removal of knowledge is obtained through the modification of that procedure. Everyone that has ever programmed knows how the change of a procedure may have consequences on other parts of the program.

The final advantage to be pointed out to the declarative position corresponds to the *ease of access* and the *ease of communication* of the representation. Most of the things that we know are expressed as declarations: natural language is mainly declarative and corresponds to the easiest way to convey information. This aspect has direct consequence in the addition of knowledge to a program and to the communication of its contents to others.

1.6.2 Advantages of the procedural representation

The first advantage to be pointed out to the procedural representation is related to *procedural modeling*. Many things that we know can be looked at as procedures and it is difficult to describe them through declarations. Consider, for example, the addition of two integer numbers. Without a procedure to compute the addition, we would need a list of every pair of integers and their corresponding sum. In this case, we replace flexibility by an enormous gain in computation.

The second advantage of the procedural representation is related with the use of *meta-knowledge*. Certain aspects of knowledge about knowledge are more easily expressed by procedures rather than declarations. For example, the nearness relation is transitive, provided that it is not applied many times in the same deduction.

Finally, procedural representation has advantages in what concerns the *computational efficiency*. Using specialized procedures gives rise to more efficient programs than the use of general-purpose procedures.

1.6.3 Result of the controversy

In the later 70s, this controversy had "dissolved" rather than solved. In fact, people reached the conclusion that both kinds of representation were important. Most of the knowledge used nowadays in AI programs is declarative, therefore in most representations, there is the possibility of calling specialized procedures, when the use of procedures is required. This possibility of calling procedures in a declarative representation is called *procedural attachment*.

1.7 Aspects to consider

The underlying theme in the field of Knowledge Representation corresponds to the study of how to include knowledge about the world in a computing system so that, given the required reasoning capabilities, the system is able to use that knowledge to adapt itself and deal with that world. The main problems that the field of Knowledge Representation has to face are the following:

1. In what form should knowledge be represented? This problem deals with the study of structures to represent the different kinds of knowledge;

2. How can the reasoning mechanism use a limited amount of knowledge (explicit knowledge) to produce the remaining knowledge that is necessary for the task (implicit knowledge)?

3. How may knowledge (both implicit and explicit knowledge) be used to guide the behavior of the system?

4. What can the system do in face of incomplete or wrong information?

5. How can the system reason in face of problems that are computational difficult or undecidable?

Knowledge Representation has focused in different ways to express (or represent) information, ranging from informal models of human memory to complex formal languages. Different systems may be more suited to solved certain kinds of problems. However, the main effort in knowledge representation has been focused in the development of general-purpose languages for knowledge representation.

Since the development of a representation is of no use without considering the reasoning that can be performed with that representation, Knowledge Representation is actually taken as synonymous of knowledge representation and reasoning. For this reason, knowledge representation is grounded in the study of logics (not only first order logic, but logics in general), in which the study of the syntax is done together with the study of inference rules and interpretations. This supplies a standard for measuring the performance of a computing system for knowledge representation.

The main areas that the field of Knowledge Representation addresses are:

1. *Representation of knowledge*, that concerns the generation of theories about particular domains. recently, work has been done towards the development of generic ontologies, axiomatizations that span several domains, development of re-usable knowledge bases, and aspects associated with knowledge elicitation and knowledge engineering;

2. *Foundational mathematics*, that corresponds to the development of complex logics and methods for comparing them;

3. *Cognitive science*, the study of the structure of human thought and its parameters;

4. *Reasoning*, the study of the different forms of common-sense reasoning;

5. *Technology*, the development of computing systems to support the needs of the fields of AI.

1.8 Summary

In this chapter we presented the principles and underlying assumptions of the field of Knowledge Representation. We discussed, in general, the importance of Knowledge Representation for AI and the main problems addressed by researchers in Knowledge Representation.

We discussed the two hypotheses that underlie the work on knowledge representation, the physical symbol systems hypothesis and the knowledge representation hypothesis.

We presented the two levels at which knowledge representation can be approached, the knowledge level and the symbol level.

We discussed the declarative vs. procedural controversy, explained how this controversy was dissolved and explained how most modern representation systems use both declarative and procedural knowledge.

1.9 Suggested readings

Important reference works in the field of Knowledge Representation include [Bobrow and Collins 75], [Brachman and Smith 80], [Brachman and Levesque 85] and [Cercone and McCalla 87].

The proceedings of the conference on the principles of knowledge representation and reasoning [Brachman, Levesque, and Reiter 89], [Allen, Fikes, and Sandewall 91], [Nebel, Rich, and Swartout 92], [Doyle, Sandewall, and Torasso 94] present an excellent perspective of the problems facing the researchers of Knowledge Representation.

The following texts discuss at a deeper level, than we have done in this book, some of the aspects discussed in this chapter:

[Audi 88]
An introduction to epistemology, seen as a theory of knowledge and justification. This book stresses on the set of concepts, theories, and problems that are fundamental for the understanding of the notions of knowledge and justification.

[Brachman and Levesque 85]
Presents a collection of the most influential papers published in knowledge representation prior to 1984, as well as an annotated reference list of papers in this area.

[Brachman 90]
A paper that presents a perspective of the main problems addressed in the field of Knowledge Representation as well as a discussion on the historical evolution of the field.

[Lehrer 90]
Discusses the theory of knowledge from a philosophical perspective.

[Levesque 84]
Presents a knowledge-level approach to Knowledge Representation and discusses the aspects that a language used to interact with a knowledge base should have. It argues about the needs of a language with higher expressive power than first order logic.

[Winograd 75]
Presents an excellent discussion on the declarative vs. procedural controversy. The discussion presented in Section 1.6 was based on this paper.

[Woods 87]
Presents a good discussion on the acting cycle of a cognitive agent.

Bill
I will add references about knowledge in Philosophy
Also the overall text will be improved and extended.

1.10 Exercises

1. One of the underlying hypotheses to Knowledge Representation is the physical symbol system hypothesis. Explain this hypothesis, in detail, and discuss its importance to the field of Knowledge Representation.

2. One of the underlying hypotheses to Knowledge Representation is the knowledge representation hypothesis. Explain this hypothesis, in detail, and discuss its importance to the field of Knowledge Representation.

3. The declarative vs. procedural controversy corresponds to a disagreement between researchers in AI.

   (a) Explain what was the source of the controversy.

   (b) Explain how it was solved.

Chapter 2

Classical Logic

1. Components of a logic
2. The language of logic
3. Deductive system
4. Semantics
5. Soundness and completeness
6. Knowledge representation using logic
7. Summary
8. Suggested readings
9. Exercises

In this chapter, we define basic concepts of classical logic, discussing both propositional logic and first-order logic. We discuss the components of a logic, the deductive system and the semantics, and the relationships between them. We discuss the role of Logic in Knowledge Representation and the advantages and drawbacks of using Logic as a knowledge representation formalism.

> Logic is invincible because in order to combat logic it is necessary to use logic
> Pierre Boutroux [Kline 72]

A characteristic of rational agents is the need of justifying the things in which they believe and the need of understanding the way other agents justify their beliefs. One of the main goals of Logic is to start from a situation described by sentences, that are assumed to be true, and to find out what other sentences have to be true in that situation. The sentences that we start from are called the *premises* and the new sentences that are generated from them are called the *conclusions*. The pair premises/conclusion is called an *argument* and the process of generating conclusions from the premises is called *inference*. In our daily life we use some streches of discourse intended as supporting a conclusion from premises, these streches of discourse are used both to support the things in which we believe or to justify our actions.

Although it is not easy to define in a few sentences what Logic is, since it addresses a large number of problems and does not have well defined borders (in one of the ends of the spectrum it addresses philosophical problems and on the other end it addresses mathematical problems), for the sake of this chapter, we can say that it addresses the analysis of arguments, or the analysis of the methods to distinguish valid from invalid arguments [Corcoran 72]. An *argument* is a pair (Δ, α), where Δ (the *premises*) is a set of propositions and α (the *conclusion*) is a single proposition. A *proposition* is a sentence that can be assigned the truth value "true" or "false".

An argument may either be valid or invalid. The argument (Δ, α) is said to be *valid* (equivalently, we can say that Δ logically implies α, or that α is a *logical consequence* of Δ) when it is logically impossible for the premises to be all true and the conclusion to be false. Otherwise, the argument is said to be *invalid*. We should clearly distinguish the notions of validity/invalidity from the notions of truth/falsity. Validity and invalidity are attributes of arguments, whereas truth and falsity are attributes of propositions. An argument may have false premises and a true conclusion and be valid or may have true promises and a true conclusion and be invalid. The only relation between the validity or invalidity of an argument and the truth values of its components is the following: an argument whose premises are all true and whose conclusion is false, is invalid. The "almost" independence between the validity/invalidity of an argument and the truth/falsity of the sentences that are part of it, is expressed by the *principle of the truth-value irrelevancy*: except in the case where all the premises are true and the conclusion is false, the truth value of the propositions that are part of an argument does not determine whether the argument is valid or invalid.

In natural language, it is common to separate premises from the conclusion by words such as "thus", "then", "therefore", "consequently", and so on. Let us consider the following examples of arguments:

A1: Tweety is a bird
No bird has fins
Therefore, Tweety has no fins

This argument corresponds to the pair ({Tweety is a bird, No bird has fins}, Tweety has no fins).

A2: Silvester is an animal
Every dog is an animal
Silvester is not a dog
Therefore, not every animal is a dog

This argument corresponds to the pair ({Silvester is an animal, Every dog is an animal, Silvester is not a dog}, Not every animal is a dog).

A3: FOO is a LISP atom
No LISP atom has a space
Therefore FOO does not have a space

This argument corresponds to the pair ({FOO is a LISP atom, No LISP atom has a space}, FOO does not have a space).

A4: Every person is a human
Therefore, every human is a person

This argument corresponds to the pair ({Every person is a human}, Every human is a person).

Logic aims at defining methods to distinguish valid from invalid arguments. This distinction should be made independently from the subject matter dealt by the arguments, otherwise, there will be no independent subject called Logic. In order to reach this goal, in Logic, we consider classes composed by similar arguments that differ only in what concerns the specific terms of the subject matter of the argument. For example, arguments A1 and A3, above, are similar except that A1 deals with Zoology and A3 deals with Computer Science. In Logic, it is said that arguments A1 and A3 have the same form.

In order to evaluate the validity or invalidity of an argument, we consider that each of its propositions contains logical terms (words such as "and", "or", "not", "implies", "all", and so on) and terms that are specific to the domain about the subject matter of the argument. With this distinction, we define the *form* of an argument as being an argument where all the specific (that is non-logical) terms are replaced by a symbol that is associated with their grammatical category.[1] As a first example, the form of arguments A1 and A3 is the following:

F1: A is a B
No B has C
Therefore, A does not have C

[1] For example, proper names, nouns, adjectives, and so on.

where A is a proper name, and B and C are nouns. Argument A2 has the form:

F2: A is a C
Every B is a C
A is not a B
Therefore, not every C is a B

where A is a proper name, and B and C are nouns.

The form of an argument may be studied independently form the subject matter about which the propositions deal. In fact, an argument is valid or invalid because of its from and not because of its subject matter. Thus, Logic studies arguments in respect to their from (this is the reason why it is called a *formal* system). A good discussion related with the form of an argument can be found in [Haack 78, pp. 22-27].

In Mathematics there is a set of symbols that is used to represent the entities that mathematics deals with; in logic we use symbols to represent the entities that are common in arguments, the logical terms. We usually represent "and" by "$\wedge$", "or" by "$\vee$", "not" by "$\neg$", "implies" by "$\rightarrow$", "all" by "$\forall$", and so on. Using this notation, an argument with form F1, for example, will be written as:

F1$'$: $B(A)$
$\forall x[B(x) \rightarrow \neg C(x)]$
$\neg C(A)$

In this chapter we present the methodology to evaluate arguments in respect to their form and to decide whether they are valid or invalid.

2.1 Components of a logic

Before we develop a logic we have to define the language that will be used in the description of our arguments. This language will define what things we can talk about, and with what level of detail. This language is defined through a set of formation rules that specify which are the legal sentences in the logic, the so called *well formed formulas* (*wffs*, for short). If we denote by $\mathcal{L}$ the language that corresponds to the well formed formulas, an argument corresponds to the pair (Δ, α), where $\Delta \subset \mathcal{L}$ and $\alpha \in \mathcal{L}$.

The sentences of this language will denote propositions, over which we will perform two types of operations: on one hand, we will carry out proofs, sequences of sentences, starting with the premises, and obtained by symbol-manipulation operations, that attempt at reaching certain conclusion; on the other hand, we will assign meaning (and, consequently, a truth value) to the propositions that made up the arguments, with the goal of accessing the validity or invalidity of the arguments. These two types of operations correspond to the two components of a Logic, the deductive system and the semantics.

The *deductive system* contains a set of rewrite rules, the rules of inference, that specify how to generate new formulas from formulas that were already derived. The rules of inference define how to produce new formulas from existing formulas. Rules of inference *do not* belong to the language $\mathcal{L}$ but "talk" about the entities of $\mathcal{L}$, that is, they belong to the *meta-language* of $\mathcal{L}$. They can be looked as rewrite rules that state that from a certain number of wffs with a given form, we are allowed to write a new wff with a certain form. The deductive system may also contain a certain number of wffs that are accepted without proof, the *axioms*.

Given the argument (Δ, α), if there is a sequence of inference rules that applied to the wffs in Δ (and to the wffs that are generated from Δ) produces α, we say that α is *derivable* from Δ, and we write

$$\Delta \vdash \alpha.$$

If $\Delta \vdash \alpha$, we say that the argument (Δ, α) is *provable*. The sequence of wffs generated from Δ in order to obtain α (including Δ and α) are called a proof of α from Δ. A *proof* is a finite sequence of wffs (usually written as a column) such that each wff is either an axiom, a theorem (a wff that is derivable from an empty set of premises), or corresponds to the application of a rule of inference to wffs above it in the proof.

Given a set of wffs $\Delta \subset \mathcal{L}$, the set of all wffs derivable from Δ is called the set of *theorems of* Δ, or the *theory generated from* Δ, and is written $Th(\Delta)$. Formally,

$$Th(\Delta) = \{\alpha : \Delta \vdash \alpha\}.$$

We should keep in mind that the symbol "$\vdash$" is dependent on the deductive system (since it depends on the rules of inference that are defined), and thus, to be precise, we should index it with an identification of the corresponding deductive system. However, to make the notation easier, when there is no danger of confusion, we will use just "$\vdash$", meaning derivability in the deductive system under consideration.

The *semantics* specifies the conditions under which the propositions (wffs) are true or false. The semantics is based on the notion of interpretation, a mapping between the entities in the language $\mathcal{L}$ and the entities in the conceptualization (the entities in the world that we are interested in – see Section 2.4). An *interpretation* is an association of the objects of the conceptualization with the objects in our language.

Semantics, specified trough interpretations of the language, enables to evaluate the truth value of sentences, under interpretations. Given the argument (Δ, α), if there is no interpretation that makes all propositions in Δ true and α false, then we say that Δ *logically implies* α, or that α *is a logical consequence of* Δ, and we write

$$\Delta \models \alpha.$$

If $\Delta \models \alpha$, then the argument (Δ, α) is valid. Note that the sentence "if there is no interpretation that makes all propositions in Δ true and α false" is quite

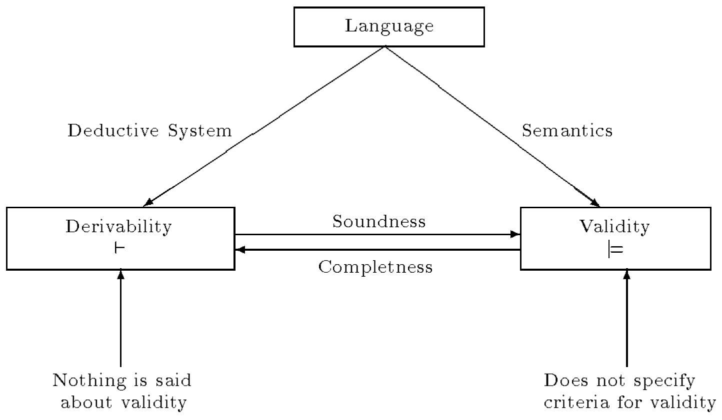

Figure 2.1: Relationship between the deductive system and the semantics.

strongly stated; it does not mean that *we are unable* to come up with an interpretation that makes all propositions in Δ true and α false, but it means that it is *impossible* to do so.

Notice that, in the deductive system nothing is said about validity (it only involves syntactic manipulation of wffs); on the other hand, in the semantics, nothing is stated about the criteria used to determine validity (in the semantics nothing is said about derivability). Although the deductive system and the semantics are distinct aspects of a logic, they should be compatible in order for the logic to make sense. A logic is said to be *sound* if any argument that is provable with its deductive system is valid according to its semantics; a logics is said to be *complete* if any argument valid according to its semantics is provable in its deductive system (Figure 2.1). Informally, a logic being sound means that we make no mistakes in the proofs or that only valid arguments are deducible; and a logic being complete means that we can prove all propositions that are true or that all valid arguments are deducible.

The concepts of soundness and completeness are not properties of the deductive systems and the semantics alone, but rather a relationship between the two systems. In a system that is solid and complete the notions of provability and validity are extensionally equivalent in the sense that they apply exactly to the same arguments. Unfortunately, the notion of completeness only applies to simple logics, and most of the logics that are interesting from the AI point of view are not complete.

2.2 The language of logic

An important aspect in the definition of a Logic is the choice of the language to the used in the representation of propositions. In the definition of this language a basic alphabet is established (that defines the terminal symbols of the language), and a set of rules is stated that determine how the basic components are combined in order to produce sentences or *well formed formulas* (*wffs*, for short).

The definition of the language determines the things that can be expressed in the language and the degree of detail with which these things can be expressed. In Classical Logic, there are two main alternatives for the definition of the language, corresponding to propositional logic and to first-order logic. In *propositional logic*, propositions have no "internal structure"; *first-order logic* is richer in the representation point of view in the sense that is allows to refer to the "parts" of a proposition.

2.2.1 The language of propositional logic

In this section, we present the definition of a language for propositional logic, denoted by $\mathcal{L}_0$.

1. *Terminal symbols*:
   (a) *Punctuation symbols*: ( )
   (b) *Logical symbols*:[2] $\neg \wedge \vee \rightarrow$
   The symbol "$\neg$" is read "not" and corresponds to the negation operator; the symbol "$\wedge$" is read "and" and corresponds to the conjunction operator; the symbol "$\vee$" is read "or" and corresponds to the disjunction operator; the symbol "$\rightarrow$" is read "implies" and corresponds to the implication operator.
   (c) *Predicate letters*, for $i > 0$: P_i
   A predicate letter corresponds to a symbol that can be assigned a truth value. Since the definition of the language is not concerned with meaning, we call these symbols predicate letters, rather than just predicates. The reason for parameterizing predicate letters, instead of using individual letters to denote them, is that resorting to a parameter gives us an unlimited capability of defining predicate letters. When there is no danger of confusion, we will use upper-case roman letters, P, Q, R, etc., in place of predicate letters.
2. *Well formed formulas* (wffs):
   (a) Predicate letters are well formed formulas;
   (b) If α is a wff, then $(\neg\alpha)$ is a wff;
   (c) If α and β are wffs, then $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, and $(\alpha \rightarrow \beta)$ are wffs;

[2] These are also called *logical connectives*.

(d) Nothing more is a wff.

According to these definitions, if P, Q, and R are predicate letters, then the following expressions are wffs:

$$((P \wedge Q) \rightarrow R)$$

$$(\neg P)$$

$$((P \vee ((P \wedge Q) \rightarrow R))) \wedge R)$$

Whenever possible, redundant parenthesis are omitted, and the wffs above are written as:

$$(P \wedge Q) \rightarrow R$$

$$\neg P$$

$$(P \vee ((P \wedge Q) \rightarrow R)) \wedge R$$

We will use lower-case greek letters (α, β, γ, etc.) as meta-variables whose domain is the set of wffs. For example, the wff $(P \wedge Q) \rightarrow R$, may be represented by $\alpha \rightarrow \beta$, where α is a meta-variable whose value is the wff $P \wedge Q$ and β is a meta-variable whose value is the wff R; but it can also be represented as $(\alpha \wedge \beta) \rightarrow \gamma$, where α is a meta-variable whose value is the wff P, β is a meta-variable whose value is the wff Q, and γ is a meta-variable whose value is the wff R.

2.2.2 The language of first-order logic

Propositional logic does not allow the use of variables. When we want to use variables as part of wffs, we obtain a logic called *first-order logic*. In first-order logics wffs have an "internal structure". First-order logic requires a new definition of rules for wffs, that introduce new logical symbols and new formation rules. We can define the language of first-order logic, denoted by $\mathcal{L}_1$ as follows:

1. *Terminal symbols*

   (a) *Punctuation symbols:* , ( ) [ ]

   (b) *Logical symbols:* $\neg \ \wedge \ \vee \ \rightarrow \ \forall \ \exists$
   Besides the symbols already used in propositional logic, the symbol "$\forall$" is read "for all" and corresponds to the universal quantification operator; the symbol "$\exists$" is read "exists" and corresponds to the existential quantification operator.

   (c) *Function letters* with n arguments, for $n \geq 0$ and $i \geq 1$: f_i^n.
   Again, the reason for parameterizing the function letters, instead of using individual letters to denote them, stems from the fact that resorting to a parameter gives a an unlimited capability of defining function names. A function letter denotes a function over elements of the language. Functions with zero arguments (corresponding to

f_i^0) correspond to constants.
When there is no danger of confusion, function letters will be denoted by lower-case roman identifiers,[3] f, g, h, $above$, $father$, etc.

(d) *Predicate letters* with n arguments, for $n \geq 0$ and $i \geq 1$: P_i^n.
A predicate letter represents an operator over the elements of the language, that has a truth value. When there is no danger of confusion, we will use identifiers beginning with an upper-case roman letter, P, Q, R, $ClearTop$, $Human$, etc., to denote predicate letters.

(e) *Individual variables*, for $i \geq 1 : x_i$
Individual variables, have as domain the objects of our conceptualization. When there is no danger of confusion, we will use lower-case roman letters, x, y, z, etc., to denote individual variables.

2. *Terms.* These are recursively defined, as follows:

   (a) Each function letter with zero arguments is a term;

   (b) Each variable is a term;

   (c) If t_1, t_2, ..., t_n are terms. then $f_i^n(t_1, t_2, \ldots, t_n)$ is a term;

   (d) Nothing more is a term.

   Terms represent entities about which we want to talk. They correspond to noun phrases in natural language.

3. *Well formed formulas* (wffs):

   (a) If t_1, t_2, ... t_n are terms, then $P_i^n(t_1, t_2, \ldots, t_n)$ is a wff (this is called an *atomic wff*);

   (b) If α is a wff, then $(\neg\alpha)$ is a wff;

   (c) If α and β are wffs, then $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, and $(\alpha \rightarrow \beta)$ are wffs;

   (d) If α is a wff, containing zero or more occurrences of the individual variable x, then $\forall x[\alpha]$ and $\exists x[\alpha]$ are wffs;

   (e) Nothing more is a wff.

As an example, let us suppose that P and $OnTopOf$ are predicate letters with two arguments, Q, $Block$, and $Clear$ are predicate letters of one argument, f is a function letter with one argument, and g is a function letter with three arguments, then

$$(\neg P(a, g(a, b, c)))$$

$$(P(a, b) \rightarrow (\neg Q(f(d))))$$

$$\forall x[(Block(x) \rightarrow \exists y[((Block(y) \wedge OnTopOf(y, x)) \rightarrow (\neg Clear(x)))])]$$

are wffs. Whenever possible, redundant parenthesis will be omitted, thus these wffs will be written as

$$\neg P(a, g(a, b, c))$$

[3] Sequences of one or more letters.

$$P(a,b) \rightarrow \neg Q(f(d))$$

$$\forall x[Block(x) \rightarrow \exists y[(Block(y) \wedge OnTopOf(y,x)) \rightarrow \neg Clear(x)]].$$

In the wffs $\forall x[\alpha]$ and $\exists x[\alpha]$ the wff α is called the *domain* of the quantifier ($\forall$ or $\exists$), and the quantifier is said to *introduce* the variable x. Note that α does not necessarily have to contain the variable x (in this case, $\forall x[\alpha]$ and $\exists x[\alpha]$ means the same as just α).

An occurrence of the variable x is said to be *bound* in the wff α if it belongs to the domain of the quantifier that introduces it; otherwise it is said to be *free*. In other words, an occurrence of the variable x in the wff α is bound in the wffs $\forall x[\alpha]$ and $\exists x[\alpha]$. For example, $A(x)$ contains a free occurrence of the variable x (also said that $A(x)$ contains the free variable x); $\forall x[A(x)]$ contains the bound variable x; $A(x) \rightarrow \exists x[B(x)]$ contains a free occurrence of x (in $A(x)$) and a bound occurrence of x (in $B(x)$). A wff with no free variables is said to be *closed*.

When dealing with wffs with variables, we often need to replace instances of variables by terms. We define a *substitution* as a finite set of ordered pairs $\{t_1/x_1, \ldots, t_n/x_n\}$, where each x_i $(1 \leq i \leq n)$ is an individual variable, and each t_i $(1 \leq i \leq n)$ is a term. The application of the substitution $s = \{t_1/x_1, \ldots, t_n/x_n\}$ to the wff α (represented by $\alpha \circ s$) is the wff obtained from α by replacing all occurrences of the free variable x_i by t_i $(1 \leq i \leq n)$. For example, given the wff $P(x, f(a,y))$, which contains the free variables x and y the substitution $s = \{a/x, f(a,b)/y\}$, the substitution application s to $P(x, f(a,y))$ originates

$$P(x, f(a,y)) \circ \{a/x, f(a,b)/y\} = P(a, f(a, f(a,b))).$$

The same result would have been obtained by the application of the substitution $s' = \{a/x, f(a,b)/y, c/z\}$ to $P(x, f(a,y))$. The application of the same substitution to $A(x) \rightarrow \exists x[B(x)]$ originates

$$A(x) \rightarrow \exists x[B(x)] \circ \{a/x, f(a,b)/y\} = A(a) \rightarrow \exists x[B(x)].$$

We write $\alpha(x_1, \ldots, x_n)$ to explicitly state that the wff α has x_1, ..., x_n as free variables. Note that this notation does not prevent α to have other free variables. Using the same notation, we write,

$$\alpha(x_1, \ldots, x_n) \circ \{t_1/x_1, \ldots, t_n/x_n\} = \alpha(t_1, ..., t_n).$$

If α is a wff and t is a term, we say that t is *free for x in α* if no free occurrence of x in α occurs inside the scope of a quantifier ($\forall x_j$ or $\exists x_j$), where x_j is a variable in t. Informally, t being free for x_i in α means that if all occurrences of x are replaced by t no occurrence of a variable in t loses its free status in $\alpha(t)$. For example, the term $g(y, f(b))$ is free for x in the wff $P(x,y)$, but it is not free for x in the wff $\forall y[P(x,y)]$. Note that, by definition, a term with no variables is always free for any variable.

2.3 Deductive system

In the deductive system, we specify rules of inference, that is, rules that enable the introduction of new wffs from existing wffs. There are two alternatives for the definition of the deductive system of a logic, corresponding to natural deduction systems and axiomatic systems. A *natural deduction system* only contains rules of inference, and no axioms; an *axiomatic system*, besides a set of rules of inference, contains a distinguished set of wffs, the *axioms*, that are accepted without proof.

2.3.1 Natural deduction systems

Natural deduction systems have their origins in the work of Gentzen in 1934 (reprinted in [Gentzen 69]) and [Jaskowski 34]. The formalism used in this chapter was introduced in [Fitch 52].

Natural deduction systems have no axioms, only rules of inference. Typically, there are two rules of inference for each logical symbol, one that states how to introduce a wff that uses the logical symbol, the *introduction rule*, and the other states how to use a wff that contains the lofical symbol, the *elimination* rule.

Natural deduction systems introduce the notion of *hypothetical proof*, a proof initiated with the introduction of a hypothesis; the wffs in a hypothetical proof are either theorems, the hypothesis that originated the hypothetical proof, or the result of the application of a rule of inference to the wffs in the proof. Wffs in a hypothetical proof are called *contingent*, since they depend on a hypothesis; wffs in a non-hypothetical proof are called *categorical* since they do not depend on hypothesis.

As an example of a natural deduction system, we will present system FH from [Fitch 52]. This system has 15 rules of inference (FH1 to FH15). There are slight differences between the rules of inference that we present and the rules of inference in [Fitch 52], namely, in what concerns negation and to implication (which is used by Fitch and not used by us). However FH system and the system that we discuss are equivalent, in the sense that they have exactly the same theorems. By definition, a *theorem in the system* is a wff that is obtained in a categorical proof (a proof that does not depend on hypothesis).

The lines in our proofs are numbered and, at the end of each line, we write the rule of inference used to derive the wff in that line and the lines that were considered by the rule of inference.

The following the rules of inference in the FH system. Notice that, since rules of inference belong to the meta-language, they talk about wffs. For this reason, rules of inference use variables that range over wffs.

FH1 *Hypothesis* (Hyp): At any point, we may introduce any wff as a hypothesis. This hypothesis initiates a new hypothetical proof, which is represented as follows:

$$\begin{array}{r|ll} n & \alpha & Hyp(n) \\ n+1 & & \end{array}$$

FH2 *Repetition* (*Rep*): Any wff may be repeated inside a proof. This is represented in the following way:

$$\begin{array}{lll} n & \alpha & \\ & \ldots & \\ m & \alpha & Rep(n) \end{array}$$

FH3 *Re-iteration* (*Reit*): Any wff in a proof may be re-iterated (repeated) in any hypothetical proof inside the proof. This is represented in the following way:

$$\begin{array}{llll} n & \alpha & & \\ & \ldots & & \\ m & \mid & \alpha & Reit(n) \end{array}$$

FH4 *Implication introduction* ($\rightarrow I$): If, in a proof initiated by the hypothesis α, we are able do derive β, then we may derive $\alpha \rightarrow \beta$ in the proof that contains the hypothetical proof initiated by the hypothesis α:

$$\begin{array}{llll} n & \mid & \alpha & Hip \\ & \mid & \ldots & \\ m & \mid & \beta & \\ k & & \alpha \rightarrow \beta & \rightarrow I(n,m) \end{array}$$

FH5 *Implication elimination*, also known as *modus ponens* ($\rightarrow E$): In a proof that contains both α and $\alpha \rightarrow \beta$ we may derive β:

$$\begin{array}{lll} n & \alpha & \\ & \ldots & \\ m & \alpha \rightarrow \beta & \\ k & \beta & \rightarrow E(n,m) \end{array}$$

FH6 *Negation introduction* ($\neg I$): If, in a proof initiated by the hypothesis α we derive a contradiction, then we can conclude $\neg\alpha$ in the proof containing the hypothetical proof initiated by the hypothesis α. This corresponds to a proof by way of contradiction.

$$\begin{array}{llll} n & \mid & \alpha & Hip \\ & \mid & \ldots & \\ m & \mid & \beta & \\ k & \mid & \neg\beta & \\ l & & \neg\alpha & \neg I(n,m,k) \end{array}$$

FH7 *Negation elimination* ($\neg E$): If, in a proof, a wff is negated twice, we can conclude that wff:

$$\begin{array}{lll} n & \neg\neg\alpha & \\ m & \alpha & \neg E(n) \end{array}$$

FH8 *Disjunction Introduction* ($\vee I$): From the wff α, we can either derive $\alpha \vee \beta$, $\beta \vee \alpha$, or both, where β is any wff:

$$\begin{array}{lll} n & \alpha & \\ m & \alpha \vee \beta & \vee I(n) \end{array}$$

$$\begin{array}{lll} n & \alpha & \\ m & \beta \vee \alpha & \vee I(n) \end{array}$$

FH9 *Disjunction elimination* ($\vee E$): In, in a proof we have (1) the wff $\alpha \vee \beta$; (2) an hypothetical proof from the hypothesis α containing the wff γ; and (3) an hypothetical proof from the hypothesis β containing the wff γ, then we can derive the wff γ in that proof:

$$\begin{array}{rcll} n & & \alpha \vee \beta & \\ o & | & \alpha & Hip \\ & | & \ldots & \\ p & | & \gamma & \\ & \ldots & & \\ r & | & \beta & Hip \\ & | & \ldots & \\ s & | & \gamma & \\ m & & \gamma & \vee E(n, (o, p), (r, s)) \end{array}$$

FH10 *Conjunction introduction* ($\wedge I$): If, in a proof, we have the wff α and the wff β, then, we can derive $\alpha \wedge \beta$:

$$\begin{array}{lll} n & \alpha & \\ & \ldots & \\ m & \beta & \\ k & \alpha \wedge \beta & \wedge I(n, m) \end{array}$$

FH11 *Conjunction elimination* ($\wedge E$): From the wff $\alpha \wedge \beta$, we can derive α, β, or both:

$$\begin{array}{lll} n & \alpha \wedge \beta & \\ m & \alpha & \wedge E(n) \end{array}$$

$$\begin{array}{lll} n & \alpha \wedge \beta & \\ m & \beta & \wedge E(n) \end{array}$$

These rules of inference can either be used in propositional logic or in first-order logic. Before we present rules of inference that are only used in first-order logic, we show some proofs using the FH system. As a first example, we show in Figure 2.2 the proof of the theorem

$$(P \rightarrow (Q \rightarrow R)) \rightarrow ((P \wedge Q) \rightarrow R).$$

$$
\begin{array}{rlll}
1 & \mid & P \rightarrow (Q \rightarrow R) & Hyp \\
2 & \mid & \mid\ P \wedge Q & Hyp \\
3 & \mid & \mid\ P & \wedge E(2) \\
4 & \mid & \mid\ P \rightarrow (Q \rightarrow R) & Reit\ (1) \\
5 & \mid & \mid\ Q \rightarrow R & \rightarrow E(3,4) \\
6 & \mid & \mid\ Q & \wedge E(2) \\
7 & \mid & \mid\ Q \rightarrow R & Rep(5) \\
8 & \mid & \mid\ R & \rightarrow E(6,7) \\
9 & \mid & (P \wedge Q) \rightarrow R & \rightarrow I(2,8) \\
10 & & ((P \rightarrow (Q \rightarrow R)) \rightarrow ((P \wedge Q) \rightarrow R)) & \rightarrow I(1,9)
\end{array}
$$

Figure 2.2: Proof of $(P \rightarrow (Q \rightarrow R)) \rightarrow ((P \wedge Q) \rightarrow R)$.

$$
\begin{array}{rlll}
1 & \mid & R & Hyp \\
2 & \mid & \mid\ Q & Hyp \\
3 & \mid & \mid\ R & Reit\ (1) \\
4 & \mid & \mid\ Q & Rep\ (2) \\
5 & \mid & \mid\ R \wedge Q & \wedge I(3,4) \\
6 & \mid & Q \rightarrow (R \wedge Q) & \rightarrow I(2,5) \\
7 & & R \rightarrow (Q \rightarrow (R \wedge Q)) & \rightarrow I(1,6)
\end{array}
$$

Figure 2.3: Proof of $R \rightarrow (Q \rightarrow (R \wedge Q))$.

You should note that this wff is obtained in a categorical proof, and thus does not depend on any hypothesis. The argument corresponding to this proof is

$$(\{\}, (P \rightarrow (Q \rightarrow R)) \rightarrow ((P \wedge Q) \rightarrow R)),$$

which means that

$$\{\} \vdash (P \rightarrow (Q \rightarrow R)) \rightarrow ((P \wedge Q) \rightarrow R).$$

When $\{\} \vdash \alpha$, that is when α is a theorem of the system, it is usual to write just $\vdash \alpha$.

Figure 2.3 shows the proof of the theorem $R \rightarrow (Q \rightarrow (R \wedge Q))$. Notice again that this proof is generated from an empty set of premises, thus $\{\} \vdash (R \rightarrow (Q \rightarrow (R \wedge Q)))$ and the reason of being a theorem. You should also note that line 6 in Figure 2.3 was derived under the hypothesis R, which enable us to write that $\{R\} \vdash (Q \rightarrow (R \wedge Q))$.

Figures 2.4 and 2.5 show, respectively, the proofs of $A \rightarrow (B \rightarrow A)$ (any proposition implies a true proposition), and $(A \wedge \neg A) \rightarrow B$ (a contradiction implies any proposition). These results as know as the *paradoxes of implication* and generated reactions against classical logic (see Chapter 3).

1	$\mid\quad A$	Hyp
2	$\mid\ \mid\ B$	Hyp
3	$\mid\ \mid\ A$	$Reit(1)$
4	$\mid\quad B \rightarrow A$	$\rightarrow I(2,3)$
5	$A \rightarrow (B \rightarrow A)$	$\rightarrow I(1,4)$

Figure 2.4: Proof of $A \rightarrow (B \rightarrow A)$.

1	$\mid\quad A \wedge \neg A$	Hyp
2	$\mid\ \mid\ \neg B$	Hyp
3	$\mid\ \mid\ A \wedge \neg A$	$Reit(1)$
4	$\mid\ \mid\ A$	$\wedge E(3)$
5	$\mid\ \mid\ \neg A$	$\wedge E(3)$
6	$\mid\quad \neg\neg B$	$\neg I(2,4,5)$
7	$\mid\quad B$	$\neg E(6)$
8	$(A \wedge \neg A) \rightarrow B$	$\rightarrow I(1,7)$

Figure 2.5: Proof of $(A \wedge \neg A) \rightarrow B$.

In first-order logic, rules of inference should handle the introduction and elimination of quantifiers. In FH, there are four rules of inference to handle quantifiers, FH12 to FH15:

FH12 *Universal quantifier introduction* ($\forall I$): If, in a proof initiated by the hypothesis $\alpha(x)$, where x is an arbitrary variable (a variable that was never used in the proof before) we have the wff $\beta(x)$, then, we can conclude that "for all x, $\alpha(x) \rightarrow \beta(x)$" in the proof that contains the hypothetical proof initiated by the hypothesis $\alpha(x)$. This type of quantification, where the universal quantifier is introduced associated with an implication, is called *relativized quantification*. The role of the antecedent of the implication $\alpha(x)$ is used to define the kind on entities being quantified. In the representation of this inference rule, x to the left of the hypothesis denotes that x is a new variable that had never been used before in the proof.

n	$x\mid\ \alpha(x)$	Hyp
	$\mid$	
	$\mid$	
m	$\mid\ \beta(x)$	
k	$\forall x[\alpha(x) \rightarrow \beta(x)]$	$\forall I(n,m)$

FH13 *Universal quantification elimination* ($\forall E$): from $\forall x[\alpha(x)]$ we may infer $\alpha(a)$, where "a" is any individual constant (a function letter with zero

arguments):

$$\begin{array}{lll} n & \forall x[\alpha(x)] & \\ & \ldots & \\ m & \alpha(a) & \forall E(n) \end{array}$$

FH14 *Existential quantification introduction* ($\exists I$): from $\alpha(a)$, we may infer $\exists x[\alpha(x)]$, where x is free in $\alpha(x)$:

$$\begin{array}{lll} n & \alpha(a) & \\ & \ldots & \\ m & \exists x[\alpha(x)] & \exists I(n) \end{array}$$

FH15 *Existential quantification elimination* ($\exists E$): From $\exists x[\alpha(x)]$, we may infer $\alpha(a)$, where "a" is a new individual constant that has never been used before in the proof:

$$\begin{array}{lll} n & \exists x[\alpha(x)] & \\ & \ldots & \\ m & \alpha(a) & \exists E(n) \end{array}$$

You should note that whereas inference rule FH13 allows that "a" be any constant (it relies on universal quantification), inference rule FH15 requires "a" to be a constant that had never been used in the proof before (we are not allowed to say anything about the constants that we already used in our proof).

2.3.2 Axiomatic systems

Axiomatic systems or *Hilbert-style systems* rely on the existence of a set of axioms (wffs that are accepted in the system without proof) and on a set (usually small) of rules of inference. As an example of an axiomatic system, we will use system L of Mendelson [Mendelson 87, p. 29], that contains, for propositional logic, three axioms (or more precisely, three axiom schemata, since they are defined in terms of meta-variables) and one rule of inference, modus ponens. The axioms of L are:

A1: $\alpha \rightarrow (\beta \rightarrow \alpha)$

A2: $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$

A3: $((\neg\beta) \rightarrow (\neg\alpha)) \rightarrow (((\neg\beta) \rightarrow \alpha) \rightarrow \beta)$

Axiom A1 states that a true proposition is implied by any proposition; axiom A2 corresponds to the distributive property of implication; axiom A3 corresponds to double negation, in fact, if $\neg\beta$ implies both α and $\neg\alpha$, then $\neg\beta$ must be false, therefore $\neg\neg\beta$, in other words, β.

The fact that axioms A1, A2, and A3 are written using meta-variables (α, β, and γ) means that each of them corresponds to an infinite collection of wffs. For example, the following wffs are instances of axiom A1: $A \rightarrow (B \rightarrow A)$,

1	$(A \to ((A \to A) \to A)) \to$	
	$((A \to (A \to A)) \to (A \to A))$	$A2$
2	$A \to ((A \to A) \to A)$	$A1$
3	$(A \to (A \to A)) \to (A \to A)$	$MP(1,2)$
4	$A \to (A \to A)$	$A1$
5	$A \to A$	$MP(3,4)$

Figure 2.6: Proof of $A \to A$.

1	$((A \to (B \to C)) \to ((A \to B) \to (A \to C))$	$A2$
2	$(A \to (B \to C))$	$in\ \Delta$
3	$(A \to B) \to (A \to C)$	$MP(1,2)$
4	$(((A \to B) \to (A \to C))$	
	$\to (B \to ((A \to B) \to (A \to C))))$	$A1$
5	$B \to ((A \to B)) \to ((A \to C))$	$MP(3,4)$
6	$((B \to ((A \to B) \to (A \to C)))$	
	$\to ((B \to (A \to B)) \to (B \to (A \to C))))$	$A2$
7	$(B \to (A \to B)) \to (B \to (A \to C))$	$MP(5,6)$
8	$B \to (A \to B)$	$A1$
9	$B \to (A \to C)$	$MP(7,8)$

Figure 2.7: Proof of $\{A \to (B \to C))\} \vdash (B \to (A \to C))$.

$((A \wedge B) \to C) \to ((D \vee E) \to ((A \wedge B) \to C))$ e $A \to (A \to A)$. For this reason, they are called *axiom schemata*.

Another aspect that should be noticed is that none of these axioms deals with disjunction or conjunction. These logical symbols are treated as syntactic abbreviations of other logical symbols, using the definitions:

$$\alpha \wedge \beta =_{Df} \neg(\alpha \to \neg\beta)$$

$$\alpha \vee \beta =_{Df} (\neg\alpha) \to \beta.$$

The rule of inference of L, modus ponens, written for short as MP, is stated as: "From α and $\alpha \to \beta$ we may infer β".

As examples of proofs in the system L we show in Figure 2.6 the proof of the theorem $A \to A$ (from [Mendelson 87, p. 30]), and in Figure 2.7 the proof of $\{(A \to (B \to C))\} \vdash (B \to (A \to C))$.

The following are some theorems from L (from [Mendelson 87, pp. 30-33]). Their proof is left as an exercise. You should note that these are in fact metatheorems, since they make use of meta-variables:

$$\begin{array}{ll}
TL1: & (\neg\alpha \rightarrow \alpha) \rightarrow \alpha \\
TL2: & (\neg\beta \rightarrow \neg\alpha) \rightarrow (\alpha \rightarrow \beta) \\
TL3: & \neg\neg\beta \rightarrow \beta \\
TL4: & \neg\alpha \rightarrow (\alpha \rightarrow \beta) \\
TL5: & \alpha \rightarrow (\alpha \vee \beta) \\
TL6: & (\beta \vee \alpha) \rightarrow (\alpha \vee \beta) \\
TL7: & (\alpha \wedge \beta) \rightarrow \alpha \\
TL8: & (\alpha \wedge \beta) \rightarrow \beta \\
TL9: & \alpha \rightarrow (\beta \rightarrow (\alpha \wedge \beta))
\end{array}$$

Similarly to what happened in natural deduction, the consideration of first-order logic requires the extension of axiomatic systems with new axioms and new rules of inference. In the system L, for example, we have in addition two axioms:

A4: $\forall x[\alpha(x)] \rightarrow \alpha(t)$
Where $\alpha(x) \in \mathcal{L}_1$ and t is a term free for x in $\alpha(x)$.

A5: $\forall x[\alpha \rightarrow \beta] \rightarrow (\alpha \rightarrow \forall x[\beta])$
Where $\alpha \in \mathcal{L}_1$ and has no free occurrences of x.

Besides modus ponens, system L for first-order logic, contains one additional rule of inference, called *generalization*: "from α we may infer $\forall x[\alpha]$".

2.3.3 Properties of the deductive system

The deductive system of the logics that we discussed in this chapter (propositional logic and first-order logic) present a certain number of properties that we state here without proof (these proofs can be found in many books about classical logic, for example, [Church 56] and [Mendelson 87]).

THEOREM (Deduction Theorem):
If Δ is a set of wffs, α and β are wffs, and if $(\Delta \cup \{\alpha\}) \vdash \beta$, then $\Delta \vdash (\alpha \rightarrow \beta)$.

THEOREM:
If Δ is a set of wffs, $\alpha_1, \ldots, \alpha_n$ are wffs, β is a wffs, if $\Delta \vdash \alpha_1, ..., \Delta \vdash \alpha_n$, and if $\{\alpha_1, \ldots, \alpha_n\} \vdash \beta$, then $\Delta \vdash \beta$.

THEOREM (monotonicity):
If Δ_1 and Δ_2 are sets of wffs, if α is a wff, and if $\Delta_1 \vdash \alpha$, then $(\Delta_1 \cup \Delta_2) \vdash \alpha$.

COROLARY:
If Δ_1 and Δ_2 are sets of wffs and $\Delta_1 \subset \Delta_2$, then $Th(\Delta_1) \subset Th(\Delta_2)$.

1	$P \rightarrow Q$	*in* Δ
2	$P \rightarrow R$	*in* Δ
3	P	*em* Δ
4	Q	$MP(1,3)$
5	$Q \rightarrow (R \rightarrow (Q \wedge R))$	$TL9$
6	$R \rightarrow (Q \wedge R)$	$MP(4,5)$
7	R	$MP(2,3)$
8	$Q \wedge R$	$MP(6,7)$

Figure 2.8: Proof of $\{P \rightarrow Q, P \rightarrow R, P\} \vdash Q \wedge R$.

$$
\begin{array}{ll}
\{P \rightarrow Q, P \rightarrow R, P\} \vdash (Q \wedge R) & \\
\{P \rightarrow Q, P \rightarrow R\} \vdash P \rightarrow (Q \wedge R) & DT \\
\{P \rightarrow Q\} \vdash (P \rightarrow R) \rightarrow (P \rightarrow (Q \wedge R)) & DT \\
\{\} \vdash (P \rightarrow Q) \rightarrow ((P \rightarrow R) \rightarrow (P \rightarrow (Q \wedge R))) & DT
\end{array}
$$

Figure 2.9: Use of the deduction theorem.

THEOREM (fixed point):
If Δ is a set of wffs, then $Th(\Delta) = Th(Th(\Delta))$.

These theorems extends our capabilities of carrying out proofs. For example, let us suppose that we want to prove, in the system L, the theorem $(P \rightarrow Q) \rightarrow ((P \rightarrow R) \rightarrow (P \rightarrow (Q \wedge R)))$ (designated by TL10). We could start to prove that $\{P \rightarrow Q, P \rightarrow R, P\} \vdash Q \wedge R$ (Figure 2.8). Using the deduction theorem, we can conclude that $\{\} \vdash (P \rightarrow Q) \rightarrow ((P \rightarrow R) \rightarrow (P \rightarrow (Q \wedge R)))$ (Figure 2.9)[4], in other words, $(P \rightarrow Q) \rightarrow ((P \rightarrow R) \rightarrow (P \rightarrow (Q \wedge R)))$ is a theorem of L.

2.4 Semantics

Semantics specify in what conditions the wffs of our language are true or false. Recall that an wff is a sequence of symbols combined using specified syntactic rules. In order to evaluate the truth or falsity of a wff it is first necessary to "interpret" each of the symbols in the wff, by assigning a meaning to them. Afterwards, we have to access whether the meaning assigned to the wff agrees with the world that we are considering. If it agrees, we say that the wff is true; otherwise, we say that it is false. For example, suppose that we are describing, through a set of wffs, the situation in the blocks world that is shown in Figure 2.10. Given the wff $P(a, b)$, it will have no meaning before we "interpret"

[4] DT means "Deduction Theorem".

Figure 2.10: Situation in the blocks world.

the symbols P, a, and b. Suppose that after our interpretation of these symbols, the resulting meaning for this wff is that "block A is on top of block B". Under this interpretation, wff $P(a, b)$ is true.

In this section we describe how the process of evaluating the truth value of wffs under given interpretation is done. Notice that what we have done so far is quite imprecise, first of all the meaning of a wff as expressed through a sentence in natural language, with all the ambiguities that are inherent to it (for example, "on top of" is ambiguous; would block A will also be on top of block B if there were other blocks in between?). Second, the description of the world was done by resorting to a figure, which cannot be considered as a formal description of the world.

In order to describe how to formalize a world or a situation, we define the concept of a conceptualization. A *conceptualization* is a triple (D, F, R), where:

1. D, the *universe of discourse*, is the set of entities that correspond to the world or situation about which we will be talking about. The universe of discourse defines all entities that we may consider. Note that these are a subset of all entities in the world, containing only those that we are interested in.

   These entities may either be real objects (such as the planet Venus, the city of Lisbon, George Washington, or a set of blocks), abstract concepts (for example, loyalty, beauty, or the set of natural numbers), or fictional concepts (for example, Sherlock Holmes, Cinderela).

2. F is the set of functions that can be applied to the entities of the universe of discourse. Likewise the specification of the universe of discourse, where we only considered a subset of the entities existing in the real world, here, we only consider a subset of all function that can be applied to the entities of the universe of discourse, the functions that we are interested in. For example, if our universe of discourse includes natural numbers, one of the possible functions in D would be the function "*successor*"; if our universe of discourse includes the blocks world, one of the possible functions in D will be "*above*", that given a block returns the block that is immediately above it.

A function of n arguments is represented by a set of $(n+1)$-tuples of entities from the universe of discourse. For example, consider a situation in the blocks world, where there are only three blocks, A, B, and C, and in which A is above B, and B is above C, the function "*above*" would be defined as $\{(A,\ B), (B,\ C)\}$, which corresponds to the definition: $above(A) = B$ and $above(B) = C$. You should note that a function may be a partial function, that is, a function that is not defined for every element of the universe of discourse, as is the case with the function "*above*".

3. R is the set of all relations or predicates that can be applied to the entities of the universe of discourse. Again, among all possible relations over the entities of the universe of discourse, we are only interested in a subset, the relations that are relevant for our intended application. For example, if the universe of discourse includes natural numbers, a possible relation would be "$Smaller$"; if our universe of discourse includes the blocks world, a possible relation would be "$ClearTop$".

   A predicate with n arguments is defined by a set of n-tuples of entities of the universe of discourse. For example, if the universe of discourse contains the natural numbers less than four, the relation "$Smaller$" would be defined by the set $\{(1,\ 2), (1,\ 3), (2,\ 3)\}$.

A conceptualization formally describes a world or a situation. It is important to stress that, given a world or a situation, there is not just one conceptualization that describes it. The choice of the conceptualization depends on the aspects that are relevant for the intended application.

Another aspect related to the choice of conceptualizations relates with the fact that we often need to talk about properties. The following example, from [Genesereth and Nilsson 87, p. 13] illustrates this aspect. Considering the blocks world, suppose that we have the following conceptualization: $(\{A,\ B,\ C\}, \{\}, \{blue,\ green,\ yellow\})$.[5] This conceptualization would be adequate in an application where we would want to refer to the colors of objects, but it wouldn't be adequate in a situation where we would like to refer that the color of certain block is nice, since we cannot talk about properties of properties. The solution in this situation is to *reify* the property whose properties we want to talk about. This corresponds to the creation of entities of the universe of discourse that correspond to the property that we want to talk about and to create functions that return the property of that property. In our example, we could have $(\{A,\ B,\ C,\ blue,\ green,\ yellow\}, \{color\}, \{nice\})$, where *color* is a function that takes an argument a block and returns its color, and *nice* is a predicate that tells whether a color is nice.

From the concept of conceptualization, we can define the concept of interpretation. An *interpretation* is a total function (I) whose domain are the entities

[5] To make the notation easier to read, we are describing relations by their names rather than by the entities of the universe of discourse that satisfy them. To be precise, we should list the objects that have a certain color. For example, if blocks A, B, and C were blue, yellow, and green, respectively, we would have $R = \{\{A\}, \{C\}, \{B\}\})$.

in our language and whose range are the entities in our conceptualization. The fact that an interpretation is a function whose domain are the entities in our language means, among other things, that each constant in our language is associated with only one entity in our conceptualization, but different constants in our language my be associated with the same entity in our conceptualization. If α is an entity in our language, we write $I(\alpha)$ to denote the result of the application of the interpretation function I to the entity α.

In this chapter we are only concerned with the interpretation of closed wffs. A presentation of the interpretation of wffs with free variables can be found in [Mendelson 87, pp. 46-51].

Given the conceptualization (D, F, R), an interpretation, I, should satisfy the following conditions:

1. Each individual constant f_i^0 is associated with an entity of the universe of discourse D;

2. Each function letter f_i^n $(n > 0)$ is associated with a function of F. If f_i^n is a function letter with n arguments, corresponding to the function $I(f_i^n)$ of our conceptualization, and if t_1, ..., t_n are terms, then $f_i^n(t_1, \ldots, t_n)$ corresponds to the entity $I(f_i^n)(I(t_1), \ldots, I(t_n))$ of our conceptualization;

3. Each predicate letter P_i^n is associated with a relation of R.

Given the wff α, and an interpretation I, we say that I *satisfies* α, under the following conditions:

1. If α is an atomic wff, a wff of the form $P_i^n(t_1, \ldots, t_n)$, the interpretation I satisfies α if and only if the n-tuple $(I(t_1), \ldots, I(t_n))$ is an element of the relation $I(P_i^n)$.

2. The interpretation I satisfies the wff $\neg\alpha$ if and only if I does not satisfy α.

3. The interpretation I satisfies the wff $\alpha \wedge \beta$ if and only if I satisfies both wffs α and β.

4. The interpretation I satisfies the wff $\alpha \vee \beta$ if and only if I satisfies at least one of the wffs α or β.

5. The interpretation I satisfies the wff $\alpha \rightarrow \beta$ if and only if I does not satisfy α or I satisfies β.

6. The interpretation I satisfies the wff $\forall x[\alpha]$ if and only if, for all substitution $\{a/x\}$, where "a" is any individual constant, I satisfies $\alpha \circ \{a/x\}$.

7. The interpretation I satisfies the wff $\exists x[\alpha]$ if and only if there is a substitution $\{a/x\}$, where "a" is an individual constant, such that I satisfies $\alpha \circ \{a/x\}$.

Figure 2.11: Situation in the blocks world.

We say that the wff α is *true under the interpretation* I, if and only if I satisfies α; otherwise, we say that α is *false under the interpretation* I. If α is true under the interpretation I, we write $\models_I \alpha$.

An interpretation that satisfies all the wffs in a set of wffs is said to be a *model* of that set if wffs. If there is a models for a set of wffs these are said to be *satisfiable*. Under this perspective, we say that the argument (Δ, α) is valid if and only if all the models of the premises are also models of the conclusion.

For example, let us suppose that our language has four individual constants, (a, b, c, t), a function of one argument ($under$), and two predicated ($ClearTop$, $OnTopOf$), with one and two arguments, respectively. With these elements, we can write, among others, the following wffs

$$ClearTop(a)$$

$$ClearTop(c)$$

$$\neg ClearTop(under(a))$$

$$OnTopOf(a, b)$$

Let us consider the situation shown in Figure 2.11. Suppose that we define the interpretation:

$I(a) = A$
$I(b) = B$
$I(c) = C$
$I(t) = T$
$I(under) = \{(A, B), (B, T), (C, T)\}$
$I(ClearTop) = \{(A), (C)\}$
$I(OnTopOf) = \{(A, B)\}$

In this interpretation, the wffs above are true, thus the interpretation is a model of these wffs.

If we consider the interpretation J, defined in the following way:

$J(a) = A$
$J(b) = B$
$J(c) = C$
$J(t) = T$

$$J(under) = \{(B, A)\}$$
$$J(ClearTop) = \{(B)\}$$
$$J(OnTopOf) = \{(A, B)\}$$

wffs $ClearTop(a)$, and $ClearTop(c)$, for example, have the value false, thus J is not a model of the wffs above.

The methodology that we presented to compute the truth value of wffs was introduced by Alfred Tarski in 1936. The original paper can be found in [Corcoran 83].

2.5 Soundness and completness

The logics that we described in this chapter satisfy the following properties (their proof can be found in [Mendelson 87]):

THEOREM (soundness): First-order logic is sound.

THEOREM (completness): First order logic is complete.

2.6 Knowledge representation using logic

Logic has been proposed as a knowledge representation of AI since the very beginnings. In fact, the first proposal to use logic as a knowledge representation formalism precedes the birth of AI, and was done in the 17-th Century by Leibniz (see [Brachman and Levesque 85, p. xiv]).

The first paper on knowledge representation in AI, written by John McCarthy in 1958 (reprinted then years later in [McCarthy 68]), proposed that the knowledge necessary to an intelligent program should be expressed by declarative sentences, in a way that was independent from the way knowledge was going to be used; it also proposed that the reasoning performed by such a program should use logical operations over those declarative sentences. McCarthy's approach originated what is know today as the *logicist approach to AI*, that has a large number of followers (see [Hayes 77], [Moore 82, 84], [Israel 87], [McCarthy 88], and [Nilsson 91]). According to [Nilsson 91] there are three basic premises underlying knowledge representation based on logic:

1. Intelligent programs have knowledge about the environment they deal with;
2. Intelligent programs represent most of their knowledge using a declarative representation;
3. The declarative language used to represent knowledge should have, at least, the expressive power of first-order logic or one of its extensions.

Besides these premises, the approach that uses logic as a knowledge representation formalism, makes particular commitments concerning what knowledge is,

Figure 2.12: Relationship between knowledge in a program and the outside world.

what is the outside world, how the program interacts with the outside world, and about the role of the procedures used by the program. This description is based on [Gensereth and Nilsson 87, pp. 307-328], [Nilsson 91], and [Pinto-Ferreira 91].

An important aspect to keep in mind is that the program will act upon an abstraction (or a model) of the real world, this corresponds, in logic, to a conceptualization. The conceptualization corresponds to an abstraction of the real world, where only relevant entities and properties are considered. This conceptualization corresponds to an "invention", from the people that designs the representation, of the existing objects, their functions, and their relations.This abstraction defines a finite number of possible states of the real world. The set of all states of the real world is denoted by M; at any given instant the world is in a state $m \in M$.

The program represents its knowledge of the world through a collection of declarative sentences, using a language based on logic (eventually extended with modal operators – see Chapter 3), denoted by $\mathcal{L}$. At any given moment, the (partial) representation of the world is expressed through a set of sentences $\Delta \in 2^{\mathcal{L}}$ (Figure 2.12).

The knowledge represented in the program evolves trough three classes of operations: (1) operations that rely on a set of sensors that inspect the outside world (note that these sensor only can identify the characteristics of the real world that exist in the abstraction) and modify the knowledge in the system; (2) operations that correspond to reasoning procedures, through which the program reflects upon its knowledge and either modifies its knowledge or decides to act upon the world; (3) operations that rely on a set of actuators that based on knowledge in the knowledge base act upon the outside world (again, these actuators only can act upon the characteristics of the real world that exist in the abstraction).

Let us now consider these operation in detail:

1. *Sensors.* Sensors correspond to operations that send information from the outside world into the program. The result of an operation corresponding to a sensor is an input command s that belongs to a set of possible commands S. The result of a sensor is translated by a function *see* that

transforms M in S. The possible results of sensors are called *epistemic inputs* [Gärdenfors 88].

2. *Reasoning.* Reasoning is originated from an epistemic input and has, as a result, either the change of the information in the knowledge base or the generation of a command for an actuator. The first aspect, corresponds to a function called *mem* that transforms $s \times 2^{\mathcal{L}}$ in $2^{\mathcal{L}}$; the second aspect, corresponds to a function called *act* that transforms $s \times 2^{\mathcal{L}}$ in A, where A corresponds to a command for an actuator.

3. *Actuators.* Actuators correspond to the operations that send information from to program to the outside world. The result of an operation from an actuator can be expressed by a command in the set of the possible commands A.

Typically, an AI program receives the information corresponding to sensors through the keyboard and the actuators correspond to commands printed on the screen.[6] Taking into account the way the interaction between the program and the outside world is done, let us consider each one of these operations in greater detail:

1. *Sensors.* These commands are supplied by the user of the system through the keyboard. for example, $deduce(Mortal(Socrates))$ and $PutOn(A, B)$.

2. *Reasoning.* Reasoning is originated by a command from sensors and has, as a result, either the change of the information in the knowledge base or the generation of a command to an actuator. Suppose, for example, that the knowledge base contained the following statements

$$\{\forall x[Man(x) \rightarrow Mortal(x)], Man(Socrates)\}$$

   and that the program received the command (an operation of type *mem*)

$$deduce(Mortal(Socrates)),$$

   As a result, the proposition $Mortal(Socrates)$ would be added to the knowledge base.

   Assuming now, that the knowledge base contained

$$\{Block(A), Block(B), ClearTop(B)\}$$

   and that the program receives the command $PutOn(A, B)$, an operation of type *act*, whose goal is to put block A on top of block B. As a result, the program may generate a command to an actuator stating $move(A, pos(B))$, that instructs to move block A to the position of block B.

[6] The question of whether this way on interacting with the outside world has an influence of the intelligent capabilities of the system is discussed in [Kirsh 91].

3. *Actuators.* These commands are usually supplied in text form to the user of the system. For example, the command $move(A, pos(B))$ is a command to an actuator.

When we use logic as a knowledge representation formalism, knowledge is expressed as wffs. The set of wffs that exist in the system corresponds to a set, Δ, that is known as the *knowledge base*. The wffs in the knowledge base are manipulated by procedures that embody the rules of inference of the underlying logic and that contain also control information to select what inferences should be produced. These procedures that are responsible for the inference of the system correspond to the *inference engine*.

Typically, in these systems, the goal is not to prove theorems but rather to find out whether a given conclusion is derivable from the knowledge base (Δ), in other words, with these systems we are interested in finding out whether $\Delta \vdash \alpha$ and not if $\{\} \vdash \alpha$ (if α is a theorem of the system).

The knowledge base does not necessary contain propositions that are universally true (logical axioms), as it is the case of axiomatic systems, but it contains propositions that are true in a certain application domain (*proper axioms*).

When we use logic as a knowledge representation formalism we are engaged in a sort of "perpetual proof", where we are constantly evaluating whether a given proposition is derivable form the wffs in the knowledge base. The knowledge base resulting from the interaction of the system with the surrounding environment can be considered as a sequence of sets of wffs, Δ_0, Δ_1, Δ_2, ..., where Δ_0 corresponds to the initial knowledge base of the system.

In the next two sections we discuss the aspects that have been pointed out as good and bad features of using logic as a knowledge representation formalism.

2.6.1 Advantages of the use of logic

The advantages of using logic as a knowledge representation formalism are the following:

1. *It is a declarative representation.*

2. *It is a uniform representation.*

3. *It has enough expressive power.* First-order logic has enough expressive power to represent quantification, conjunction, disjunction, implication, and negation. This corresponds to an advantage over other knowledge representation formalisms, for example, certain kinds of semantic networks and frame systems, where certain of these aspects are non existing.

   However, first-order logic does not allow quantification over predicates, a feature that may be useful in certain applications, where we want to specify properties of relations. For example, it may be useful to express the following property of transitive relations:

$$\forall R[Transitive(R) \rightarrow \forall x, y, z[(R(x, y) \wedge R(y, z)) \rightarrow R(x, z)]]$$

This can be represented in second-order logic that is sound but nor complete.

Despite its expressive power, first-order logic with the traditional logical symbols, may be cumbersome to represent some sentences (for example, out of five objects, three of them have a certain property). In order to avoid this problem, new logical symbols may be added to first order logic (see Section 6.3.4);

4. *It guarantees that the inferences produced are sound.* First-order logic is sound, that means that the results produced by a system based on logic do not have mistakes;

5. *It has a formal semantics.* As we discussed in this chapter, there is a systematic way of associating the entities in the language with the entities in the object of discourse and of assigning truth values to wffs. This aspect is of paramount importance in any knowledge representation formalism because it is necessary to link the objects represented with objects in the real world. The simple way how this is achieved in logic corresponds to a big advantage in its use.

2.6.2 Disadvantages of the use of logic

The disadvantages of the use of logic as a knowledge representation formalism are the following (for a more detailed discussion see [Birnbaum 91]):

1. *It lacks ways of structuring knowledge.* It is often desirable to link and to group pieces of knowledge in order to form the description of a complex entity. This aspect is not only important from the computational point of view, but also from the abstraction perspective, under which we want to consider the representation. Traditionally, logic does not enable structuring knowledge, since it is represented by a collection of independent wffs. There are some attempts at supplying syntactical variations of wffs where structure is introduced (see [Kowalski 75] and [Sickel 76]).

2. *Monotonicity.* An aspect that has been pointed out as a drawback of logic [Minsky 75] is the fact that logic is monotonic: if Δ_1 and Δ_1 are sets of wffs, if α is a wff, and if $\Delta_1 \vdash \alpha$, then $(\Delta_1 \cup \Delta_2) \vdash \alpha$.

   Informally, monotonicity means that new information does not invalidate old conclusions. A large number of conclusions reached by human reasoning are obtained based on partial information (for example, knowing that a certain animal is a bird, we may conclude that it flies, although there numerous exceptions; if later on, we learn that it is a penguin, we will conclude that it does not fly). Minsky's objections gave rise to a new research area in AI known as non-monotonic reasoning (see Chapters 4 and 5).

3. *Semi-decidability of first-order logic.* A logic is said to be *decidable* if, given any wffs α_1, ..., α_n, e β, there is a mechanical procedure that takes a finite amount of time to decide whether $\{\alpha_1, ..., \alpha_n\} \vdash \beta$. A logic is said to be *semi-decidable* if, given any wffs α_1, ..., α_n, e β, there is a mechanical procedure that takes a finite amount of time to decide that $\{\alpha_1, ..., \alpha_n\} \vdash \beta$, but may never terminate if $\{\alpha_1, ..., \alpha_n\} \not\vdash \beta$. Propositional logic is decidable, but first-order logic is semi-decidable. This means that a reasoning system based on first-order logic may never answer to a query involving a wff β, if this wff is not derivable from the knowledge base ($\Delta \not\vdash \beta$).

4. *Exclusive treatment of deductive reasoning.* To reason is to use knowledge in order to generate new knowledge. It has been pointed out that logic only addresses deductive reasoning, making explicit knowledge that is implicit. There are other kinds of reasoning that are needed in order to characterize intelligent behavior, for example:

   (a) *Procedural reasoning.* This kind of reasoning uses procedures in order to find answers. for example, in order to compute the sum of 4 and 5, we can use the procedure "sum".

   (b) *Reasoning by analogy.* This type of reasoning is easy for humans, but very difficult to use in programming. : for example, if we are asked whether Swallows fly, we can reason as follows "Swallows are similar to blackbirds, and blackbirds fly, thus swallows fly".

   (c) *Reasoning by generalization and abstraction.* This kind of reasoning is carried out trivially by humans, and, again, is very difficult for a computer. Suppose that we know that "blackbirds have wings", "blackbirds are birds", "swallows have wings", "swallows are birds", "cucos have wings", and "cucos are birds". We may be lead to conclude that "birds have wings". The capability of producing generalizations is one of the fundamental aspects of human learning.

2.7 Summary

In this chapter we presented an introduction to classical logic, describing the motivations for the development of a logic and the coronets of a logic, the deductive system and the semantics. We discussed the two approaches for the development of the deductive system, originating natural deduction and axiomatic systems. We described how the semantics of a logic are evaluated and stated some of the properties that relate the deductive system with the semantics.

We described the role of logic in knowledge representation, by presenting the logicist approach to AI. Finally, we discussed the advantages and drawbacks of using logic as a knowledge representation formalism.

2.8 Suggested readings

A deeper study of mathematical logic can be done using one of the following books: [Church 56], [Fitch 52], [Kleene 52], [Lemon 78], [McCawley 81], [Mendelson 87], [Tarski 65], or [Whitehead and Russell 10].

In 1991 the Journal *Artificial Inteligence* devoted a special issue on the foundations of AI. In this issue there are two important papers, one by Nils Nilsson [Nilson 91], defending the logicist point of view, and an answer by Lawrence Birnbaum [Birnbaum 91] presenting the point of view of the people that do not believe in the logicist approach. The study of these papers helps understanding the arguments in favour and agains logic-based knowledge representation systems.

Additional topics regarding the material presented in this chapter can be found in:

[Brachman and Smith 80]
It presents a perspective of the positions of several researchers that work on knowledge representation. Many of the questions that are discussed (for example, Q.17, Q.31, Q.35, Q.36, Q.44) directly concern the use of logic as a knowledge representation formalism.

[Corcoran 72]
An excellent paper on the concept of validity.

[Fitch 52]
A very nice introduction to classical logic, through the use of natural deduction.

[Genesereth and Nilsson 87]
A book on the use of logic in AI, discussing, among others, the following topics: resolution, non-monotonic reasoning, induction, knowledge and belief, and meta-knowledge.

[Kneale e Kneale 1988]
A treaty about the evolution of logic.

[McCawley 81]
An introduction to logic for linguists. Its is an easy to read book, with many examples from Natural Language.

[Quine 86]
An introduction to the basic concepts of philosophical logic, discussing, in particular, the meaning of truth, the domain of logic, and the principles for the study of truths.

[Rapaport 87a, 87b]
Two papers on logic and on the role of logic in AI.

[Thayse 88]
A good introduction to logic and to the field of logic programming in AI.

[Wolfram 89]
An introduction to philosophical logic, discussing the differences between mathematical logic and philosophical logic.

2.9 Exercises

1. Represent, using logic, the following propositions. Note that the representation of some of then may be harder that it seems at first. Eventually, some of them are not representable in first-order logic. When this is the case explain why they are not representable. Note that a sentence may require more that one wff.
   (a) Only American citizens belong to FBI.
   (b) Every farmer that owns a donkey beats it.
   (c) If a relation is reflexive, the for any x and y, if x and y satisfy the relation, the y and x also satisfy the relation.
   (d) Typically, students complain about the work in the knowledge representation course, Typically, students are not right about their complaints.
   (e) John believes that he knows Mary's phone number.
   (f) Every employee receives a salary.
2. Represent the following statements using classical logic:
   (a) I saw an animal and I could not tell whether its was a cat or a dog.
   (b) Any persistent person may learn logic.
   (c) No politician is honest.
   (d) Not every bird flies.
   (e) If there is someone who is able to do so, then Anna can do it.
   (f) John hates everyone that does not hate himself.
3. One of the problems pointed out to first-order logic is the semi-decidability problem. Explain this problem and tell how severe it is.

Chapter 3

Non-classical Logics

In this chapter we describe the motivations for development of a logic and the reasons that lead to the development of non-classical logics. Afterwards, we present two examples of non-classical logics that have been used in AI, modal logic and the logic of relevant implication.

Ils n'avaint pas la même logique, l'un et l'autre. Ou peut-être ne voulait-elle pas avoir la même logique que le commisaire?
Georges Simenon, *Les scrupules de Maigret*

The concepts of derivability and validity presented in the last chapter are defined for a formal system and are only applicable to formal arguments. In our daily life, we deal with informal arguments and we are able to classify them as valid or as invalid (sometimes as good or bad arguments). In fact, we do much more than this, we consider arguments as "acceptable" or as "non-acceptable", where an acceptable argument, besides being intuitively valid present a coherent connection between premises and conclusion.

The question that we want to address in this section concerns the relationship between our intuitive notion of validity and the formal notion of validity introduced in a logic. Since logic formalizes arguments, representing them in a precise and generalizable form, in order for a logic system to be acceptable, it should present the following property: when an informal argument is translated into a corresponding formal argument, the the formal argument must be valid if and only if the corresponding informal argument is intuitively valid.

In fact, we can start the development of a formal system based on the consideration of intuitive concepts regarding the validity of informal arguments. We represent informal arguments using a symbolic representation and we develop rules of inference and a method to evaluate the truth value of propositions. We do this task with the goal that informal arguments that are considered as valid, originate formal arguments that are valid.

Using the rules of inference and the semantic system, some of the formal arguments that are classified as valid by the logic may correspond to informal arguments that are intuitively considered as invalid. In face of this situation, we can revise the rules of the system in such a way that the arguments become invalid, or, if our intuitions about the invalidity of the argument is not particularly strong, we may revise our opinion about the validity of the informal argument. Once a logic becomes well established it is likely that it will become the standard that we use to guide our intuitions about the validity and invalidity of informal arguments.

The intuitive logic that we use in our daily life to evaluate the validity or invalidity of arguments is called *logica utens.* The system that is used for the formal evaluation of arguments is called *logica docens.*[1] In Figure 3.1 we show the relationship between *logica utens* and *logica docens* and the connections that these hold with the notions of validity.

Since informal arguments contain ambiguities, the translation from informal

[1]This distinction was introduced by the English Logician Charles Pierce, who, in turn, adopted it from medieval logicians.

Figure 3.1: *Logica utens* and *logica docens.*

argument to the corresponding formal arguments entails the loss of something. We have seen that the first step towards formalizing a logic corresponds to the choice of the set of logical connectives and of a logical language. This choice will determine what formal arguments are admissible in our logic and also what is lost when we go from an informal argument to its formal counterpart. Based on this choice, the formal system is developed. The results produced by this system are compared with our intuitive notion of validity. Once the system is "tuned", it becomes the decision element about the validity or invalidity of arguments.

Different logics are constructed with the goal of capturing different aspects of informal arguments. When we develop a *non-classical logic*, a logic where the set of valid arguments does not coincide with the set of valid arguments of classical logic, there are two alternatives to consider:

1. *To extend classical logic.* According to this alternative, the logic accepts all the arguments of classical logic, plus some additional arguments. With this approach, the vocabulary of classical logic is extended in order for new notions to be representable. All the axioms, rules of inference, and theorems of classical logic still belong to these new logics. As examples of logics of this type we can mention, modal logic, epistemic logic, and erotetic (interrogative) logic.

2. *To question classical logic.* According to this alternative, the logic does not accept as valid some of the arguments that are considered valid by classical logic. With this approach, the vocabulary of the logic may be the same as the vocabulary of classical logic, but the axioms or rules of inference are different. As examples of logics of this type we can mention relevant logic, multiple-valued logic, and intuitionistic logic.

In this chapter we look into logics that were developed following each one of these approaches: modal logic, as a logic that extends classical logic; and relevance logic as a logic that questions classical logic.

3.1 Modal logic

Modal logic [Hughes and Cresswell 68], [Chellas 80] corresponds to an extension of classical logic that deals with arguments that include the notions of necessity and possibility. A proposition is necessarily true if it cannot be false under any circumstances (these propositions are also called logical truths). For example, "any person that has green eyes, has eyes". A proposition is possible if it is not necessarily true, for example, "the GPA in CS625 is greater than 3.2". The notions of "necessarily" and "possible" are called modal notions.

Modal logic is also adequate to analyze other notions such as "knowledge" and "belief" (epistemic logic), "always" and "sometimes" (temporal logic). One of the interests of modal logica for AI stems from the fact that the logic operators introduced by modal logic can be considered under other interpretations than necessarily and possibly.

In this section we look into the language of modal logic, its deductive system, its semantics, and to the interrelationship between the deductive system and the semantic of modal logic.

3.1.1 The language of propositional modal logic

The language of propositional modal logic, represented by $\mathcal{L}_{M_0}$, is defined as:

1. *Terminal symbols*:
   (a) *Punctuation symbols*: ( )
   (b) *Logical symbols*: $\neg \wedge \vee \rightarrow \Box \Diamond \bot \top$
   Symbols "$\neg$", "$\wedge$", "$\vee$", and "$\rightarrow$" have the same readings as in classical logic. The symbol "$\Box$" is read "it is necessary" and corresponds to the modal operator "necessary"; symbol "$\Diamond$" is read "it is possible" and corresponds to the modal operator "possible"; symbol "$\bot$" is read "false" and corresponds to the logical constant false; and symbol "$\top$" is read "true" and corresponds to the logical constant true.
   (c) *Predicate letters*, for $i > 0$, P_i
   A predicate letter corresponds to a symbol that can be assigned a truth value. Since the definition of the language is not concerned with meaning, we call these symbols predicate letters, rather than just predicates. The reason for parameterizing predicate letters, instead of using individual letters to denote them, is that resorting to a parameter gives us an unlimited capability of defining predicate letters. When there is no danger of confusion, we will use upper-case roman letters, P, Q, R, etc., in place of predicate letters.
2. *Well formed formulas* (wffs):
   (a) $\bot$ and $\top$ are wffs;
   (b) Predicate letters are wffs;

(c) If α is a wff, then $(\neg\alpha)$, $(\Box\alpha)$, and $(\Diamond\alpha)$ are wffs;

(d) If α and β are wffs, then $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, and $(\alpha \rightarrow \beta)$ are wffs;

(e) Nothing more is a wff.

Out of the two modal symbols, only one is really needed since one of them can be defined in terms of the other using the definition:

$$\Diamond\alpha =_{Df} \neg\Box\neg\alpha$$

This corresponds to our intuition that α is possible if $\neg\alpha$ is not necessary.

Most of the formalizations of modal logic only use a subset of these logical symbols, for example, $\neg$, $\rightarrow$, and $\Box$, the other logical symbols are treated as syntactic abbreviations of combinations of these symbols.

3.1.2 Deductive system for propositional modal logic

The most common way to define a modal logic is to resort to an axiomatic system. Since modal logic extends classical logic, all the axioms of classical logic hold for modal logic. For this reason, when we present a modal logic, we consider the axioms of classical logic (for example, of the system L of [Mendelson 87]) to which we add specific axioms for modal logic.

We will not discuss here a formalization of modal logic using natural deduction, an example of this can be found in [Huges and Cresswell 68, pp. 331-339] or [Fitch 52, pp. 64-80 and 164-166].

Depending on the modal axioms used, we will have different kinds of modal logics. The names of these axioms are usually used to identify the names of the logics that they originate. For example, if we use only axiom K, we originate logic K.

The first axiom that we consider, is axiom K or distribution axiom.

$$K: \quad \Box(\alpha \rightarrow \beta) \rightarrow (\Box\alpha \rightarrow \Box\beta)$$

You should notice that this axiom uses meta-variables, whose domain are the elements of $\mathcal{L}_{M_0}$. In fact, it corresponds to an infinite (but enumerable) collection of axioms. The same will be true for all modal axioms that we present.

There are two rules of inference in modal logic, modus ponens and necessitation:

- *Modus ponens.* From α and $\alpha \rightarrow \beta$ we can infer β.

- *Necessitation.* If α is a theorem of classical logic, then we can infer $\Box\alpha$.

A modal logic that uses these rules of inference, the axioms of classical logic and the axiom of distribution, is called *normal modal logic* or *normal modal system.* This is designated as system K.

There are several modal logics, depending on the axioms that they use. The following are the most commonly used axioms:

$$\begin{array}{rl} T: & (\Box\alpha) \rightarrow \alpha \\ D: & (\Box\alpha) \rightarrow (\Diamond\alpha) \\ 4: & (\Box\alpha) \rightarrow (\Box\Box\alpha) \\ 5: & (\Diamond\alpha) \rightarrow (\Box\Diamond\alpha) \\ B: & \alpha \rightarrow (\Box\Diamond\alpha) \end{array}$$

The reason why there are symbols associated with these axioms ("T", "D", "4", etc) is that these symbols are used to define the names of the different modal systems.[2] For example, the system $KD5$ is the modal system that contains axioms K, D, and 5.

The choice of the axioms depends on the uses one wants to make with the system. For example, axiom T is used in systems that model the notion of knowledge, considered as absolute knowledge. In this case, "$\Box$" is read "I know" and axiom T states that "if I know α, then α is true". It should be clear that if we would like to model the notion of belief, this axiom should not be used, since it would mean "if I believe in α, then α is true", which contradicts the intuitive notion of belief.

Similarly to what happened with classical logic, the wffs that are derivable in a logic depend on the deductive system that is used (rules of inference and axioms). In classical logic we do not index the derivability symbol, "$\vdash$", with the indication of the deductive system because all the systems are equivalent. In modal logic, however, the deductive system makes a difference and we associate the symbol "$\vdash$" with an identification of the deductive system. for example, $\Delta \vdash_{KD5} \alpha$ means that "α is derivable from Δ, in the system $KD5$". In Figure 3.2 we show the proof of the theorem $(\Box(P \wedge Q)) \rightarrow (\Box P \wedge \Box Q)$ in the system K.[3]

3.1.3 Semantics for propositional modal logic

The semantics of modal logic is based on the notion of possible worlds, introduced by Leibnitz and formalized by [Kripke 63]. Informally, we use a set of possible worlds and a accessibility relation between possible worlds. A wff of the form $\Box\alpha$ is true in a given world, if α is true in every world that is accessible from that world; a wff of the form $\Diamond\alpha$ is true in a given world, if α is true at least in one world accessible from that world.

We define a *Kripke structure* (usually refereed just by *structure*) as a pair (W, R), where W is a set of worlds, the possible worlds (each word corresponds, roughly, to a conceptualization — note that we are just considering propositional modal logic) and R is a relation between worlds, the *accessibility relation*. We write $w_1 R w_2$ to say that the world w_2 is accessible from the world w_1.

[2] This notation is due to [Chellas 80].

[3] In this figure, $TL7$, $TL8$, and $TL10$ refer to theorems in the system L of [Mendelson 87]. These were presented in Chapter 2.

1	$(P \wedge Q) \rightarrow P$	$TL7$
2	$\Box((P \wedge Q) \rightarrow P)$	$Necessitation(1)$
3	$\Box((P \wedge Q) \rightarrow P) \rightarrow$ $(\Box(P \wedge Q) \rightarrow \Box P)$	K
4	$\Box(P \wedge Q) \rightarrow \Box P$	$MP(2,3)$
5	$(P \wedge Q) \rightarrow Q$	$TL8$
6	$\Box((P \wedge Q) \rightarrow Q)$	$Necessitation(5)$
7	$\Box((P \wedge Q) \rightarrow Q) \rightarrow$ $(\Box(P \wedge Q) \rightarrow \Box Q)$	K
8	$\Box(P \wedge Q) \rightarrow \Box Q$	$MP(6,7)$
9	$(\Box(P \wedge Q) \rightarrow \Box P) \rightarrow$ $((\Box(P \wedge Q) \rightarrow \Box Q) \rightarrow (\Box(P \wedge Q)$ $\rightarrow (\Box P \wedge \Box Q)))$	$TL10$
10	$(\Box(P \wedge Q) \rightarrow \Box Q) \rightarrow (\Box(P \wedge Q)$ $\rightarrow (\Box P \wedge \Box Q))$	$MP(4,9)$
11	$(\Box(P \wedge Q)) \rightarrow (\Box P \wedge \Box Q)$	$MP(8,10)$

Figure 3.2: Proof of $\{\} \vdash_K (\Box(P \wedge Q)) \rightarrow (\Box P \wedge \Box Q)$.

We define the function V, that has, as arguments, a world and a predicate letter and assigns one of the values true or false to that predicate letter (the function V roughly corresponds to an interpretation for predicate letters — from a given world, it evaluates the truth value of a wff corresponding to a predicate letter). For example, $V(w, \alpha) = \top$, means that α is true in the world w.

A *model* is defined as a triple $<W, R, V>$, where W, R, and V have the following meanings:

1. W is a set of possible worlds;
2. $R \subset W \times W$ is the accessibility relation;
3. $V : W \times \mathcal{L}_{M_0} \Rightarrow \{\top, \perp\}$ is the evaluation function.

A model $<W, R, V>$ determines, for each of the worlds in W, which wffs are true in that world. We write $<W, R, V> \models_w \alpha$ to mean that the wff α is true in the world w, in model $<W, R, V>$.

The relationship between models and wffs is defined as:

1. $<W, R, V> \models_w \top$, for all $w \in W$, that is, $\top$ is true in every world;
2. $<W, R, V> \not\models_w \perp$, for all $w \in W$, that is, $\perp$ is not true in any world;
3. $<W, R, V> \models_w \alpha$ if $V(w, \alpha) = \top$, where α is a predicate letter ($\alpha \in \mathcal{L}_{M_0}$);
4. $<W, R, V> \models_w \neg\alpha$ if $<W, R, V> \not\models_w \alpha$;

5. $<W, R, V> \models_w \alpha \rightarrow \beta$ if $<W, R, V> \models_w \alpha$ when $<W, R, V> \models_w \beta$, where α and β are wffs;

6. $<W, R, V> \models_w \Box\alpha$, if for every $x \in W$, such that wRx, we have $<W, R, V> \models_x \alpha$, where α is a wff.

From this definition, we can prove that $<W, R, V> \models_w \Diamond\alpha$, if there is at least one world $x \in W$, such that wRx and $<W, R, V> \models_x \alpha$. You should note that we have not defined the truth value for wffs containing conjunction ($\wedge$) or disjunction ($\vee$) since these are considered as syntactic abbreviations of wffs containing negation ($\neg$) and implication ($\rightarrow$).

We say that the wff α is *true in the model* $<W, R, V>$ (also said *valid in the model*, which should not be confused with the notion of validity defined for arguments) if, for all $w \in W$, we have $<W, R, V> \models_w \alpha$, that is, if α is true in every world of the model. We say that a modal wff α is true in a structure (W, R) if it is true in every model $<W, R, V>$ based on that structure.

3.1.4 The deductive system and the semantics

In this section, we discuss the relationship between the deductive system and the semantics. In Section 3.1.2 we mention several modal systems that differ from one another by the axioms that they use. In this section we show that it is possible to semantically characterize these systems, in the sense that we can specify structures (W, R) in such a way that the wffs that are true in the different structures correspond to the wffs that are derivable in the different systems. This is done resorting to the properties of the accessibility relation R.

Let us start with the axiom K, $\Box(\alpha \rightarrow \beta) \rightarrow (\Box\alpha \rightarrow \Box\beta)$, and let us discuss which conditions must be imposed on the accessibility relation so that it becomes false. The definition of $\rightarrow$ imposes that, for this axiom to be false, both $\Box(\alpha \rightarrow \beta)$ and $\Box\alpha$ must be true in a given world, w_i, and $\Box\beta$ must be false in that world. By definition, both $\alpha \rightarrow \beta$ and α must be true in every world w accessible from w_i and β must be false in at least one of those worlds.

Let us suppose that there is a world, w_k, accessible from w_i, where α and $\alpha \rightarrow \beta$ are true and β was false. Since the theorems of classical logic are true in w_k (and, in fact, in every world, since modal logic extends classical logic), β must be true in that world, and this w_k cannot exist. This means that axiom K is always true, independently on the properties of the accessibility relation.

Let us now consider axiom T ($\Box\alpha \rightarrow \alpha$). In order for T to be false in world w_i, $\Box\alpha$ must be true in w_i and α must be false in that world. In order for $\Box\alpha$ to be true in w_i, α must be true in every world accessible from w_i. Thus, in order for $\Box\alpha$ to be true in w_i and α false in w_i, the world w_i may not be accessible from itself, in other words, the accessibility relation cannot be reflexive (Figure 3.3).

An alternative proof of this fact is the following: Let us suppose that the accessibility relation is reflexive, in other words $\forall w \in W : wRw$. If $\Box\alpha$ is true in world w_i then, by definition, α is true in every world accessible from w_i. Since the accessibility relation is reflexive, α is true in w_i and thus $\Box\alpha \rightarrow \alpha$.

Figure 3.3: $\Box\alpha$ is true in w_i and α is false in w_i.

Therefore axiom T holds in all the structure where the accessibility relation is reflexive.

A possible application of a non-reflexive accessibility relation is the relation "later than", that give the operator "$\Box$" the interpretation "will be true and will remain true".

In a similar way, we can prove the following results:

1. $\Box\alpha \rightarrow \Diamond\alpha$ is true in every structure whose accessibility relation is linear, that is, structures whose accessibility relation verifies the condition $\forall w_1 \in W(\exists w_2 \in W : w_1 R w_2)$.

2. $\Box\alpha \rightarrow \Box\Box\alpha$ is true in every structure whose accessibility relation is transitive, that is, structures whose accessibility relation verifies the condition $\forall w_1, w_2, w_3 \in W$ if $w_1 R w_2$ and $w_2 R w_3$ then $w_1 R w_3$.

3. $\Diamond\alpha \rightarrow \Box\Diamond\alpha$ is true in every structure whose accessibility relation is Euclidean, that is, structures whose accessibility relation verifies the condition $\forall w_1, w_2, w_3 \in W$ if $w_1 R w_2$ and $w_1 R w_3$ then $w_2 R w_3$.

4. $\alpha \rightarrow \Box\Diamond\alpha$ is true in every structure whose accessibility relation is symmetric, that is, structures whose accessibility relation verifies the condition $\forall w_1, w_2 \in W$ if $w_1 R w_2$ then $w_2 R w_1$.

It is common to characterize the structures in a modal system by the properties of its accessibility relation. For example, we say that a structure (W, R) is transitive if its accessibility relation R is transitive.

For the different modal systems soundness and completeness results can be shown. For example, for the system $KT4$, the soundness proof corresponds to proving that if $\{\} \vdash_{KT4} \alpha$, that is, if α is derivable in this system, then α is true in every structure that is reflexive and transitive. The completeness proof corresponds to the converse proof.

We conclude this section by reminding that the nature of a structure completely defines each one of the modal systems.

3.1.5 The language for first-order modal logic

The modal logic that we discussed up to now did not contemplate the existence of variables. Similarly to classical logic, the move from propositional modal logic into first-order modal logic requires that the rules for the formation of wffs be revised (taking into account more symbols) and the introduction of new rules of inference.

In first-order modal logic, we define wffs from the following components:

1. *Terminal symbols.*
   (a) *Punctuation symbols*: , ( ) [ ]
   (b) *Logical symbols*: $\neg \wedge \vee \rightarrow \Box \Diamond \forall \exists$
   (c) *Function letters* with n arguments, for $n \geq 0$ and $i \geq 1 : f_i^n$.
   Again, the reason for parameterizing the function letters, instead of using individual letters to denote them, stems from the fact that resorting to a parameter gives a an unlimited capability of defining function names. A function letter denotes a function over elements of the language. Functions with zero arguments (corresponding to f_i^0) correspond to constants.
   When there is no danger of confusion, function letters will be denoted by lower-case roman identifiers,[4] f, g, h, $above$, $father$, etc.
   (d) *Predicate letters* with n arguments, for $n \geq 0$ and $i \geq 1$: P_i^n.
   A predicate letter represents an operator over the elements of the language, that has a truth value. When there is no danger of confusion, we will use identifiers beginning with an upper-case roman letter, P, Q, R, $ClearTop$, $Human$, etc., to denote predicate letters.
   (e) *Individual variables*, for $i \geq 1 : x_i$
   Individual variables, have as domain the objects of our conceptualization. When there is no danger of confusion, we will use lower-case roman letters, x, y, z, etc., to denote individual variables.
2. *Terms.* These are recursively defined, as follows:
   (a) Each function letter with zero arguments is a term;
   (b) Each variable is a term;
   (c) If t_1, t_2, ..., t_n are terms. then $f_i^n(t_1, t_2, \ldots, t_n)$ is a term;
   (d) Nothing more is a term.
3. *Well formed formulas* (wffs):

[4] Sequences of one or more letters.

(a) If $t_1, t_2, \ldots t_n$ are terms, then $P_i^n(t_1, t_2, \ldots, t_n)$ is a wff (this is called an *atomic wff*);

(b) If α is a wff, then $(\neg\alpha)$, $(\Box\alpha)$, and $(\Diamond\alpha)$ are wffs;

(c) If α and β are wffs, then $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, and $(\alpha \rightarrow \beta)$ are;

(d) If α is a wff, containing zero or more occurrences of the individual variable x, then $\forall x[\alpha]$ and $\exists x[\alpha]$ are wffs;

(e) Nothing more is a wff.

The language of first-order modal logic is represented by $\mathcal{L}_{M_1}$.

3.1.6 The deductive system for first-order modal logic

One of the problems that fist.order modal logic has to address concerns the correspondence between entities existing in the different possible worlds, as well as correspondence between the functions applied to these entities. Some of the possible alternatives (that will not be discussed in detail in this book) are: the possibility to create or to destroy entities in the different worlds, the possibility of existence of functions (the rigid designators) that remain unchanged in the different worlds and their identification.

Again, similarly to classical logic, the move from propositional into first-order modal logic requires the introduction of new axioms and new rules of inference.

The most commonly used axioms in first-order modal logic, known as *Barcan formulas* are the following:

$$\begin{array}{ll}
B1: & \Box(\forall x[P(x)]) \rightarrow \forall x[\Box P(x)] \\
B2: & \forall x[\Box P(x)] \rightarrow \Box(\forall x[P(x)]) \\
B3: & \exists x[\Box P(x)] \rightarrow \Box(\exists x[P(x)]) \\
B4: & \Box(\exists x[P(x)]) \rightarrow \exists x[\Box P(x)]
\end{array}$$

In order to get a intuitive idea about what is stated in these axioms, lest us look into the meaning of the formulas that compose them, under the perspective of the possible worlds, we obtain the following "interpretations":

- The wff $\Box(\forall x[P(x)])$ states that in every world that is accessible from the current world, the property P holds for every entity;

- The wff $\forall x[\Box P(x)]$ states that every entity in the current world satisfies the property P in every world that is accessible from the current world;

- The wff $\exists x[\Box P(x)]$ states hat there is an entity in the current world that satisfies the property P in every world that is accessible from the current world;

- The wff $\Box(\exists x[P(x)])$ states that in every world that is accessible from the current world, property P holds for some entity.

Under this perspective, axiom $B1$, for example, states that if in every world, accessible from the current world, the property P holds for every object, then all the objects in the current world satisfy property P in all the worlds that are accessible from it.

Again, in first-order modal logic. there is a relationship between these axioms and the accessibility relation between worlds (this is discussed in [Ramsay 88]).

3.2 Logic of relevant implication

As an example of a logic that questions classical logic, we present the logic of relevant implication of [Anderson and Belnap 75]. Classical logic does not consider relevance issues when it evaluates the validity of arguments. The irrelevancy between premises and conclusions is not well accepted in common reasoning and is considered a flaw in one's argument, as discussed in [Anderson and Belnap 75, pp.. 17-18]:

> "A mathematician writes a paper on Banach spaces and ... concludes with a conjecture. As a footnote to the conjecture he writes: "In addition to its intrinsic interest, this conjecture has connections with other parts of mathematics which might not immediately occur to the reader. For example, if the conjecture is true, then the first order functional calculus is complete; whereas if it is false, then it implies that Fermat's last conjecture is correct" ... the first order functional calculus is complete, and necessarily so, so anything implies that fact - and if the conjecture is false it is presumably impossible, and hence it implies anything ... the editor counters: "I understand the technical bit all right, but it is simply not correct ... the standards maintained by this journal require that the antecedent of an 'if...then-' statement must be relevant to the conclusion drawn ... " [Anderson and Belnap 75, pp.17-18]

Reacting against the lack of account of relevance in classical logic, Anderson and Belnap proposed a new logical system, the *relevance logic* [Anderson and Belnap 75], which challenges classical logic in the following two aspects:

1. *Regarding the classical concept of validity*: Anderson and Belnap argue that if one proposition entails another then there must be an element of causality that relevantly connects them, and, for that reason, they do not recognize some of the arguments classified as valid by classical logic. In particular, they explicitly deny [Anderson and Belnap 75, p.3] the so-called paradoxes of implication: $\alpha \rightarrow (\beta \rightarrow \alpha)$, anything implies a true proposition; and $(\alpha \wedge \neg\alpha) \rightarrow \beta$, a contradiction implies anything.

   To their (semantic) notion of entailment corresponds a (syntactic) notion of deducibility according to which β is deducible from α only if the derivation of β genuinely uses, and does not simply take a detour via, α.

To define deducibility, Anderson and Belnap modify the classical rules of natural deduction. First, they associate each well formed formula (wff) in the line of a derivation with a set which references all hypotheses actually used in its derivation. Second they restate the classical rules of inference to take these sets into account, blocking the irrelevant applications which are allowed in classical logic.

2. *Regarding the applicability of the disjunctive syllogism*: Anderson and Belnap (pp. 165-167) distinguish two kinds of "or", the truth functional and the intensional. In the *truth functional sense*, the truth value of the formula $\alpha \vee \beta$ is defined *only* in terms of the truth values of α and β. For this sense of "or", the classical rule of or-introduction (from α we can derive either $\alpha \vee \beta$ or $\beta \vee \alpha$ regardless of the truth value of β or its connection with α) is applicable. According to Anderson and Belnap (p.176), the *intensional sense* which assumes that the disjuncts are relevant to each other, corresponds to the way "or" is used in everyday language, meaning something like "if it isn't one, then it is the other". This case clearly involves an entailment (or better, two entailments), and disjunctive syllogism (from $\alpha \vee \beta$ and $\neg\alpha$, infer β) can be explained in terms of modus ponens.

   Anderson and Belnap argue that classical logic confuses the two senses of "or", using the first in the rule of or-introduction and the second in the rule of or-elimination. For this reason, they consider disjunctive syllogism a "fallacy of relevance" (p.165) and explicitly deny that it is a valid rule of inference.

We discuss the rules of inference of a system for propositional relevance logic, using natural deduction, system FR [Anderson and Belnap 75, pp. 346-348]. We will not discuss the semantics of relevance logic (this can be found in [Read 88]).

FR system deal with extended wffs, that belong to the language $\mathcal{L}^+$ which is defined as:

$$\mathcal{L}^+ = \mathcal{L}_0 \times \{,\} \times 2^N$$

Each wff of $\mathcal{L}^+$ is associated with a set of identifiers; if α is a wff and $a \in N$ (where N represents a set of identifiers), an extended wff is of the form α, a.

The following are the rules of inference of FR:

FR1 *Hypothesis* (Hyp): This rule of inference enables the introduction, as a hypothesis, of any wff in any point of a proof. This hypothesis is associated with an identifier ($k \in N$) that never appeared in the proof before and initiates a new sub-proof:

$$\begin{array}{r|ll} n & \alpha, \{k\} & Hyp(n) \\ n+1 & & \end{array}$$

FR2 *Repetition*(*Rep*): Any wff may be repeated in a proof, keeping its identifier set:

$$\begin{array}{lll} n & \alpha, a & \\ & \ldots & \\ m & \alpha, a & Rep(n) \end{array}$$

FR3 *Re-iteration* (*Reit*): Any wff may be re-iterated (repeated) in a sub-proof contained in the proof where the wff exists, keeping its identifier set:

$$\begin{array}{llll} n & \alpha, a & & \\ & \ldots & & \\ m & \quad | & \alpha, a & Reit(n) \end{array}$$

FR4 *Implication introduction* ($\rightarrow I$): If, in a proof initiated with the hypothesis $\alpha, \{k\}$, we can derive $\beta, a \cup \{k\}$, we can infer $\alpha \rightarrow \beta, a$ in the proof that contains the proof initiated by the hypothesis $\alpha, \{k\}$:

$$\begin{array}{llll} n & | & \alpha, \{k\} & Hyp \\ & | & \ldots & \\ m & | & \beta, a \cup \{k\} & \\ k & & \alpha \rightarrow \beta, a & \rightarrow I(n, m) \end{array}$$

FR5 *Implication elimination*, also known as *modus ponens* ($\rightarrow E$): If, in a proof, we have both α, a and $\alpha \rightarrow \beta, b$, we can infer $\beta, a \cup b$:

$$\begin{array}{lll} n & \alpha, a & \\ & \ldots & \\ m & \alpha \rightarrow \beta, a & \\ k & \beta, a \cup b & \rightarrow E(n, m) \end{array}$$

FR6 *Negation introduction* ($\neg I$): From the extended wff $\alpha \rightarrow \neg\alpha, a$, we can derive $\neg\alpha, a$:

$$\begin{array}{lll} n & \alpha \rightarrow \neg\alpha, a & \\ l & \neg\alpha, a & \neg I(n) \end{array}$$

FR7 *Negation introduction* ($\neg E$): From $\neg\beta, a$ and $\alpha \rightarrow \beta, b$, we can infer $\neg\alpha, a \cup b$:

$$\begin{array}{lll} n & \neg\beta, a & \\ m & \alpha \rightarrow \beta, b & \\ k & \neg\alpha, a \cup b & \neg E(n, m) \end{array}$$

FR8 *Double negation introduction* ($\neg\neg I$): If, in a proof, we have α, a, then we can infer $\neg\neg\alpha, a$:

$$\begin{array}{lll} n & \alpha, a & \\ l & \neg\neg\alpha, a & \neg\neg(n)I \end{array}$$

FR9 *Double negation elimination* ($\neg\neg E$): If, in a proof, we have $\neg\neg\alpha, a$, then we can infer α, a:

$$\begin{array}{lll} n & \neg\neg\alpha, a & \\ l & \alpha, a & \neg\neg E(n) \end{array}$$

FR10 *Conjunction introduction* ($\wedge I$): If, in a proof, we have both α, a and β, a, then we can infer $\alpha \wedge \beta, a$:

$$\begin{array}{lll} n & \alpha, a & \\ & \ldots & \\ m & \beta, a & \\ k & \alpha \wedge \beta, a & \wedge I(n, m) \end{array}$$

FR11 *Conjunction elimination* ($\wedge E$): If, in a proof we have $\alpha \wedge \beta, a$, then we can infer either α, a, or β, a, or both:

$$\begin{array}{lll} n & \alpha \wedge \beta, a & \\ m & \alpha, a & \wedge E(n) \end{array}$$

$$\begin{array}{lll} n & \alpha \wedge \beta, a & \\ m & \beta, a & \wedge E(n) \end{array}$$

FR11 *Introduction of disjunction* ($\vee I$): If in a proof we have α, a, then we can infer $\alpha \vee \beta, a$, or $\beta \vee \alpha, a$, or both, where β is any wff:

$$\begin{array}{lll} n & \alpha, a & \\ m & \alpha \vee \beta, a & \vee I(n) \end{array}$$

$$\begin{array}{lll} n & \alpha, a & \\ m & \beta \vee \alpha, a & \vee I(n) \end{array}$$

FR13 *Disjunction elimination* ($\vee E$): If, in a proof, we have (a) the extended wff $\alpha \vee \beta, a$; (b) a proof initiated by the hypothesis $\alpha, \{k\}$, having $\gamma, b \cup \{k\}$ as one of its extended wffs;and (c) A proof initiated by the hypothesis $\beta, \{k+1\}$, having $\gamma, b \cup \{k+1\}$ as one of its extended wffs, then, we can infer $\gamma, a \cup b$ in that proof:

$$\begin{array}{llll} n & & \alpha \vee \beta, a & \\ o & | & \alpha, \{k\} & Hyp \\ & | & \ldots & \\ p & | & \gamma, a \cup \{k\} & \\ & \ldots & & \\ r & | & \beta, \{k+1\} & Hyp \\ & | & \ldots & \\ s & | & \gamma, b \cup \{k+1\} & \\ m & & \gamma, a \cup b & \vee E(n, (o, p), (r, s)) \end{array}$$

In Figure 3.4 we show the proof of $(P \to (Q \to R)) \to ((P \wedge Q) \to R)$ in the FR system.

In relevant logic the wff $(\alpha \wedge \neg\alpha) \to \beta$ is not a theorem. Logics with this property are called *paraconsistent logics* [Jeffreys 42], [Tennant 84].

1	|	$P \rightarrow (Q \rightarrow R), \{1\}$	Hyp
2	| |	$P \wedge Q, \{2\}$	Hyp
3	| |	$P, \{2\}$	$\wedge E(2)$
4	| |	$P \rightarrow (Q \rightarrow R), \{1\}$	$Reit$ (1)
5	| |	$Q \rightarrow R, \{1, 2\}$	$\rightarrow E(3, 4)$
6	| |	$Q, \{2\}$	$\wedge E(2)$
7	| |	$Q \rightarrow R, \{1, 2\}$	$Rep(5)$
8	| |	$R, \{1, 2\}$	$\rightarrow E(6, 7)$
9	|	$(P \wedge Q) \rightarrow R, \{1\}$	$\rightarrow I(2, 8)$
10		$((P \rightarrow (Q \rightarrow R)) \rightarrow ((P \wedge Q) \rightarrow R)), \{\}$	$\rightarrow I(1, 9)$

Figure 3.4: Proof of $(P \rightarrow (Q \rightarrow R)) \rightarrow ((P \wedge Q) \rightarrow R)$ in FR.

3.3 Summary

In this chapter we discusses non-classical logics, presenting the motivations for their development and discussing two logics that have been used in AI, modal logic and the logic of relevant implication.

Modal logic has been used in AI to deal with knowledge and belief (for example logic auto-epistemic [Moore 83, 84, 85] and the logic of [Konolige 86]) and in logics that deal with changing world (for example , TWEAK of [Chapman 87]).

Logic of relevant implication can be used to keep track of dependencies among propositions (for example, in the belief revision system of [Martins and Shapiro 88]) or in systems that are not affected by contradictions [Joy, Murty and Krishna 90].

3.4 Suggested readings

A good perspective on the work on non-classical logics can be found in [Rautenberg 87]. The following references can provide additional information about the topics presented in this chapter:

[Haack 78]
An outstanding introduction to the problems associated with philosophical logic and with non-classical logics.

[Hughes and Cresswell 68]
A good perspective on modal logic.

[Ramsay 88]
An introduction to the use of non-classical logics in AI, discussing modal logic, temporal logic, and non-monotonic logics.

[Turner 84]
Discusses some of the non-classical logics used in AI. In particular, it addresses modal logic, dynamic logic, intuitionistic logic, temporal logic, fuzzy logic, and non-monotonic logics.

[Wolfram 89]
An introduction to philosophical logic, discussing the differences of approach between mathematical logic and philosophical logic.

3.5 Exercises

1. Prove that $\Box\alpha \rightarrow \Diamond\alpha$ is true in every structure whose accessibility relation is linear, that is, structures whose accessibility relation verifies the condition $\forall w_1 \in W(\exists w_2 \in W : w_1 R w_2)$.

2. Prove that $\Box\alpha \rightarrow \Box\Box\alpha$ is true in every structure whose accessibility relation is transitive, that is, structures whose accessibility relation verifies the condition $\forall w_1, w_2, w_3 \in W$ if $w_1 R w_2$ and $w_2 R w_3$ then $w_1 R w_3$.

3. Prove that $\Diamond\alpha \rightarrow \Box\Diamond\alpha$ is true in every structure whose accessibility relation is Euclidean, that is, structures whose accessibility relation verifies the condition $\forall w_1, w_2, w_3 \in W$ if $w_1 R w_2$ and $w_1 R w_3$ then $w_2 R w_3$.

4. Prove that $\alpha \rightarrow \Box\Diamond\alpha$ is true in every structure whose accessibility relation is symmetric, that is, structures whose accessibility relation verifies the condition $\forall w_1, w_2 \in W$ if $w_1 R w_2$ then $w_2 R w_1$.

Chapter 4

Non-Monotonic Logics

1. Basic principles
2. Default Logic
3. Other approaches
4. Objections to non-monotonic logics
5. Summary
6. Suggested readings
7. Exercises

In this chapter, we present the motivations and the principles that lead to the development of non-monotonic logics, that is, logics that allow jumping to conclusions based on incomplete information. We discuss one logic of this type, Default Logic, and present the main objections to non-monotonic logics that have been pointed out.

"Every rule has an exception."
Portuguese saying

AI studies methods to allow computers to perform tasks that, when done by humans, are said to require intelligence. One of these tasks is, clearly, reasoning. A characteristic of human reasoning is the ability to deal with incomplete information. In our daily life, we are constantly faced with situations in which only part of the information is known. This fact does not prevent us from producing conclusions, even when the available information is not enough to guarantee the correctness of these conclusions. The kind of reasoning in which some of the conclusions may have to be revised in face of additional information is called *non-monotonic reasoning* ([Minsky 75]).

Whereas the basic concept underlying reasoning with complete information (the kind of reasoning corresponding to traditional logics) is the concept of truth, non-monotonic reasoning is based on the concept of rationality. We cannot expect all of our conclusions to be true; nevertheless, we do not want to produce any conclusion whatsoever. To minimize the risk of producing wrong conclusions, we use our knowledge and experience to decide whether a conclusion should or should not be produced. Although many of our conclusions may turn out to be wrong, we require them to be rational. The existence of a rational justification for a conclusion is a requirement for a conclusion produced by humans.

Rationality is a vague concept, whose exact definition may be impossible to produce. As opposed to the concept of truth, rationality has two particular characteristics: (1) it depends on the agent that is reasoning; in other words, two rational agents may not agree with what is rationally produced in a given situation; (2) it depends on how the conclusion is to be used.

Donald Perlis tried to characterize a rational conclusion through the following definition [Perlis 87]: "A proposition α is believed by an agent A, or A considers α as a rational conclusion, if A is prepared to use α as if it were true".

We will try to characterize the way rational conclusions may be obtained. In order to do that, we should stress that human reasoning is based both on the presence of information and also on its absence. A rational conclusion is frequently connected to inference patterns of the form: "from α, lacking β, we can conclude γ". For example, knowing that a given animal is a bird, and lacking information about the fact that it does not fly, we may conclude that it does fly. The intuition behind this pattern of reasoning says that α supports the conclusion γ, while the lack of β guarantees its rationality.

We should notice two things regarding this reasoning pattern: (1) The lack of information is usually identified with the absence of contradictory information. In this case, the pattern is written as "from α, in the lack of contradictory information, we may conclude γ". (2) Our discussion of this reasoning pattern lacks rigor (and thus the reason why we called it "reasoning pattern" and not inference rule or proposition).[1] Our discussion leaves unanswered two important questions: "How are reasoning patterns represented?" and "What are the conditions for a given proposition to be believed?".

Non-monotonic reasoning is a fairly recent area of AI and has attracted a considerable interest ([Bobrow 80], [Ginsberg 87], [Perlis 87], [Reiter 88]). There are two important issues associated with non-monotonic reasoning: the formalization of reasoning and the process of belief revision. The first problem addresses the ways of formalizing the reasoning pattern that we discussed and the reasoning process associated with it. The second addresses the issue of how to update information about the world when additional information clashes with previously produced conclusions.

In this chapter, we present a logic, Default Logic, that enables us to draw rational conclusions from a set of premises (conclusions that are more than the conclusions that can be drawn using traditional logic) and that has mechanisms to invalidate conclusions in the presence of additional information.

4.1 Basic principles

One of the properties of classical logic is monotonicity; that is, the conclusions that can be drawn from a set of premises are never invalidated if the set of premises increases. There are many situations in which we want to produce conclusions that may have to be discarded in the face of future information. This aspect of our reasoning is very desirable, because if we could only produce conclusions that were true, we would not be able to act upon the world. In fact, most of the time, we do not have complete information about what is the state of the world or what are the consequences of our actions.

Since AI is concerned with the construction of programs that exhibit intelligent behavior, it has been concerned with the study of reasoning whose conclusions are more than just the logical consequences (in the classical sense) of a set of premises. Non-monotonic logics are an attempt to formalize this kind of reasoning. This reasoning is usually associated with sentences of the forms:

In general, A;
Typically, A;
In the absence of the contrary, assume A.

For example, given the sentence "In general, birds fly", when we hear about a given bird, say Tweety, we may be led to the conclusion that Tweety flies, although there are a large number of exceptions to this rule: penguins, ostriches, newborn birds, dead birds, and so on. We should notice that our conclusion about Tweety flying is based not only on the information that in general birds fly and that Tweety is a bird, but also on the assumption that Tweety is a

[1] These two alternatives, "rule of inference" and "proposition", have been used in different logics to represent this reasoning pattern.

normal bird with respect to flying. This assumption is based on the *lack* of information about the abnormality of Tweety. If later on we learn that, for some reason, Tweety is abnormal with respect to flying, we must withdraw the conclusion that Tweety flies.

We should notice the difference between the sentence "In general, birds fly" and a sentence with a universal quantifier, for example,"All birds are mortal". In the latter case, we can conclude that Tweety is mortal, and this conclusion is not revisable: no matter what we learn, we will never abandon the conclusion that Tweety is mortal.

We could try to use classical logic to represent the sentence that in general birds fly. One attempt corresponds to the following wff:

$$\forall x[(Bird(x) \wedge \neg Abnormal(x)) \rightarrow Flies(x)],$$

which states that all birds that are not abnormal (with respect to flying) do fly. We must now define what we mean by being abnormal with respect to flying, which we do with the following wff:

$$\forall x[(Penguim(x) \vee Ostrich(x) \vee Dead(x) \vee) ... \rightarrow Abnormal(x)].$$

The "..." in our wff indicates our inability to exhaustively enumerate all the possible conditions that may lead us to conclude the abnormality of a bird. However, even if we could come up with all these conditions, we could not conclude anything from the fact that Tweety is a bird since there is not enough information to decide its normality or abnormality. The goal of non-monotonic reasoning is to develop mechanisms that enable us to "jump" to rational conclusions from incomplete information.

When we develop non-monotonic logics, we are opening the possibility of producing conclusions that are not true (in other words, we are accepting arguments that are not valid). Under the perspective of logic, we want to be able to infer propositions that are consistent with the premises, that is, propositions that are true in at least one of the models of the premises. For example, consider the set of premises:

$$\{\text{Tweety is a bird, typically birds fly}\}.$$

The proposition "Tweety flies" is consistent with this set, since it is true in at least one model of the premises (it belongs to a state of the world that we can conceive, based on these premises). On the other hand, "Tweety does not fly" is also consistent with this set of premises. However, "Tweety flies" and "Tweety does not fly" cannot be inferred simultaneously.

Non-monotonic logics enable us to infer propositions that are consistent with the set of premises and that are mutually consistent. In general, the propositions that are inferred depend on the order of application of the rules of inference. In our example, if we infer that "Tweety flies", we cannot further infer that "Tweety does not fly", and viceversa.

Monotonic inference, the kind of inference associated with traditional logics, can be seen as the mechanical application of all the inference rules in all possible ways to the premises and to the propositions generated from them. Once a proposition is generated, it is never removed. This process enables us to enumerate all the theorems of a logic. Whenever there is a mechanical procedure to generate all the elements of a set, the set is said to be *recursively enumerable* [Machtey and Young 78].

On the other hand, non-monotonic inference, the kind of inference associated with non-monotonic logic, does not guarantee that a given proposition, once derived, will remain in all future steps, because a proposition inferred in a further step may invalidate it. This means that the set of theorems of a non-monotonic logic is not recursively enumerable. Given a set of premises we are interested in computing the so-called extensions, in a non-monotonic logic. Intuitively, given a set of premises Δ and a non-monotonic formalism, an *extension*, Ω, of Δ in that formalism is a set of propositions that contains all the consequences of Δ in the classical sense and is closed under certain conditions: Extensions are fixed points with respect to the application of rules of inference. A *fixed point* with respect to the application of rules of inference is a set of propositions from which no further propositions can be generated by the application of rules of inference.

Given that a set of premises may have several extensions, we may wonder what theorems are generated by a set of premises. There are two different ways to define what a theorem is:

1. Theorems are the propositions that belong to every extension. This is called the *skeptical* approach, and is typically used in formalisms where many extensions are expected, without any intuitive criteria to select among them.

2. Theorems are the propositions that belong to least one extension. This is called the *liberal* approach, and is used in formalisms where there are few extensions or when there is a criterion to select one extension as preferable.

4.2 Default Logic

Default Logic was introduced in [Reiter 80] and was revised in [Reiter and Criscuolo 81]. A semantics for Default Logic was provided in [Etherington 87, 88].

4.2.1 Deductive system

Default Logic uses the language of classical logic ($\mathcal{L}_1$), and, besides the rules of inference of classical logic, it uses *rules of inference* of the form:

$$\frac{\alpha(\vec{x}) : \beta_1(\vec{x}), ..., \beta_m(\vec{x})}{\gamma(\vec{x})}$$

in which $\alpha(\vec{x})$, $\beta_1(\vec{x}), ..., \beta_m(\vec{x})$, and $\gamma(\vec{x})$ are wffs whose free variables belong to the vector $\vec{x} = (x_1, ..., x_n)$. Such a rule of inference, called *default rule*, is interpreted in the following way: From $\alpha(\vec{x_0})$,[2] if it is consistent to assume $\beta_1(\vec{x_0})$,...,$\beta_m(\vec{x_0})$, then we can infer $\gamma(\vec{x_0})$. Default rules can be looked at as suggestions with respect to what we should believe in addition to what is dictated by classical logic.

The wff $\alpha(\vec{x})$ is called the *precondition* of the rule; wffs $\beta_1(\vec{x}), ..., \beta_m(\vec{x})$ are called the *justifications* of the rule; and $\gamma(\vec{x})$ is called the *consequent* of the rule. If the wffs α, β_1, ..., β_m, and γ have no free variables, then the default rule is said to be *closed*. Free variables in a default rule are considered to be universally quantified.

We should note the different nature that default rules have as compared with classical inference rules. The applicability conditions of the default rule

$$\frac{\alpha(\vec{x}) : \beta_1(\vec{x}), ..., \beta_m(\vec{x})}{\gamma(\vec{x})}$$

require that $\alpha(\vec{x_0})$ holds (which is similar to what happens in classical logic), but they also require that $\neg\ \beta_1(\vec{x_0})$, ...,$\beta_m(\vec{x_0})$ not be derivable from the premises, using *all* rules of inference, which include the rule under consideration. In other words, in order to find whether a given default rule is applicable, it is necessary to take into account the results produced by the application of that rule.

As an example of a default rule, let us consider the statement that in general adults who are not students are employed. This statement is expressed by the following default rule:

$$\frac{Adult(P) : \neg Student(P)}{Employed(P)}$$

A particular case of default rules, called *normal default rules*, are rules of following form:

$$\frac{\alpha(\vec{x}) : \beta(\vec{x})}{\beta(\vec{x})}$$

For example, the statement "typically birds fly" can be represented by the following normal default rule:

$$\frac{Bird(x) : Flies(x)}{Flies(x)}$$

which can be read as: "if x is a bird and if it is consistent to assume that x flies, then we can conclude that x flies".

Exceptions to default rules can be expressed by wffs, for example,

$$\forall x[(Penguin(x) \rightarrow \neg Flies(x)].$$

[2] $\vec{x_0}$ is an instance of $\vec{x}$.

A *default theory* is a pair (Ψ, Δ), composed of a set of default rules, Ψ, and by a set of closed wffs, $\Delta \subset \mathcal{L}_1$, that represent the basic knowledge and are treated as premises. Both Ψ and Δ can be infinite sets (but have to be denumerable).

A default theory that only contains closed default rules is called *closed*, and a default theory that only contains normal default rules is called *normal*.

Given a default theory (Ψ, Δ), we want to compute the sets of wffs derivable from Δ using the rules of inference of classical logic and the default rules in Ψ. These sets correspond, in classical logic, to the theorems derivable from Δ . However, in non-monotonic logics, there may by more than one of these sets or even none. Each one of these sets is called an *extension* of the default theory (Ψ, Δ). Each extension may be interpreted as a set of beliefs generated from Δ, using the default rules in Ψ.

There are three properties that an extension of the default theory (Ψ, Δ) should have:

1. It should contain Δ.
2. It should be closed with respect to derivability in the classical sense (using only the rules of inference of classical logic). This aspect guarantees that an extension is as complete as possible with respect to the classical notion of consequence.
3. It should be closed with respect to the applications of default rules in Ψ. In other words, all default rules that are consistent with the extension should be applied.

We should note that these conditions say nothing about what should not exist in an extension. For example, the set of all wffs in $\mathcal{L}_1$ satisfies the three conditions above. In order to avoid the introduction, in an extension, of propositions without a proper justification, we should require that an extension be a minimal set. With this additional constraint, we still allow the existence of non-justified wffs in an extension, as illustrated by the following example (from [Lukaszewicz 90, pp. 169-170]). Let us consider the default theory:

$$(\{\frac{P : Q}{Q}\}, \{P\}).$$

There are two minimal sets that satisfy the three conditions above:

$$\Phi_1 = Th(\{P, Q\})$$

and

$$\Phi_2 = Th(\{P, \neg Q\}).$$

Clearly, only Φ_1 should be considered as an extension of the default theory. The source of the difficulty is the fact that the criteria for the application of a default rule take into account both the wffs that have already been derived and the wffs that have not been derived (but can be derived). This fact enables blocking the application of default rule by the introduction of the negation of its justification. If this negated wff is not justified, it should not belong to the extension.

In order to come up with a proper definition of extension, let us suppose that both Ω and $\Gamma(\Omega)$ represent the same extension of the default theory (Ψ, Δ), and let us try to define $\Gamma(\Omega)$ in terms of Ω. In other words, let us suppose that already know the extension Ω, and, based on this fact, we re-construct that extension, giving rise to $\Gamma(\Omega)$. Let us consider the following conditions:

1. $\Delta \subset \Gamma(\Omega)$.
2. $\Gamma(\Omega)$ is closed under derivability, in other words, $Th(\Gamma(\Omega)) = \Gamma(\Omega)$.
3. If $\frac{\alpha:\beta}{\gamma} \in \Psi$, $\alpha \in \Gamma(\Omega)$, and $\neg\beta \notin \Omega$, then $\gamma \in \Gamma(\Omega)$.

Although these conditions may resemble those stated before, there is a fundamental difference between them. Given Ω and $\Gamma(\Omega)$, we are able to formally distinguish what should be in the extension and what should not. This enables us to formally define an extension.

Let (Ψ, Δ) be a default theory and let Ω be a set of wffs ($\Omega \subset \mathcal{L}_1$). Let $\Gamma(\Omega)$ be the smallest set of wffs of $\mathcal{L}_1$ that satisfies the three conditions above. The set Ω is an *extension* of the default theory (Ψ, Δ) if and only if $\Gamma(\Omega) = \Omega$, in other words, if Ω is a fixed point of the operator Γ.

Let us re-consider the default theory

$$(\{\frac{P:Q}{Q}\}, \{P\}).$$

and the sets:

$$\Phi_1 = Th(\{P, Q\})$$

and

$$\Phi_2 = Th(\{P, \neg Q\}).$$

Since

$$\Gamma(\Phi_1) = Th(\{P, \neg Q\}) = \Phi_1$$

and

$$\Gamma(\Phi_2) = Th(P) \neq \Phi_2$$

only Φ_1 is an extension.

As a second example, the default theory[3]

$$(\{\frac{:P}{\neg Q}, \frac{:Q}{\neg P}\}, \{\})$$

has two extensions: $\Phi_1 = Th(\{\neg P\})$ and $\Phi_2 = Th(\{\neg Q\})$.

Finally, the following default theory (from [Reiter 80, p. 91]) has no extensions:

$$(\{\frac{:A}{\neg A}\}, \{\}).$$

[3] The default rules of this theory have empty antecedents. The first default rule, for example, is read as "if it is consistent to assume P, then we can conclude $\neg Q$".

Default theories are non-monotonic in the following sense: if $T = (\Psi, \Delta)$ is a default theory with extension Ω, Ψ' is a set of default rules, and Δ' is a set of wffs ($\Delta' \subset \mathcal{L}_1$), then $T' = (\Psi \cup \Psi', \Delta \cup \Delta')$ may have no extension Ω' such that $\Omega \subset \Omega'$.

However, normal default theories have three important properties:

1. *Semi-monotonicity.* If the set of default rules of a normal default theory increases, then, for every extension of the original theory, there is an extension of the new theory that contains it.

2. *Guarantee of extensions.* Every closed normal default theory (a theory only with closed normal rules) has an extension [Reiter 80, p. 95, Theorem 3.1].

3. Existence of a *mechanical decision procedure for formulas in an extension.* Given the closed normal default theory $T = (\Psi, \Delta)$ and a wff $\beta \in \mathcal{L}_1$, it is possible to find out whether there is an extension Ω of T such that $\beta \in \Omega$,

It should be noticed that the operator Γ used to define the extensions of a theory is defined in non-constructive way; that is, it is clearly insufficient to compute the extensions of default theory. Recall that this operator relies on the existence of a "guess" of an extension.

Although normal default rules yield theories that are easy to formalize, they may have certain undesirable conclusions, as can be shown in the following example (from [Reiter and Criscuolo 81]). Let us consider the normal default theory $T = (\Psi, \Delta)$ in which Ψ has two default rules:

$$\psi_1 = \frac{Student(x) : Adult(x)}{Adult(x)}.$$

and

$$\psi_2 = \frac{Adult(x) : Employed(x)}{Employed(x)}.$$

and Δ has just one wff, $\Delta = \{Student(John)\}$. Default rule ψ_1 states that "Typically, students are adults", and default rule ψ_2 states that "Typically, students are employed". These two default rules, taken together, enable the inference that "Typically, students are employed", which, in general, is false.

In order to avoid transitivity in the application of default rules, we may increase Ψ with the default rule:

$$\psi_3 = \frac{Student(x) : \neg Employed(x)}{\neg Employed(x)}.$$

The default theory $T' = (\{\psi_1, \psi_2, \psi_3\}, \{Student(John)\})$ has two extensions:

$$\Omega_1 = Th(\{Student(John), Adult(John), \neg Employed(John)\})$$

and

$$\Omega_2 = Th(\{Student(John), Adult(John), Employed(John)\}).$$

Although only extension Ω_1 is reasonable, nothing in the logic makes it preferable to Ω_2. In order to avoid extension Ω_2, we can modify default rule ψ_2 in the following way:

$$\psi_2' = \frac{Adult(x) : Employed(x) \wedge \neg Student(x)}{Employed(x)}.$$

Default theory $T'' = (\psi_1, \psi_2'), \{Student(John)\})$ only has the required extension (Ω_1).

Default rule ψ_2', which has the form:

$$\frac{\alpha(x) : \beta(x) \wedge \neg\gamma(x)}{\beta(x)}.$$

is called a *semi-normal* default rule. Semi-normal default theories do not have a guarantee of extension, nor do they have the semi-monotonic property.

4.2.2 Semantic system

A semantics for Default Logic was introduced in [Etherington 87, 88]. This semantics deals with sets of models, in the classical sense.

The basic idea underlying the computation of the models of the extensions of the default theory (Ψ, Δ) is to start with the set of all the models of Δ and to use the default rules in Ψ to generate smaller sets of models.[4] The smallest sets of models correspond, with some additional conditions, to the models of the extensions.

The main idea underlying the semantics of Default Logic is to introduce a partial order among the sets of models of a default theory. Let $\mathcal{M}$ be a set of models and let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two subsets of that set ($\mathcal{M}_1$, $\mathcal{M}_2 \in 2^{\mathcal{M}}$). Let

$$\psi = \frac{\alpha : \beta_1, ..., \beta_n}{\gamma}$$

be a default rule. This default rule introduces a partial order $\geq_\psi$ on $2'\mathcal{M}$. We say that the default rule ψ *prefers* the set of models $\mathcal{M}_1$ to the set of models $\mathcal{M}_2$, written as $\mathcal{M}_1 \geq_\psi \mathcal{M}_2$, if and only if:

$$\begin{gathered} \forall \mathcal{M} \in \mathcal{M}_2, \models \alpha \wedge \\ \exists \mathcal{N}_1, ..., \mathcal{N}_n \in \mathcal{M}_2 : \mathcal{N}_i \models \beta_1 \wedge \\ \mathcal{M}_1 = \mathcal{M}_2 - \{\mathcal{M} : \mathcal{M} \models \neg\gamma\} \end{gathered}$$

Intuitively, the partial order $\geq_\psi$ captures the preference by ψ of more specialized descriptions of the word, in which the consequent of the rule is true, over other descriptions in which the preconditions of the rule are true, its justifications are consistent, but that do not satisfy the consequent.

[4] We should note that the smaller a set of models is, the larger the number of satisfied wffs will be. In particular, the empty set of models satisfies all the wffs.

The idea underlying the models preferred by a default rule can be extended to a set of default rules. Let Ψ be a set of default rules, and let $\mathcal{M}$ be a set of models. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two subsets of the set of models ($\mathcal{M}_1, \mathcal{M}_2 \in 2^{\mathcal{M}}$). The partial order, $\geq_\Psi$ introduced by Ψ on the set $2^{\mathcal{M}}$ is defined as the union of the partial orders introduced by the default rules in Ψ. We say that the set of default rules Ψ prefers the set of models $\mathcal{M}_1$, to the set of models $\mathcal{M}_2$, written as $\mathcal{M}_1 \geq_\Psi \mathcal{M}_2$, if and only if there exists a default rule in Ψ that prefers the set of models $\mathcal{M}_1$ to the set of models $\mathcal{M}_2$, *or* there is a set of models $\mathcal{M}'$ such that the set of default rules Ψ prefers the set of models $\mathcal{M}'$, to the set of models $\mathcal{M}_2$ and the set of default rules Ψ prefers the set of models $\mathcal{M}_1$, to the set of models $\mathcal{M}'$:

$$(\exists \psi \in \Psi[\mathcal{M}_1 \geq_\psi \mathcal{M}_2]) \vee (\exists \mathcal{M}' \in 2^{\mathcal{M}}[\mathcal{M}_1 \geq_\Psi \mathcal{M}' \geq_\Psi \mathcal{M}_2]).$$

For normal default theories (Ψ, Δ), it is enough to consider the maximal models in respect to $\geq_\Psi$ containing elements of $2^{Mod(\Delta)}$ (in which $Mod(\Delta)$, is the set models of Δ, that is, $Mod(\Delta) = \{M : M \models \Delta\}$). Each one of these maximal models corresponds to the model of an extension of the default theory (Ψ, Δ).

Non-normal default theories, since they do not satisfy the semi-monotonicity property, require a more complex approach. This approach is based on the notion of *stability*, which guarantees that the maximal models satisfy all the preference conditions of the default rules used to generate them.

Let (Ψ, Δ) be a default theory, and let $\mathcal{M} \in 2^{Mod(\Delta)}$. We say that $\mathcal{M}$ is *stable* in (Ψ, Δ) if and only if there is $\Psi' \subseteq \Psi$ such that $\mathcal{M} \geq_{\Psi'} Mod(\Delta)$ and for each default rule

$$\frac{\alpha : \beta_1 ... \beta_n}{\gamma} \in \Psi' \exists N_1 ... N_n \in \mathcal{M} : N_i \models \beta_i.$$

In other words, a set of models is stable in the default theory (Ψ, Δ) if it is a specialization of the set of models of Δ and does not invalidate the justifications of any default rule used in the specialization.

Etherington proved the following results regarding this semantics [Etherington 88, pp. 174-176]:

THEOREM (*soundness*): If Ω is an extension of the default theory (Ψ, Δ), then $\{M : M \models \Omega\}$ is stable and maximal for (Ψ, Δ).

This theorem guarantees that the extensions obtained in the deductive system correspond to extensions in the semantic system.

THEOREM (*completeness*): If $\mathcal{M}$ is a stable and maximal set of models of (Ψ, Δ), then $\mathcal{M}$ is the set of models for some extension of (Ψ, Δ). In other words, the set $\{\alpha : \forall M \in \mathcal{M}, M \models \alpha\}$ is an extension of (Ψ, Δ).

This theorem guarantees that every extension obtained in the semantic system have a corresponding extension in the deductive system.

As a first example, let us consider the normal default theory $T_1 = (\Psi, \Delta)$, where

$Model$	A	B	C	D	E	F
M_1	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
M_2	$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$
M_3	$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$
M_4	$\bot$	$\bot$	$\top$	$\top$	$\top$	$\top$
M_5	$\top$	$\top$	$\bot$	$\top$	$\top$	$\top$
M_6	$\bot$	$\top$	$\bot$	$\top$	$\top$	$\top$
M_7	$\top$	$\bot$	$\bot$	$\top$	$\top$	$\top$
M_8	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\top$
M_9	$\top$	$\top$	$\bot$	$\bot$	$\top$	$\top$
M_{10}	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\top$
M_{11}	$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\top$
M_{12}	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$
M_{13}	$\bot$	$\top$	$\top$	$\top$	$\bot$	$\top$
M_{14}	$\top$	$\bot$	$\top$	$\top$	$\bot$	$\top$
M_{15}	$\bot$	$\bot$	$\top$	$\top$	$\bot$	$\top$
M_{16}	$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\top$
M_{17}	$\top$	$\bot$	$\bot$	$\top$	$\bot$	$\top$
M_{18}	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	$\top$
M_{19}	$\top$	$\top$	$\bot$	$\bot$	$\top$	$\bot$
M_{20}	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\bot$
M_{21}	$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$
M_{22}	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$

Table 4.1: Models of $\{C \to D, (A \wedge B) \to E, E \vee D, D \to F\}$.

Ψ has the following four default rules:

$$\psi_1 = \frac{E \vee F : A \wedge F}{A \wedge F}$$

$$\psi_2 = \frac{A : B}{B}$$

$$\psi_3 = \frac{A \wedge E : C}{C}$$

$$\psi_4 = \frac{: \neg E}{\neg E}$$

and Δ contains the following wffs:

$$\Delta = \{C \to D, (A \wedge B) \to E, E \vee D, D \to F\}.$$

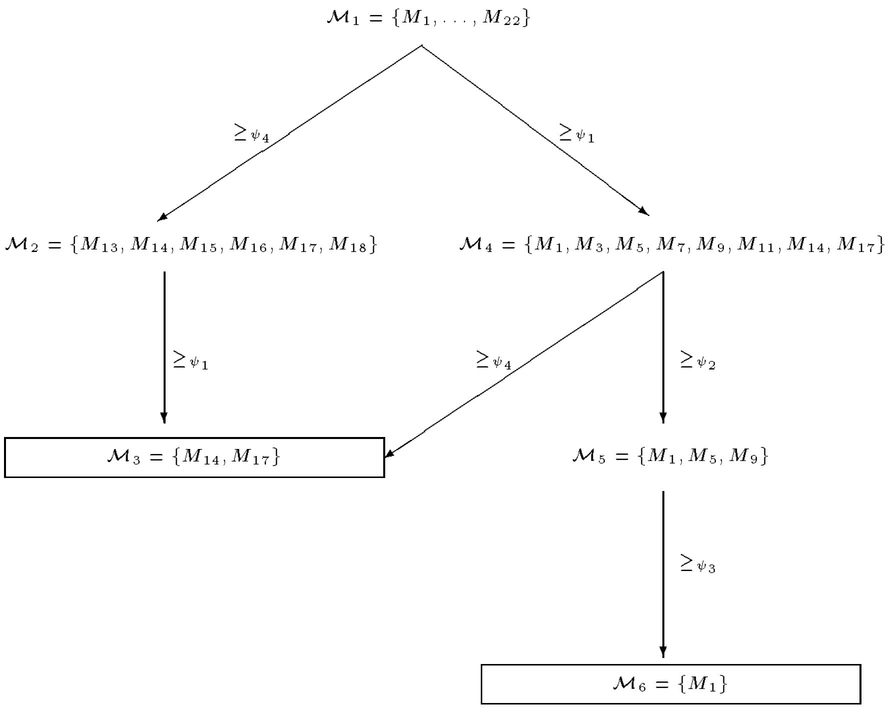

Figure 4.1: Partial order between the models of the theory T_1 (using an explicit representation of models).

Table 4.1 shows all models of Δ, that is, all combinations of truth values that make all the wffs in Δ true.[5] If we apply the definition of preferential models introduced by the default rules, we obtain the partial order presented in Figure 4.1, in which the maximal models are represented inside a rectangle. From the completeness theorem, we can conclude that T_1 has two extensions, which are defined by the sets of models $\mathcal{M}_3 = \{M_{14}, M_{17}\}$ and $\mathcal{M}_6 = \{M_1\}$.

Figure 4.1 makes use of an explicit representation of the models in each of the preferential sets of models. An alternative to represent the sets of models, uses an implicit representation and is shown in Figure 4.2. In this figure:

1. The set of models $\mathcal{M}_1$ contains all the models that satisfy the wffs in the set $\Delta = \{C \rightarrow D, (A \wedge B) \rightarrow E, E \vee D, D \rightarrow F\}$.

2. The set of models $\mathcal{M}_2$ contains all the models that satisfy $\Delta = \{C \rightarrow D, (A \wedge B) \rightarrow E, E \vee D, D \rightarrow F\}$ and the consequent of the default rule ψ_4,

[5] In this table, $\top$ represents the truth value "true", and $\perp$ represents the truth value "false".

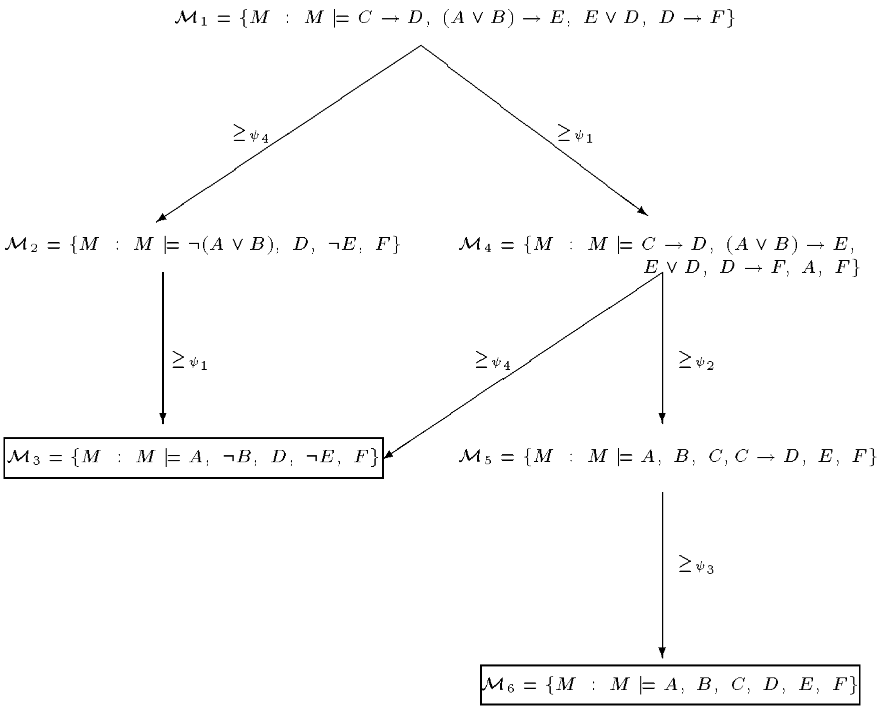

Figure 4.2: Partial order between the models of the theory T_1 (using an implicit representation of models).

that is, the set of models that satisfies $\{C \rightarrow D, (A \wedge B) \rightarrow E, E \vee D, D \rightarrow F, \neg E\}$. We should note that this set can be simplified in the following way: Since both $E \vee D$ and $\neg E$ have to be true, we may conclude that D has to be true; since both D and $D \rightarrow F$ have to be true, we may conclude that F has to be true; since $(A \wedge B) \rightarrow E$, and $\neg E$ have to be true, we may conclude that $\neg(A \wedge B)$ has to be true. Thus, $\mathcal{M}_2$ represents the set of models $\{\neg(A \wedge B), D, \neg E, F\}$.

3. The set of models $\mathcal{M}_3$ contains all the models that satisfy $\{\neg(A \wedge B)$, D, $\neg E$, $F\}$ and the consequent of default rule ψ_1, that is, $\{\neg(A \wedge B)$, D, $\neg E$, F, $E \vee F$, $A \wedge F\}$. In a similar way, we may conclude that $\mathcal{M}_3$ represents the set of models $\{A, \neg B, D, \neg E, F\}$.

4. The same line of reasoning is applied to the sets of models $\mathcal{M}_4$, $\mathcal{M}_5$, and $\mathcal{M}_6$.

Let us now consider the non-normal default theory

$$T_2 = (\{\psi_1, \psi_2\}, \{WeekDay(Today)\}),$$

where ψ_1 and ψ_2 are the following default rules:

$$\psi_1 = \frac{Weekday(Today) : \neg HasExcuse(Peter, Today)}{Works(Peter, Today)}$$

$$\psi_2 = \frac{Weekday(Today) : \neg Works(Peter, Today)}{Sick(Peter, Today)}$$

In other words, in an weekday, if it is consistent to assume that Peter has no excuse, then Peter works (ψ_1); in a weekday, if it is consistent to assume that Peter does not work, then Peter is sick (ψ_2). Today is a weekday.

In order to compute what can be concluded from this theory, we will determine the models if its extensions. Figure 4.3 shows the partial order introduced by the default rules of the theory T_2. In fact,

$$\mathcal{M}_2 \geq_\Psi \mathcal{M}_1$$

$$\mathcal{M}_4 \geq_\Psi \mathcal{M}_3 \geq_\Psi \mathcal{M}_1.$$

In this partial order, there are two maximal sets of models $\mathcal{M}_2$ and $\mathcal{M}_4$. The set of models $\mathcal{M}_2$ results from the application of default rule ψ_1 to the premisses; it states that Peter works, because it is consistent to assume that he has no excuse (nothing in the premisses lead to the conclusion of this fact). The set of models $\mathcal{M}_4$ results from the application of default rule ψ_2, followed by the application of default rule ψ_1; it states that Peter works today and Peter is sick today.

Out of these two sets of models, only $\mathcal{M}_2$ is stable. In order for $\mathcal{M}_4$ to be stable, there should be $\Psi' \subseteq \Psi$ such that $\mathcal{M}_4 \geq_{\Psi'} \mathcal{M}_1$, and for each default rule[6]

$$\frac{\alpha : \beta}{\gamma} \in \Psi'[\exists M \in \mathcal{M}_4 : M \models \beta].$$

[6]Note that this is the application of the stability conditions to this particular theory.

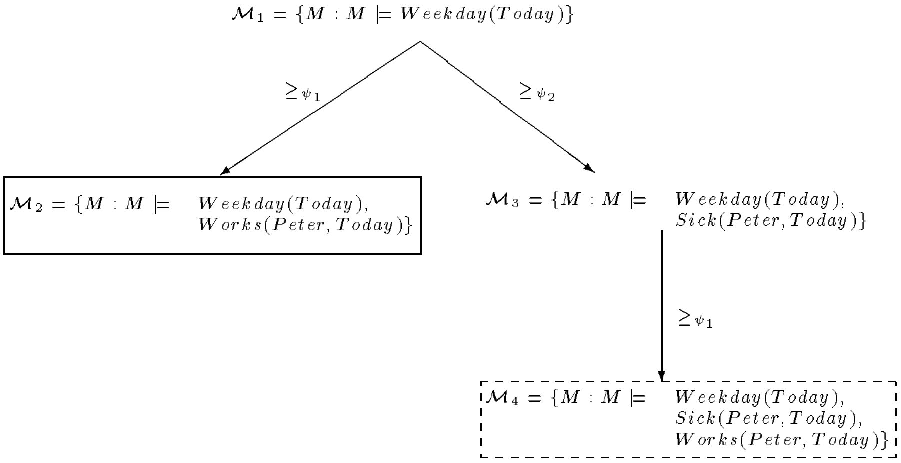

Figure 4.3: Partial order between the models of the default theory T_2.

This condition is not satisfied, because $\forall M \in \mathcal{M}_4 : M \models \neg Works(Peter, Today)$. Which goes against the justification of default rule ψ_2. In fact, the application of default rule ψ_1 goes against the justification of default rule ψ_2 ($\neg Works(Peter, Today)$), and for this reason model $\mathcal{M}_4$ is not stable. In other words, T_2 has only extension, defined by the set of models $\mathcal{M}_2$.

This situation never happens when only normal rules are used because the justification of a normal rule is equal to its conclusion.

4.3 Other approaches

In this section, we mention two alternative approaches to the development of non-monotonic logics, Circumscription and Auto-epistemic Logic.

4.3.1 Circumscription

Circumscription was introduced in [McCarthy 80], was generalized in [McCarthy 84], and has been explored by many researchers. Circumscription is not a non-monotonic logic but rather an attempt to add to classical logic a way of "jumping to conclusions" and to infer certain properties about the objects that satisfy certain relations. The idea underlying Circumscription is to state that all the objects that satisfy a certain property are those for which it is possible to prove that property. For example, circumscribing the property of "being a block" corresponds to assume that all objects about which we cannot prove that are blocks, are not blocks.

Let α be a first-order wff containing the n-ary predicate letter P. Let $\alpha(\Phi)$ be the result obtained by replacing all occurrences of P in α by the predicate Φ. The circumscription of P in α, with the predicate Φ, is the schemata:

$$\alpha(\Phi) \wedge \forall \vec{x}[\Phi(\vec{x}) \rightarrow P(\vec{x})) \rightarrow \forall \vec{x}[P(\vec{x}) \rightarrow \Phi(\vec{x})]),$$

where $\vec{x}$ is a vector of n elements, $\vec{x} = (x_1, \ldots, x_n)$. Notice that, if the antecedent of the implication is false, nothing can be concluded with Circumscription. The interesting case happens when $\alpha(\Phi)$ is true (that is, when Phi shares with P the property of satisfying α), and having the property Φ implies having the property P. In this case, Circumscription allows to conclude that Φ and P are equivalent. All instances of this schemata, together with α, may be used to derive new information. The parameter Φ may be replaced by *any* expression containing a n-ary predicate.

The above schemata states that the only entities ($\vec{x}$) that satisfy P are those that *have to* satisfy it, assuming the proposition α. Since the schemata corresponds to an implication, if we assume both antecedents we may infer the consequent. The first antecedent, $\alpha(\Phi)$ corresponds to the assumption that Φ satisfies the conditions required by P, and the second antecedent $\forall \vec{x}[\Phi(\vec{x}) \rightarrow P(\vec{x})]$ corresponds to the assumption that the entities that satisfy Φ are a subset of the entities that satisfy P. The conclusion states that the converse is true, which makes Φ and P extensionally equivalent.

Let us suppose that α was the following proposition (from [McCarthy 80, p. 32]):

$$Block(A) \wedge Block(B) \wedge Block(C),$$

that states that A, B, and $C$are blocks. This proposition contains the unary predicate "*Block*". The circumscription of "*Block*" in α originates the schemata:

$$((\Phi(A) \wedge \Phi(B) \wedge \Phi(C)) \wedge \forall x[\Phi(x) \rightarrow Block(x)]) \rightarrow \forall x[Block(x) \rightarrow \Phi(x)].$$

In order to proceed, we have to chose Φ in such a way that guarantees that Φ corresponds to all blocks that exist. If we chose

$$\Phi(x) = (x = A \vee x = B \vee x = C)$$

and we replace $Phi(x)$ in the schemata that defines circumscription, and that into account that α, the antecedent of the implication, is satisfied, we can infer that

$$\forall x[Block(x) \rightarrow (x = A \vee x = B \vee x = C)]$$

which states that the only blocks that exist are A, B, and C.

The difficulty in the use of Circumscription is the choice of the predicate Φ. This choice has to be done in such a way to guarantee that it defines (circumscribes) the entities in which we are interested. For complex situations, the choice of this predicate is very difficult, which prevents the use of Circumscription in systems for automatic reasoning.

4.3.2 Auto-epistemic Logic

Auto-epistemic Logic was proposed in [Moore 83, 85, 88] and uses the modal operator B, that is read "believes". The word "auto-epistemic" stems from the fact that this logic corresponds to the introspection of knowledge by the agent that performs the reasoning, and thus it is adequate to model agents that reflect on their own beliefs. In Auto-epistemic Logic, it is possible to represent propositions such as "If I don't believe that P then Q is true" and "If P is true, then I believe that P".

The language of Auto-epistemic logic extends the language of first-order logic with the introduction of the modal operator B. The intuitive interpretation of $B\alpha$ is "I believe in α". For example, the wff

$$\forall x[(Bird(x) \wedge BFlies(x)) \rightarrow Flies(x)],$$

may be read as "If x is a bird and I believe that it flies, then it flies", in other words, I will believe that all birds fly, except for those that I explicitly know not to fly. Like other non-monotonic logics formalisms, in Auto-epistemic logic, it is not possible to define, in a constructive way, the set of wffs that may be deduced for a set of premises. Since the goal of Auto-epistemic logic is to model the beliefs of an agent that reflects upon its own beliefs, we will be interested in sets of auto-epistemic formulas that are interpreted as the beliefs of such agents.

Moore defines, in syntactic terms, the conditions that should be satisfied by the set of beliefs of a ideal rational agent. By *rational*, it is meant an agent whose reasoning process only originates valid arguments; by *ideal*, it is meant an agent in all the consequences of its beliefs. Intuitively, an auto-epistemic theory generated from a set of premises, corresponds to a maximal and consistent set of propositions that my be deduced from that set of premises. This set should contain everything that can be derived from the premises, using classical logic, plus everything that can be deduced by reflection over the wffs that were deduced. An *auto-epistemic theory* is a set of wffs, Ω. An auto-epistemic theory is said to be *stable* if it satisfies the following conditions:

1. If $\{\alpha_1, \ldots, \alpha_n\} \in \Omega$ and $\{\alpha_1, \ldots, \alpha_n\} \vdash \beta$, then $\beta \in \Omega$.
2. If $\alpha \in \Omega$, then $B\alpha \in \Omega$.
3. If $\alpha \notin \Omega$, then $\neg B\alpha \in \Omega$.

The stability stems from the fact that if conditions 1, 2, and 3 are satisfied, no additional proposition may be inferred by a rational agent. These conditions were defined in Stalnaker 80] and are known as *Stalnaker conditions*.

Considering an auto-epistemic theory a the set of beliefs to be held by a rational agent, Stalnaker conditions state that a rational agent knows all the consequences of its beliefs (it is logically omniscient), it believes in its beliefs and does not believe in what does not belong to its beliefs.

An auto-epistemic theory, Ω that is stable and consistent, besides Stalnaker conditions, satisfies two additional conditions:

1. If $B\alpha \in \Omega$, then $\alpha \in \Omega$.
2. If $\neg B\alpha \in \Omega$, then $\alpha \notin \Omega$.

In fact, assuming that $B\alpha \in \Omega$ and $\alpha \notin \Omega$, we could conclude, using the third Stalnaker condition, that $\neg B\alpha \in \Omega$, which contradicts the fact that Ω is consistent; assuming that $\neg B\alpha \in \Omega$ and $\alpha \in \Omega$ we could conclude, using the second Stalnaker condition, that Ω was not consistent because it would contain both $\neg B\alpha$ and $B\alpha$.

When dealing with auto-epistemic theories, we will be interested in computing the theories generated from an set of premises Δ. When we consider the stability conditions we can realize that they will not be sufficient for this job, once nothing prevents Ω from having wffs that are not derived from Δ (neither in the classical sense nor by reflection on the beliefs of the agent). It is then necessary to add additional conditions that guarantee that the only wffs in an auto-epistemic theory are the premises and the wffs derivable from them. This is obtained through the definition of a theory grounded in a set of premises.

An auto-epistemic theory is *grounded* in a set of premises Δ if and only if all the wffs in Ω were derived (using classical logic) from the set

$$\Delta \cup \{B\alpha : \alpha \in \Omega\} \cup \{\neg B\alpha : \alpha \notin \Omega\}.$$

Defining the operator AE_Δ as:

$$AE_\Delta(\Omega) = Th(\Delta \cup \{B\alpha : \alpha \in \Omega\} \cup \{\neg B\alpha : \alpha \notin \Omega\}),$$

an auto-epistemic theory Ω is grounded on the set of premises Δ if and only if:

$$AE_\Delta(\Omega) = \Omega,$$

in other words, if Ω is a fixed point of the operator AE_Δ.

Consider, for example, the set of premises

$$\Delta = \{\forall x[\neg BFlies(x) \rightarrow \neg Flies(x)], Flies(Tweety)\}.$$

The first wff in this set states that "If I don't believe that something flies, then it does not fly"; the second wff states that "Tweety flies". A grounded auto-epistemic theory generated from this set contains, among others, the following wffs

$$\{\neg BFlies(Silvester), \neg Flies(Silvester)\}.$$

In fact, the first wff is generated from the non-derivability of $Flies(Silvester)$:

$$\Delta \nvdash Flies(Silvester).$$

According to the third Stalnaker condition $\neg BFlies(Silvester)$ belongs to the auto-epistemic theory. This enable to generate, using modus ponens,

$$\neg Flies(Silvester).$$

Starting now with the set

$$\Delta' = \Delta \cup \{Flies(Silvester)\}$$

we do not have anymore the wff $\neg BFlies(Silvester)$ in the auto-epistemic theory, since

$$\Delta \vdash Flies(Silvester)$$

consequently, the new theory will contain $BFlies(Silvester)$.

Moore defines *stable expansion* of a set of premises Δ as an auto-epistemic theory, Ω, that contains Δ, and is stable and grounded in Δ, that is, as the set of consequences of

$$\Delta \cup \{B\alpha : \alpha \in \Omega\} \cup \{\neg B\alpha : \alpha \notin \Omega\}.$$

From a set of premises Δ, the possible sets of beliefs that a ideal rational agent may hold, from Δ are given by the stable expansions of Δ. Note that there may be more than one stable expansion of s set Δ. As an example, from [Moore 88, p.110], consider the set of premises

$$\Delta = \{\neg BP \rightarrow Q, \neg BQ \rightarrow P\}.$$

Any auto-epistemic theory that contains these premises, if P does not belong to the theory, then Q belongs to the theory, and the other way around. This means that a stable expansion contains either P or Q. but not both.

As a final remark, we should note that an agent following Auto-epistemic Logic is omniscient about what it knows and about what it does not know (if $\alpha \in \Omega$, then $B\alpha \in \Omega$; if $\alpha \notin \Omega$, then $\neg B\alpha \in \Omega$).

There are two alternatives to define the semantics of Auto-epistemic Logic, both proposed by Moore. The first semantics proposed [Moore 83, 85], is based on the notion of auto-epistemic model; the second one [Moore 84], is based on the notion of possible words from modal logic.

4.4 Objections to non-monotonic logic

Since non-monotonic logic correspond to a radical change to the behavior of a logical system, we should expect that objections should be raised against them. In this section, we overview some of these. A more complete discussion of these objections can be found in [Ginsberg 87, p. 11] and [Etherington 88, pp. 63-73].

1. *Objections to the designation.* According to [Israel 80], logic, by definition, is monotonic, thus the designation "non-monotonic logic" is contradictory.

2. *Objections to the use of logic to model non-monotonic reasoning.* This objection suggest the use of probabilistic and fuzzy notions, instead of logical notions, to model statements such as "Typically birds fly" [Cheeseman 85], [Zadeh 85]. In these approaches, propositions are associated with numerical values (in general, real numbers in the interval [0,1]) that define the degree of belief of those statements.

To answer this objection, we should keep in mind that it is very difficult, even impossible, to assign numerical values to propositions. Furthermore, some conclusions obtained by non-monotonic logic formalisms may have a degree of plausibility near to zero but nevertheless are plausible and interesting [Reiter 88, pp. 472-473].

3. *Objection regarding the lack of decidability of logic.* Remember that a logic is semi-decidable if there is a mechanical procedure that, given any propositions, $\alpha_1, ..., \alpha_n$, and β, is guaranteed to terminate in a finite amount of time $\{\alpha_1, ..., \alpha_n\} \vdash \beta$ and may never terminate if $\{\alpha_1, ..., \alpha_n\} \vdash not \beta$. First-order logic is semi-decidable, which means that we are guaranteed to have an answer if a given formula is a theorem. The objection raised to non-monotonic logic (in [Israel 80], [Reiter 80] and [Perlis 87]) is the fact that they are not decidable: The application of a rule of inference in Default Logic, for example, requires the verification of the consistency of the conclusion; in other words, we have to prove that the negation of the conclusion is not derivable from the premises.

   This aspect is, in fact, a problem that has two practical consequences: On one hand, the implementations of non-monotonic reasoning systems tend to be very slow, since they require multiple consistency checks (in theory, these may never return an answer to certain questions); on the other hand, a system that uses non-monotonic reasoning may produce wrong answers once in a while (or execute forever, which is even worse).

4. *Objection to the multiple sets of theorems.* The fact that the set of theorems of a non-monotonic logic is not unique (this corresponds to the existence of multiple extensions in Default Logic) has led [Kramosil 75], [Sandewall 72], and [Reiter 78a] to object to these formalisms. In particular, [Reiter 78a] states that "such behavior is clearly unacceptable. At least we should require a non-monotonic theory to satisfy a Church-Rosser property: independently of the order of application of the rules of inference, the set of premises must be unique".

   This is an objection that no longer makes sense. In fact people have understood the need of multiple extensions: thinking of an extension as the set of beliefs that a rational agent may have, given the premises and the suggestions in the default rules, it is just natural that more than one of these sets may exist.

5. *Objections to qualification and expressive power.* The inability to distinguish in non-monotonic logic between what is known and between what is suggested by the default rules has been pointed out by [Nutter 83] as a negative aspect of these logics.

   The answer to this objection is given, in part, by the use of Truth Maintenance Systems, described in the next chapter.

4.5 Summary

In this chapter, we addressed the study of Default Logic, a logic whose conclusions may be invalidated in the presence of additional information. These kind of logics, called non-monotonic logics, are important to model our reasoning pattern in face of incomplete information. We argued that in traditional logic it is not possible to perform this type of reasoning and presented arguments in favor of the need of non-monotonic logics.

Default Logic extends the rules of inference of classical logic with default rules. Rules that are triggered, not only by what has been derived, but also by what can be derived. This second aspect is responsible for the non-monotonic behavior of the logic. We show that in these logics the set of theorems cannot be computed in a constructive way and needs some for of "guessing" what will be in the final set of theorems.

We described other approaches to non-monotonic logics: Circumscription (that adds to classical logic a mechanism that enables jumping to conclusions), Auto-epistemic Logic (that uses a modal operator).

Finally, we listed some of the objections raised to non-monotonic logic.

4.6 Suggested readings

The topics presented in this chapter are discussed further in the following references:

[Besnard]
An introduction to Default Logic, describing, in parallel, other formalisms for non-monotonic reasoning.

[Ginsberg 87]
A collection of the main papers on non-monotonic reasoning.

[Lucaszewicz 90]
A book that presents a detailed study of the various approaches to non-monotonic reasoning, in particular, Default Logic, the non-monotonic logic of McDermott and Doyle, Auto-epistemic Logic, and Circumscription.

4.7 Exercises

1. Non-monotonic logics were proposed by AI researchers. Justify the need for these logics.

2. Discuss the objections to non-monotonic logics and the arguments against them.

3. Compute the models of the extensions of the closed normal default theory $T = (\Psi, \Delta)$, where Ψ has the following default rules:

$$\psi_1 = \frac{: B}{B}$$

$$\psi_2 = \frac{: A}{A}$$

and

$$\Delta = \{\neg A \vee \neg B\}.$$

4. Compute the models of the extensions of the closed normal default theory $T = (\Psi, \Delta)$, where Ψ has the following default rules:

$$\psi_1 = \frac{: B}{B}$$

$$\psi_2 = \frac{: A}{A}$$

$$\psi_3 = \frac{: C}{C}$$

and

$$\Delta = \{B \rightarrow (\neg A \wedge \neg C)\}.$$

5. Prove that the following default theory has no extension.

$$(\{\frac{: A}{\neg A}\}, \{\})$$

   (a) Using the syntactic definition of extension.

   (b) Using the semantic definition of extension.

6. Compute the models of the extensions of the default theory $T = (\Psi, \Delta)$, where Ψ has the following default rules:

$$\psi_1 = \frac{P : (R \wedge \neg S)}{R}$$

$$\psi_2 = \frac{Q : S}{S}$$

and

$$\Delta = \{P, Q\}.$$

7. The default rule

$$\frac{\alpha(\vec{x}) : \beta_1(\vec{x}), \ldots, \beta_m(\vec{x})}{\gamma(\vec{x})}$$

is interpreted in the following way, "from $\alpha(\vec{x_0})$, if it is consistent to suppose $\beta_1(\vec{x_0})$, ..., $\beta_m(\vec{x_0})$, then, we can derive $\gamma(\vec{x_0})$". Explain the reason why we need the wffs $\beta_1(\vec{x_0})$, ..., $\beta_m(\vec{x_0})$, as justifications for this rule, rather than their conjunction $\beta_1(\vec{x_0}) \wedge \ldots \wedge \beta_m(\vec{x_0})$. Suggestion: consider the default rule

$$\frac{P(x) : Q(x) \wedge \neg P(x)}{R(x)}.$$

8. Compute the models of the extensions of the default theory, using an implicit representation of models:

$$(\{\psi_1, \psi_2, \psi_3\}, \{\})$$

where

$$\psi_1 = \frac{: P \wedge \neg Q}{\neg Q}$$
$$\psi_2 = \frac{: Q \wedge \neg R}{\neg R}$$
$$\psi_3 = \frac{: R \wedge \neg P}{\neg P}.$$

9. Compute the models of the extensions of the default theory, using an implicit representation of models:

$$(\{\psi_1, \psi_2\}, \{P, Q\})$$

where

$$\psi_1 = \frac{P : R \wedge \neg S}{R}$$
$$\psi_2 = \frac{Q : S}{S}.$$

10. Compute the models of the extensions of the default theory $(\{\psi_1, \psi_2\}, \{P\})$, where

$$\psi_1 = \frac{P : Q}{Q}$$
$$\psi_2 = \frac{P : Q \wedge \neg Q}{R},$$

using an implicit representation of models.

11. Compute the models of the extensions of the default theory, using an implicit representation of models:

$$(\{\psi_1, \psi_2, \psi_3\}, \{P\})$$

where

$$\psi_1 = \frac{P : Q}{Q}$$
$$\psi_2 = \frac{Q : R}{R}$$
$$\psi_3 = \frac{Q : \neg R}{\neg R}.$$

12. Compute the models of the extensions of the default theory, using an implicit representation of models:

$$(\{\psi_1, \psi_2\}, \{P \vee Q\})$$

where

$$\psi_1 = \frac{: \neg P}{\neg P}$$

$$\psi_2 = \frac{: \neg Q}{\neg Q}.$$

13. Using Default Logic, represent the following information:

    (a) Typically, typist are women;

    (b) Typically, woman have children.

    What can be conclude from the fact that Mary is a woman?

14. Consider the statements above, plus the information that "Typically, women over 50 have no more children". What can you infer from the fact that Mary is s typist 55-years old? Are all the inferences produced acceptable? How can the problem be corrected?

15. Describe two examples, from the real world, where default rules are used. represent these rules in Default Logic.

Chapter 5

Truth Maintenance Systems

1. Justification-based systems
2. Assumption-based systems
3. Other approaches
4. Applications
5. Summary
6. Recommended readings
7. Exercises

In this chapter we discuss aspects that are important to use in reasoning systems that deal with incomplete or inconsistent information or with domains that are subject to change. We discuss the two alternatives for the support of these reasoning systems and present some application areas.

'Always speak the truth – think before you speak – and write it down afterwards.'
Lewis Carroll, *Through the Looking Glass*

Most computer programs constructed by researchers in AI maintain a model of their environment, which is updated to reflect the perceived changes in the environment. This model is typically stored in a knowledge base which contains propositions about the environment (represented in some formal language, for example, first order logic, a semantic network, etc.), and the program draws inferences from the information in the knowledge base. The set of propositions in the knowledge base under consideration at a given instant of time is called the *beliefs* of the program, that is, the set of propositions in which the program believes. New beliefs may be added to the knowledge base either through observation (propositions that are added to the knowledge and are originated outside the system) or through inference using the beliefs in the knowledge base (Figure 5.1). New beliefs are always added to the knowledge base.

Programs that follow this model have to face the belief revision problem, i.e., the problem of changing their beliefs when some unexpected condition arises. This revision may have to be done because new observations lead to conflicts with the beliefs in the knowledge base; or because new beliefs originated during inference show that some of the beliefs in the knowledge base are not plausible or are mutually inconsistent.

The study of system with defeasible conclusions is somewhat recent in AI (e.g., [Ginsberg 87]) and has drawn a considerable interest. The research carried out in this area has followed two different, but complementary lines:

1. The study of logical systems in which some of the conclusions drawn can be invalidated in face of future evidence,*non-monotonic logics*;

2. The study of the aspects which are necessary to incorporate in a computer program in order to behave non-monotonically, these systems are called *truth maintenance systems, reason maintenance systems,* or *belief revision systems* (in this chapter we use the first designation).

Figure 5.1: Architecture of an AI system.

In non-monotonic logics, additional information (i.e., additional premises) entails the computation of a new theory, that is, the computation of the consequences of the new set of premises. The use of this approach in an AI program would correspond to the removal from the knowledge base of *all* derived beliefs and the start of the inference from scratch. This approach is not desirable because there are certainly many conclusions from the first set of premises that are still derivable from the second set. Instead, we would like to have a mechanism through which the *only* beliefs that are removed are those which are no longer derivable. Truth Maintenance Systems (TMS for short) are computer systems that allow for this behavior. Although there are different kinds of TMSes they all have the following aspects in common: they keep the consistency of a set of beliefs; they only keep beliefs as long as there are reasons for it; they explain the reasons for holding a belief; they identify contradictions; they identify culprits for contradictions; and they avoid the reappearance of known contradictions.

For a TMS, the word "belief" is taken to denote *justified belief*: a proposition is believed either because the TMS was told so — this corresponds to an observation (these beliefs are called hypotheses or premises), or because it depends on believed propositions.

An essential aspect associated with TMSes corresponds to recording dependencies between beliefs. If it would be impossible to distinguish beliefs that depend upon a premise and beliefs that do not then any belief may be questioned when the premise is removed; if it is not possible to identify the premises that underlie a contradiction then every premise in the knowledge base may be questioned when a contradiction occurs.

Depending on the process used to record dependencies between beliefs we can distinguish two main types of TMSes:

1. *Justification-based systems* in which each belief is associated with those beliefs that were immediately used in its derivation; and

2. *Assumption-based systems* in which each belief is associated with the premises that underlie its derivation.

A TMS extends the operations that can be performed upon the knowledge base. Typically, these systems do not have deductive capabilities (with the exception of the systems of [Martins and Shapiro 83, 88] and [McAllester 78]), they receive information from the program that works with the application domain (the problem solver) and they supply the problem solver with the information about which propositions should be believed.

The architecture of a system that uses a TMS is shown in Figure 5.2. This figure corresponds to a more detailed version of the architecture of Figure 5.1. Inference is performed by the problem solver, which receives from the TMS the information about which propositions are believed and supplies the TMS with the results of the inference, alongside with the reasons that justify the inference. Observations are also supplied to the TMS with appropriate justifications; the TMS is responsible for the revision of the beliefs in the knowledge base. Typically, a TMS explores alternatives, makes choices, explores the consequences of

Figure 5.2: Architecture of a TMS-based problem solver.

its choices, and compares results obtained when using different choices. If, during this process, a contradiction is detected, the TMS will revise the knowledge base, changing its beliefs in order to get rid of the contradiction.

5.1 Justification-based systems

Justification-based TMS (JTMS for short) historically corresponds to the first kind of TMS developed, as the result of the work of Jon Doyle [Doyle 79] the first researcher to separate the TMS from the problem solver.

In a JTMS each belief is associated with two sets of beliefs: (1) the set that should be believed for the belief to be held, and (2) the set that should not be believed for the belief to be held. Let us suppose that the problem solver believes that "by default birds fly", "penguins do not fly", and "Tweety is a bird". Using Default Logic, these propositions would be represented as:

$$\frac{Bird(x) : Flies(x)}{Flies(x)}$$

$$\forall(x)[Penguin(x) \rightarrow \neg Flies(x)]$$

$$Bird(Tweety)$$

With this information the problem solver could be lead to believe that "Tweety flies" and justify this belief with the belief that "Tweety is a bird" and the lack of belief in that "Tweety does not fly". Figure 5.3 shows the justification of the belief that "Tweety flies" using the notation introduced by [Goodwin 82], which will be discussed in detail later on. In this figure the word IN next to a proposition means that the proposition is believed and the word OUT next to a proposition means that the proposition is not believed. Besides the justifications for derived beliefs, as is the case for the justification to the belief that "Tweety flies", there are justifications for propositions that are the result

Figure 5.3: Justifications to believe that "Tweety flies".

Figure 5.4: Justifications not to believe that "Tweety flies".

of observations (or that are told by the problems solver as basic beliefs), as is the case of "Tweety is a bird", these are called *premises.* Premises are always believed, and are represented as shown in Figure 5.3. We should also notice that in Figure 5.3 there is no justification to believe that "Tweety does not fly" and. for this reason, it is not believed.

If we now add the premise that "Tweety is a penguin", which enables the derivation that "Tweety does not fly" the belief that "Tweety flies" is no longer held, and the justifications for this are as shown in Figure 5.4. We should note the lack of representation (in Figures 5.3 and 5.4) of the beliefs

$$\forall(x)[(Bird(x) \wedge \neg B \neg Flies(x)) \rightarrow Flies(x)$$

and

$$\forall(x)[Penguin(x) \rightarrow \neg Flies(x)].$$

In fact, a TMS only considers atomic propositions and their negation; all propositions using connectives or quantifiers ($\wedge$, $\vee$, $\rightarrow$, and $\forall$) are only used by the problem solver and cannot be disbelieved (with the exception of the systems of [McAllester 78] and [Martins and Shapiro 83, 88], in which the TMS and the problem solver are merged into a single component).

Since a TMS is only concerned about dependencies among beliefs, ignoring their contents, it is usual to replace those beliefs by identifiers (typically of the

Figure 5.5: Dependency network, supporting the disbelief in node n3.

form `n1`, `n2`, ...), which are associated with objects called *nodes.* There is a one-to-one function, T, that translates between beliefs and nodes. Using the translation

$$\begin{array}{r}
T(Bird(Tweety)) = \texttt{n1} \\
T(\ \ Flies(Tweety)) = \texttt{n2} \\
T(Flies(Tweety)) = \texttt{n3} \\
T(Penguin(Tweety)) = \texttt{n4}
\end{array}$$

the dependencies of Figure 5.4 are represented as shown in Figure 5.5.

In summary, a TMS works with nodes and justifications. Nodes stand for beliefs, and justifications support belief in nodes by relating them to other nodes. Associated with nodes there are labels that indicate what is the belief status of the associated node. It should be noticed that the existence of labels is due to computational efficiency, since their value can always be computed given the current nodes and justifications.

5.1.1 Dependency networks

One of the goals of the work in TMSes is the specification of the dependencies among beliefs and, based on such specification, the computation of which beliefs do hold and which beliefs do not hold. One way of doing this is to use a formal system called dependency networks, which is described in this section. Dependency networks were first proposed by [Doyle 83] and are nicely presented in [Reinfrank 89].

Let $\mathcal{L}$ be the language used to define the identifiers for nodes, a *dependency network* is a triple $[N, C, J]$ where $N \subset \mathcal{L}$, represents the nodes in the dependency network, $C \subset \mathcal{L}$, represents a distinguished set of nodes, the contradiction nodes, and $J \subset 2^{\mathcal{L}} \times 2^{\mathcal{L}} \times \mathcal{L}$, is a set of justifications of the form $<A_M \mid A_{NM} \Rightarrow c>$ in which $A_M \subset \mathcal{L}$, $A_{NM} \subset \mathcal{L}$, and $c \in \mathcal{L}$ (A_M is called

Figure 5.6: Representation of the premise p.

Figure 5.7: Representation of the justifications for c.

the set of *monotonic antecedents,* A_{NM} is called the set of *non-monotonic antecedents,* and c is the *conclusion*). The justification $<A_M \mid A_{NM} \Rightarrow c>$ supports the belief in c if all nodes in A_M are believed and no node in A_{NM} is believed. We say that$<A_M \mid A_{NM} \Rightarrow c>$ justifies node c. Justifications divide nodes into two categories:

1. *Premises*, which correspond to nodes for which there is a justification with no antecedents. Given the dependency network $[N, C, J]$ and a node $p \in N$, we say that p is a premise if $<\{\} \mid \{\} \Rightarrow p> \in J$. Premises are always believed and are represented as shown in Figure 5.6.

2. *Conclusions*, which correspond to nodes which have a justification with at least one non-empty antecedent. In Figure 5.7 we represent the justification $<\{a_1 ... a_n\} \mid \{a_{n+1} ... a_m\} \Rightarrow c>$. Node c is believed if nodes $a_1 \ldots a_n$ are all believed and none of the nodes $a_{n+1} \ldots a_m$ is believed.

5.1.2 Interaction between JTMS and the problem solver

JTMS communicates with the problem solver receiving information about new nodes and new justifications and supplying the problem solver with information about which nodes should be believed, taking into account the existing justifications (see Figure 5.2). The operations that can be performed by the JTMS can be divided into three categories:

1. *Inspection operations.*

   These operations correspond to an inspection of the dependency network and are triggered by questions from the problem solver. These questions are of three different types:

(a) Should I believe in node n? This question is answered by retrieving the label associated with node n.

(b) Why should I believe in node n? This question is answered by supplying the nodes that should be believed and the nodes that should not be believed so that node n is believed.

(c) What assumptions underlie node n? This question is answered by supplying all the premises that should be believed in order for node n to be believed.

2. *Modification operations.*

These operations modify the dependency network. They are originated by information supplied by the problem solver. This information can be of five different types:

(a) Add node n. This adds a new node, with identifier n and without any justification, to the dependency network. As a result, $[N, C, J]$ is transformed into $[N \cup \{n\}, C, J]$.

(b) Add premise p. This adds the premise with identifier p to the dependency network. If node p already exists it is justified as a premise. As a result, $[N, C, J]$ is transformed into $[N \cup \{p\}, C, J \cup \{< \{\} \mid \{\} \Rightarrow p >\}]$.

(c) Add contradiction p. This adds the node with identifier p to the dependency network, marking it as a contradiction node. As a result, $[N, C, J]$ is transformed into $[N \cup \{p\}, C \cup \{p\}, J]$.

(d) Justify node c, based in the belief in nodes a_1 ... a_n and in the lack of belief in nodes a_{n+1} ... a_m. The effect of this operation is to transform $[N, C, J]$ into $[N \cup \{a_1 ... a_n, a_{n+1} ... a_m, c\},\ C, J \cup \{<\{a_1 ... a_n\} \mid \{a_{n+1} ... a_m\} \Rightarrow c>\}]$.

(e) Remove premise p. The effect of this is the removal of the justification of p as a premise. As a result, $[N, C, J]$ is transformed into $[N, C, J - \{<\{\} \mid \{\} \Rightarrow p>\}]$.

3. *Updating operations.*

These operations are executed whenever a change is performed upon the dependency network. Their goal is to recompute the labels associated with the nodes in the network in such a way that they are consistent with the existing justifications.

5.1.3 Computation of labels for nodes

Given the dependency network $[N, C, J]$ we want to determine sets $B \subset N$ in which it is reasonable to believe, given the justifications J. The computation of reasonable beliefs is somewhat similar to the computation of an extension of a non-monotonic logic and cannot be formalized in a constructive way.

Figure 5.8: Odd and even loops.

The requirements imposed upon a reasonable set of beliefs are: (1) It is *well founded*, i.e., the beliefs are either premises or are supported by a justification whose monotonic antecedents are all believed (using well founded arguments) and whose non-monotonic antecedents are all disbelieved; (2) it is *closed*, i.e., the consequences of a justification whose monotonic antecedents are all believed and whose non-monotonic antecedents are all disbelieved is believed; (3) it is *not contradictory*, that is $B \cap C = \{\}$.

In the process of computation of reasonable beliefs there are two kinds of loops involving justifications that are of particular importance. The *odd loop* (Figure 5.8a) corresponds to a belief that cannot be decided whether or not it holds; and the *even loop* (Figure 5.8b) corresponds to two possible states of belief — if node n_1 is not believed, then node n_2 is believed and if node n_2 is not believed, then node n_1 is believed.

Algorithms for computing labels for nodes were developed by [Doyle 79], [Goodwin 82] (who developed a labeling algorithm that detects odd loops and always terminates, which is not the case with Doyle's algorithm), and [Petrie 86] (who modified Doyle's labeling algorithm by eliminating the necessity of a certain type of justifications, the CP-justifications).

Given the dependency network, $[N, C, J]$, labeling algorithms correspond to the computation of the values of the total function $L : N \Rightarrow \{\mathsf{IN}, \mathsf{OUT}\}$. It should be noticed that a labeling may not be unique as illustrated by the dependency network of Figure 5.9. In this figure we show, next to each node, its two possible labels. One labeling corresponds to all labels with white background, and the other labeling corresponds to all labels with shadowed background.

5.1.4 Removal of contradictions

One of the goals of a TMS is to maintain consistent sets of beliefs. Whenever the TMS discovers that its beliefs are inconsistent, i.e., that $B \cap \neq \{\}$, it should take action in order to restore consistency. This action can be of two kinds: (1) if one choice made during the labeling of an even loop can be changed so that the inconsistency is removed, then this choice is changed and the dependency network is unchanged; or (2) if the inconsistency does not depend on choices made during the labeling process, then the dependency network has to be changed in order to remove the inconsistency. This change can either correspond to the

Figure 5.9: Two possible labeling for nodes.

Figure 5.10: A contradiction is detected.

removal of a premise or to the addition of a justification to a node that is not believed.

Suppose that we have the dependency network $[\{n_1, n_2, n_3, n_4, n_5\}, \{n_5\}, \{<\{\} \mid \{n_2\} \Rightarrow n_1>, <\{\} \mid \{n_4\} \Rightarrow n_3>, <\{n_1, n_3\} \mid \{\} \Rightarrow n_5> \}]$. This dependency network and its only labeling are represented in Figure 5.10. According to this labeling the nodes believed are n_1, n_3, and n_5, which included the (only) contradictory node of this network, n_5.

In order to obtain a consistent set of beliefs we should label node n_5 OUT. This can only be achieved if one of n_1 or n_3 is labeled OUT, which, again can only be done if we label one of the nodes n_2 or n_4 as IN. Suppose that we decide to label n_4 IN (the decision of which node to label IN is taken outside the TMS. We can only label this node IN if we can provide a justification for it. This is done by creating a new node, called a *nogood*, that corresponds to a premiss and justifies the belief in n_4 based on the fact that n_1 is IN (see Figure 5.11). Often more than one iteration, and the addition of a nogood, are needed in order to reach a network with a consistent set of beliefs.

Figure 5.11: The contradiction is removed.

Further details on the identification of possible culprits for contradictions and the removal of contradiction can be found in [Stallman and Sussman 77] who describe dependency-directed backtracking, and in [Doyle 79].

5.2 Assumption-based systems

Assumption-based TMS (ATMS for short) associates nodes with the premises underlying their derivation. In these systems the labels of nodes are no longer IN or OUT but rather the sets of premises underlying their derivation. The terminology "Assumption-based" was introduced by [de Kleer 84, 86] although similar ideas had been investigated earlier by [Martins and Shapiro 83].

In this section we present a simplified version of the Kleer's ATMS — de Kleer considers a distinction between *premises* which are nodes that hold universally and *assumptions* made by the problem solver and which can be retracted. Since they both play a similar role in ATMS concerning the computations of labels for nodes, we will make no further distinction between them.

The big advantage of the ATMS over JTMS stems from the additional flexibility it provides in dealing with multiple possible states of affairs. In fact, by labeling nodes with the sets of premises under which they hold we have no longer a single state of belief (as is the case in JTMS, whose state of belief contains all nodes whose label is IN) but rather a number of possible states that equals the cardinality of the power set of the premises introduced. This allows a comparison of the results obtained under different choices (i.e., different premises), the existence of alternative solutions to a problem, the detection of the possible culprits for a contradiction, and the recovery from contradictions. The disadvantages of the ATMS over JTMS are the additional difficulty involved in representing justifications with non-monotonic antecedents (most versions of the ATMS do not allow for non-monotonic justifications — for a solution to this problem see [Dressler 88]) and the control over the behavior of the problem solver (see [de Kleer and Williams 86], [Dressler 90], and [Forbus and de Kleer 88]).

5.2.1 Interaction between ATMS and the problem solver

The communication between the ATMS and the problem solver is basically similar to the one described for JTMS except that with ATMS we must keep in mind that we have no longer a single state of belief. This interaction uses the following operations:

1. *Inspection operations*
   (a) Should I believe in node n, given the premises p_1, ..., p_n? This question is answered by retrieving the label associated with node n and checking whether in this label there is a set contained in $\{p_1, ..., p_n\}$.
   (b) Why should I believe in node n? This question is answered in the same way by JTMS and ATMS.
   (c) What assumptions underlie node n? This question is answered by supplying the label associated with node n.
2. *Modification operations*
   (a) Add node n. Similar to JTMS.
   (b) Add premise p. Similar to JTMS.
   (c) Add contradiction p. Similar to JTMS.
   (d) Justify node c based in the belief in nodes a_1 ... a_n (note the lack of non-monotonic antecedents). This transforms $[N, C, J]$ into $[N \cup \{a_1...a_n, c\}, C, J \cup \{<\{a_1...a_n\} \Rightarrow c>\}]$.
   (e) Remove premise p. Similar to JTMS.
3. *Updating operations.*

   These operations are executed whenever a modification operation is performed upon the dependency network. Their goal is to recompute the labels associated with the nodes in the network which are affected by the operation.

5.2.2 Computation of labels for nodes

The goal in computing ATMS labels is to find minimal sets of premises that underlie the derivation of each node. This computation is done by propagating and combining labels, starting with the labels of the premises:

1. The premise$<\{\} \Rightarrow p>$ is labeled with $\{\{p\}\}$.
2. The label of c, resulting from the justification $<\{a_1, \ldots, a_n\} \Rightarrow c>$ is a set containing all the combinations of the labels in a_1, ..., a_n, and keeping minimal sets.

Figure 5.12: ATMS labeling.

In Figure 5.12 we show the ATMS labels for the dependency network $[\{n_1, n_2, n_3, n_4, n_5, n_6, n_7\}, \{\}, \{<\{\} \Rightarrow n_1>, <\{\} \Rightarrow n_2>, <\{\} \Rightarrow n_4>, <\{\} \Rightarrow n_5>, <\{n_1, n_2\} \Rightarrow n_3>, <\{n_3\} \Rightarrow n_7>, <\{n_4\} \Rightarrow n_7>, <\{n_4, n_5\} \Rightarrow n_6>, <\{n_6\} \Rightarrow n_7>\}]$. Notice that the label of n_7, computed directly from the labels of n_3, n_4, and n_6, is $\{\{n_1, n_2\}, \{n_4\}, \{n_4, n_5\}\}$. Since $\{n_4, n_5\}$ is contained in $\{n_4\}$, and we want to keep minimal sets, the set of premisses $\{n_4, n_5\}$ is discarded from the label. Algorithms for computing ATMS labels can be found in [de Kleer 86] and [Reinfrank 89].

The power set lattice of the premises provides a good way to visualize the space of possible combinations of premises. Figure 5.13 shows this lattice for the dependency network of Figure 5.12. The edges between sets of premises represent subset/superset relations. For example, node n_6 of Figure 5.12 will be believed under all sets of assumptions which, in the lattice, are above $\{n4, n5\}$.

5.2.3 Removal of contradictions

In ATMS contradictions are removed by removing from the nodes' labels those sets of premises that are discovered to be inconsistent. Suppose, for example, that we have the following dependency network: $[\{n_1, n_2, n_3, n_4, n_5, n_6, n_7\}, \{n_6\}, \{<\{\} \Rightarrow n_1>, <\{\} \Rightarrow n_2>, <\{\} \Rightarrow n_4>, <\{\} \Rightarrow n_5>, <\{n_1, n_2\} \Rightarrow n_3>, <\{n_3\} \Rightarrow n_7>, <\{n_4\} \Rightarrow n_7>, <\{n_4, n_5\} \Rightarrow n_6>, <\{n_6\} \Rightarrow n_7>\}]$. This network is similar to the one presented in Figure 5.12, except for the fact that node n_6 is a contradiction node. Since the label of node n_6 is $\{\{n4, n5\}\}$ this set of premises is discovered to be inconsistent. When this happens, all the sets of premises which are supersets of $\{n4, n5\}$ are marked as nogoods (shaded areas in the lattice of Figure 5.14) and these sets are removed from the labels of all nodes in the dependency network. A description of the contradiction removal in ATMS can be found in [de Kleer 86].

Figure 5.13: Environment lattice for premises n_1, n_2, n_4, and n_5.

Figure 5.14: Inconsistent sets of premises.

5.3 Other approaches

In this section we list two alternative approaches to TMS-like systems. The commonality among them is that they both support reasoning by the TMS, rather than passing this task to the problem solver, as is the case with JTMS and ATMS.

5.3.1 Multiple Belief Reasoner

Multiple Belief Reasoner (MBR) is an ATMS-like system is which the problem solver and the TMS are merged into one single system. MBR is based on a logic called SWM* [Martins 91].[1] SWM* deals with knowledge states. A knowledge state, written $[[KB, KIS]]$, is a pair containing a knowledge base (KB) and a set of known inconsistent sets (KIS). The knowledge base contains propositions (written as wffs) associated with an indication of the dependencies between a particular wff and other wffs in the knowledge base; the set of known inconsistent sets is a set containing the sets of premises in the KB known to be inconsistent.

The knowledge base is a set of supported wffs. A *supported wff* consists of a wff and an associated pair, its *support*, containing an origin tag and an origin set. The *origin tag* indicates how the supported wff was placed in the knowledge base (i.e., whether it was supplied by an outside system or was generated during deduction). The *origin set* is a set of premises, it contains all and only the premises that were actually used in the derivation of that wff.

The growth of knowledge states is dictated by the rules of inference of SWM* which are grouped into two sets:

1. *Pure logic rules*, which correspond to traditional inference rules. They have the effect of adding new supported wffs to the KB. This addition may be done in two different ways: a new supported wff is introduced from the outside (this corresponds to premise — a premise in SWM* is called a *hypothesis*) or a supported wff is derived from other supported wffs in the KB. These rules transform $[[KB, KIS]]$ into $[[KB', KIS]]$ where $KB \subset KB'$.

2. *Computational rules* are rules that update the information about sets known to be inconsistent. These rules are obligatorily applied whenever a new inconsistent set is discovered, in which case a new set is added to KIS. These rules transform $[[KB, KIS]]$ into $[[KB, KIS']]$, where $KIS \subset KIS'$.

Based on SWM*'s features the behavior of an ATMS-like system called MBR (for Multiple Belief Reasoner) is defined. MBR uses two notions: context and belief space. A context is defined to be a set of premises. A context determines a belief space, which is the set of all premises defining the context and all the wffs in the KB that were derived exclusively from them. A belief space is represented by $\ll [[KB, KIS]], C \gg$, where C is a set of wffs that corresponds to premises

[1] A successor to SWM [Martins and Shapiro 83, 88].

in the KB ($(\forall H \in C)[<H, hyp, \{H\}> \in KB]$). The belief space determined by a context is the subset of all the wffs in KB whose origin set is contained in the context. Any operation performed by MBR (query, addition, etc.) is associated with a context, the current context which defines the current belief space. A proposition is said to be believed if it belongs to the current belief space.

SWM* does not consider non-monotonic justifications, but there is an extension of it, SWMC, which does [Cravo and Martins 91]. The language of SWMC is an extension of the language of SWM*. It contains a new quantifier, the default quantifier which allows it to express default rules. It also has a distinguished predicate, Applicable, that, among other things, allows it to express exceptions to default rules.

MBR has been used in the SNePS system (discussed in Chapter 6).

5.3.2 Logic-based TMS

Logic-based TMS (LTMS) is based on the work of [McAllester 78]. In this kind of TMS, relations among propositions are represented by clauses. These clauses can be used to deduce the truth value of any of the involved propositions. The description presented here is a simplification of McAllester's work (a detailed introduction to LTMS, together with a discussion of its implementation, can be found in [Charniak, Riesbeck, McDermott, and Meehan 87]).

LTMS works with nodes, truth states, terms, clauses, and supports. *Nodes* represent propositions supplied by problem solver. Any atomic proposition is represented by a node with no meaning to LTMS. A proposition involving logical connectives is also represented by a node, but in this case LTMS "understands" the meaning of the logical connectives that appear in the proposition; each node has an associated *truth state.* This truth state may be TRUE (believed to be true), FALSE (believed to be false), or UNKNOWN (no belief either way). It is said that a node has a truth value when its truth state is either TRUE or FALSE; A *term* is a pair containing one node and one truth value, TRUE or FALSE, a term is TRUE if the node has the truth value of the term; A *clause* is a set of terms one of which must be TRUE, for the clause to be TRUE. For example, the clause ((A. FALSE)(B. TRUE)) states that either A is FALSE or B is TRUE. A clause represents the relationship between the truth values of the nodes appearing in the clause.

A clause may be used by LTMS to deduce a truth value for any of the nodes it contains, and in this case the clause will act as a *justification* for the deduced truth value. A clause may be given multiple interpretations. For example, the clause ((A . FALSE) (B . TRUE)) may be interpreted either as $A \rightarrow B$ or, equivalently, $\neg B \rightarrow \neg A$. If this clause were used to deduce the truth value TRUE for node B (given the truth value TRUE of node A), it would act as a justification for the truth value of B. In a JTMS the corresponding justification would be $<\{A\} \mid \{\} \Rightarrow B>$.

There are two differences worth pointing out, between a JTMS justification and a LTMS clause: (1) A JTMS justification is a special kind of clause, a Horn clause. This means that only one term in the clause has TRUE as its second component. (2) A JTMS justification is not symmetrical, i.e. the label (IN/OUT) of its consequent may be concluded from the labels of its antecedents, but the label of the consequent does not allow JTMS to conclude anything about the label of any one of the antecedents, in LTMS justifications are symmetrical: the truth value of any node is a clause may be computed from the truth values of the other terms in the clause.

At any time the problem solver can add a clause and supply it to LTMS. Whenever this happens a check is immediately made to see if it is possible to derive new information. This will happen whenever all the terms of the clause are FALSE, except for one, which is UNKNOWN. In this case, this term must be TRUE (since the truth value of the clause is TRUE), and LTMS can conclude the truth value of this term's node. The other way a truth value can be added to a node is by deducing it from a clause. In this case the support of the truth value is the clause used to deduce it. Whenever a truth value is added, a check is made to see if this enables the deduction of some other truth value, from a clause. Obviously, the only clauses that are relevant are the ones that contain a term that has been given a truth value. The procedure used to compute truth values for clauses is called *boolean constraint propagation.*

The truth value of a node can be removed, meaning that the truth state of that node changes from TRUE or FALSE to UNKNOWN. The removal of a truth value corresponds to a removal of information. There are several ways in which the truth value of a node can be removed. One way for a truth value to be removed is when problem solver decides to retract a premise or an assumption. This decision may be taken either because the problem solver decides that a premise is no longer known, or to resolve a contradiction (a contradiction is a clause whose terms are all false).

We said that LTMS "understands" the meaning of the logical connectives. To make this possible each of the logical symbols has an associated axiom set which is used to generate clausal constraints. For example, the axioms for implication are: (1) (($P \rightarrow Q$. FALSE) (P . FALSE) (Q . TRUE)); (2) (($P \rightarrow Q$. TRUE) (P . TRUE)); (3) (($P \rightarrow Q$. TRUE) (Q . FALSE)).

5.4 Applications

The capability of determining the source of contradictions coupled with the possibility of changing beliefs are essential features of any intelligent system. In general, any system that follows the paradigm described at the outset of this chapter can uses (and benefit from) TMS techniques. However, there are some problem areas in which these techniques are of paramount importance, in particular (other application areas can be found in [Martins 90]):

1. *Reasoning based on partial information, default assumptions and potentially inconsistent data.* This kind of reasoning is likely to produce contradictions and a system that resorts to it should be able to recover from them. This aspect is discussed in [Cravo and Martins 91], [Etherington 87], and [Morris 87a].

2. *Constraint Satisfaction.* This is a problem area that involves making choices and their retraction when a violation of constraints is detected. Again, the capability of discovering the source of the violation and its removal can be guided by TMS techniques. There is a good tradition in the application of TMSes to this area, see, for example, [Dhar and Ranganathan 90].

3. *Diagnosis.* The use of TMS techniques, in particular the recording of dependencies, is important to detect possible causes of malfunctioning. Among others, [de Kleer and Williams 87] and [Struss 88] discuss this aspect .

4. *Planning.* The application of TMSs to planning pertain to the aspect of reasoning about change, i.e., reasoning about the propositions that change whenever an action is performed. Among others, the following papers discuss this aspect [Drummond 87], [Janlert 87], [Morris 87b], and [Pinto-Ferreira and Martins 90] .

5. *Search.* The ATMS introduces a new approach to search in which several alternatives are considered and switching between nodes in the search tree is done by changing the premises under consideration. Among others, [Forbus and de Kleer 88] discuss this aspect.

5.5 Suggested readings

Several papers, although do not specifically address TMSes, were influential in their development. Their study helps at understanding the general setting under which TMSes were introduced. TMSes have their roots in the frame problem [McCarthy and Hayes 69], the problem of deciding which conditions change and which conditions do not change in a system when it undergoes some modification. Among the approaches to this problem [Fikes 75] proposes a system that uses dependencies among propositions and [Hayes 75] argues about the need for TMS-style systems.

In the mid 70's [Stallman and Sussman 77] designed EL, a system in which dependencies of propositions are permanently recorded. The recording of dependencies has particular importance in finding faulty assumptions responsible for the generation of contradictions and for explanation of the program's beliefs. Their work had two major influences in AI: It opened a new perspective to the handling of alternatives (dependency-directed backtracking), and it influenced the creation of systems that, using dependencies, handle and can recover from contradictions (EL was a direct influence in the development of Doyle's TMS.). At the same time, and independently from Doyle's work, [London 78] describes a planning system relying on the use of dependencies. A reasoning system, called AMORD [de Kleer, Doyle, Steele, and Sussman 77], was an influential system on the development of the ATMS.

During the last few years there has been some effort in providing foundations for TMSes and relating them with other non-monotonic formalisms: [Reinfrank 89] discussed the relationship between TMS and non-monotonic logics. [Fujiwara and Honiden 89] and [Reinfrank, Dressler and Brewka 89] have shown the equivalence between the reasonable beliefs of a dependency networks and the strongly grounded extensions of autoepistemic logic. [Brown 88] provided a formal semantics to TMS by offering a logic that characterizes some models of TMSes. [Doyle 83] was first attempt to present a formal description of a TMS. [Elkan 90] provided a characterization of the inferences performed by a non-monotonic TMS in terms of logic programming with stable set semantics and autoepistemic logic. [Popchev, Zlatareva, and Mircheva 90] provides a logical theory of truth maintenance reasoning subsuming plausible reasoning that distinguishes between true and plausible arguments. They propose a behavior in presence of inconsistencies that corresponds to blocking the propagation of the contradiction rather than trying to get rid of them.

A detailed study on the topics discussed in this chapter can be done, using the following references:

[Martins 90]
An annotated bibliography on Truth Maintenance Systems.

[Reinfrank 89]
An excellent overview of Truth Maintenance Systems.

[Smith e Kelleher 88]
An introduction to Truth Maintenance Systems with a discussion of several applications.

5.6 Exercises

1. Discuss the relationship between non-monotonic logics and belief revision systems.

2. Compare ATMS and JTMS regarding the following aspects: (a) Ease of dealing with multiple alternatives; (b) Ease of representing non-monotonic justifications.

3. Consider the following JTMS dependency network:

Compute an admissible labeling for this network.

Chapter 6

Semantic Networks

In this chapter we present the basic concepts of semantic networks, discussing their historical evolution and the main problems associated with them. We discuss, in detail, one example a semantic network, SNePS. Finally, we evaluate semantic networks from the knowledge representation point of view.

> "When you follow two separate chains of thought, Watson, you will find some point of intersection which should approximate to the truth."
> Connan Doyle, *The Disappearance of Lady Frances Carfax* (Sherlock Holmes)

Semantic networks are knowledge representation formalisms that attempt at capturing the idea that concepts used in the human memory are associated, or linked, with other concepts in our mind.

There is a large number of representations that can be classified as semantic networks. The commonality among them is the idea that knowledge is represented as a labeled graph composed by nodes and labeled directed arcs. Semantic networks were initially proposed by researchers associated with psychology and linguistics that attempted at characterizing the nature of human understanding.

6.1 Early semantic nets

Semantic networks were first proposed, as a knowledge representation formalism in AI, by M.R. Quillian in 1966, who was influenced by the work of Bertran Raphael some two years earlier. However, a formalism similar to semantic networks had been proposed in the late 19th Century by the mathematician Charles S. Pierce [Roberts 73]. The work of Pierce has deeply influenced the semantic networks developed by John Sowa [Sowa 84].

In this section we look at the early work developed in the area of semantic networks.

6.1.1 The work of Raphael

Bertran Raphael [Raphael 68], as part of his Ph.D. dissertation, completed at MIT in 1964, developed a system called SIR (Semantic Information Retrieval), that represents knowledge corresponding to a dialogue with a user and is capable of answering questions about the knowledge conveyed in this dialogue.

To represent the knowledge supplied during a session, SIR uses a labeled directed graph, composed by arcs and nodes. *Nodes* represent entities or classes of entities. Each node has a unique identifier. *Arcs* represent binary relations between the entities represented by nodes. Each arc has a label that identifies the binary relation it represents. Everything that is known in respect to a given entity represented in the graph is connected, directly or indirectly, to the node that represents the entity.

In his work, Raphael only considered a small number of binary relations, some of them are shown in Table 6.1. The first column in this table shows the mathematical relation, the second shows the representation used by SIR for that

Relation	*Representation of relation*	*Representation of inverse relation*	*Meaning (corresponding sentence)*
$x \subset y$	$x \xrightarrow{\text{subset}} y$	$y \xrightarrow{\text{superset}} x$	Every x is a y An x is a y
$x \in y$	$x \xrightarrow{\text{member}} y$	$y \xrightarrow{\text{element}} x$	x is a y The x is a y
$x \equiv y$	$x \xrightarrow{\text{equiv}} y$	$y \xrightarrow{\text{equiv}} x$	x is y

Table 6.1: Some of the relations defined in SIR.

relation, the third shows the representation of the inverse relation, and, the last column, shows the sentence that originates the creation of an arc corresponding to the relation between two nodes (in the description of this sentence, x and y represent variables). Each of these relations is associated with procedures that define how to answer questions (and to produce inferences) that use the relation. These procedures define, in fact, the semantics of the relations. In order to simplify our description, of the figures that we present about SIR's network, we do not show the representation of the inverse relations, although it should be kept in mind that they are represented inside the system.

Whenever SIR receives a statement from the user, it matches it against the patterns defined in the last column of Table 6.1, creating the corresponding nodes and relations in the network.

As an example, consider the following interaction with SIR (adapted from [Raphael 68, pp. 68 and 134]), where "`*`" represents the prompt of the system, the sentence following the prompt correspond to the information supplied by the user, and the line, or lines, that follow it correspond to the answer given by SIR. This interaction assumes the definition of the relations shown in Table 6.1.

```
* The man is a jerk
  (G02840 is a man)
  I understand
```

This first sentence originates the network shown in Figure 6.1. Since it is not known which man we are talking about, the user said "*the* man", SIR creates a node `G02840` to represent the man in the sentence (an entity that is both a man and a jerk). The answer supplied by SIR contains an indication about the node that was created, followed by "`I understand`", indicating that a structure was

Figure 6.1: Network in SIR (part 1).

Figure 6.2: Network in SIR (part 2).

created in the network to represent the information supplied. If the following sentences are given,

```
* Jack is a Lisper
  I understand
* John is Jack
  I understand
```

SIR expands the network as shown in Figure 6.2. Based on the network of this figure, SIR is able to answer in this way to the following questions:

```
* Is John a Lisper?
  Yes
* Is the man a Lisper?
  Insufficient information
```

It should be noticed that the network of Figure 6.2 enables to state that `John` is a `Lisper` but it has no relation between "the man" (represented by node `G02840`) and `Lisper`.

The following sentence adds information to the network about the relation between `John` and "the man", giving rise to the network shown in Figure 6.3.

```
* John is the man
  I understand
```

After the addition of this information, SIR is able to answer in the following way to the question of whether the man is the Lisper:

Figure 6.3: Network built during the described interaction.

```
* Is the man a Lisper?
  Yes
```

The main problem associated with SIR corresponds to the fact that the meaning of the relations is expressed by procedures that define how SIR should answer questions involving the relation, what kinds of inferences the relation allows, and how the relation interacts with the other relations. In this way, whenever a new relation was defined, procedures had to be written to specify how the system should behave in face of the relation, and worst of all, how the relation is connected with the other relations in the system. This last aspect, prevented the expansion of the relations beyond a small number, and lead to other approaches (in other systems) to the definition of relations.

Although SIR represents knowledge as a labeled directed graph, it is not considered a *real* semantic network, because there is nothing in the system that represents the meaning of the representation. It just records the result of an interaction.

6.1.2 The networks of Quillian

The first semantic networks in AI correspond to the work developed in 1966 by M.R. Quillian [Quillian 68] at the Carnegie Institute of Technology. Quillian's goal was to develop a reasonable representation for the way semantical information is organized in our memory. In his work, he was concerned with the representation of common words, such as plant, chair, food, and so on. He addressed, only briefly, the acquisition of new knowledge.

Knowledge is represented by a network of arbitrary size, composed of nodes and associative links. Each node corresponds to a word concept and associative links represent the relations between words. The meaning of a given word is specified by the relations it bears with other words.

Quillian distinguishes two kinds of nodes, type and token nodes:

1. A *type* node corresponds to the definition of the word associated with it. Each type node is the head of a plane (represented by a rectangle) that contains the definition of the word. Connections between planes are used

Figure 6.4: Some arcs used by Quillian.

to refer to meanings of words. Each word has exactly a type node. Type nodes are graphically represented enclosed by a circle;

2. A *token* node corresponds to a reference to the word associated with it (as part of the definition of another word). A word may have many token nodes.Token nodes are not enclosed by a circle.

Quillian defines several kinds of relations, some of them are shown in Figure 6.4). The graphical aspect of these relations can be looked at as being the arc label associated with it.

In Figure 6.5, we show a simplified example, with the possible meanings of the word "plant" (adapted from [Quillian 68, p. 236]). According to this figure, this word may have three meanings, each one of them defined in an appropriate plane: plant as a living organism, plant associated with manufacture, and plant as a verb.

In Figure 6.6 we show, with more detail than in Figure 6.5, two of the meanings of the word plant (example from [Quillian 68, pp. 236-237]). The meaning of the word "PLANT" states that it is a live structure, not an animal, it has leaves, and gets food from the air, water, and earth; the meaning of the word "PLANT3" states that it is the putting action, done by a person, of a seed or plant in the earth so that it grows.

Quillian introduced the concept of *complete meaning*, everything that could be reached from a given node. Quillian used his model to find out differences and similarities between words. With this goal, he introduced the concept of *spreading activation*, that corresponds to a search by intersection: given two words, their relation could be computed by a breath-first search, using the links

Figure 6.5: Possible meanings of the word “plant”.

that emanate from those words. For example, given the words “plant” and “live”, Quillian gets the following list of similarities (from [Quillian 68, p. 253]):

```
Compare: PLANT, LIVE
A.   1st Intersect
     (1)  PLANT IS A LIVE STRUCTURE
B.   2nd Intersect
     (1)  PLANT IS STRUCTURE WHICH GETS3 FOOD FROM AIR.
          THIS FOOD IS THING WHICH BEING2 HAS-TO TAKE
          INTO ITSELF TO7 KEEP LIVE.
```

The important aspects regarding the work of Quillian are the following:

1. The meaning is represented by relations;
2. The use of a class/sub-class hierarchy to infer properties of sub-classes: the properties of classes would be extended to all their sub-classes, except those were they were modified;
3. The description of the properties of each class was done through the use of property-value attributes.

The main problems of Quillian's model relates to the number of relations he defines, which is clearly insufficient, and to the fact that the search trough the network does not consider the type of relation traversed.

The work of Quillian, was later extended, together with Collins [Collins and Quillian 70, 72] and the resulting work was used to argue about the psychological plausibility of semantic networks. The networks of Quillian and Collins were composed of an hierarchy of concepts, such as animal, bird, canary, and so on (Figure 6.7, adapted from [Brachman 79, p. 10]). Each concept was associated with properties that defined it. The more general properties were associated

Figure 6.6: Detailed meanings of plant.

Figure 6.7: Taxonomy used by Collins and Quillian.

with concepts near the top of the hierarchy. This lead to the ideas of *property inheritance* and *semantical distance* (the number of nodes to be visited to reach a concept from another concept). Quillian and Collins compared the response time, given by the system when answering questions about the properties of a given class, with the response time given by humans and they concluded that the behavior was similar. For example, the answer to the question of whether a canary has skin takes longer that the time needed to answer whether a canary sings.

6.2 Concerns with the foundations

By the end of the 1960s and beginning of the 1970s there was a growing interest in semantic networks. As semantic networks address more complex problems and try to represent a larger scope of domains, the explanation of the meaning of the structures used, become more and more difficult. In the beginning of the 1970s researchers attempted at precisely defining the meaning of the representations in semantic networks (the semantics of semantic networks). Among the important papers in this area are [Woods 75] and [Shapiro 71]. The paper by [Brachman 79] presents a good summary of these attempts.

According to [Woods 75], in the specification of a semantic network, we have to consider:

1. The kinds of nodes and arcs that can be used and the rules for their combination, the *syntax of the network*.

2. The meaning of the different kinds of arcs and structures, the *semantics of the network*.

Most semantic networks are based on the notion of attribute-value, introduced by Raphael: the name of the attribute is represented by the name of the arc that points to the corresponding value. For example, the sentence "John is

Figure 6.8: John is 6 feet high, has dark hair and is a scientist.

Figure 6.9: John is taller than 5 feet.

6 feet high, has dark hair, and is a scientist" can be represented as shown in Figure 6.8.

If we now want to represent that "John is taller that 5 feet", we could be tempted to use the representation shown in Figure 6.9. However, when we do this, the meaning of our representation changes! The thing at the end of an arc is no longer a value but becomes a proposition that is true for the value. With this new representation, there are now two interpretations to the node at the end of an arc, either a value or a predicate associated with the value.

With this representation, it is not possible to use only two nodes to represent that "John is taller than Mary" since in the predicate associated with John's height we would have to represent Mary's height. In order to solve this problem, Woods proposes the use of intentional objects, $h1$ and $h2$, to represent, respectively, "John's height" and "Mary's height". Woods argues that it is not necessary to represent the extension of these intentional objects, that is, their value. The proposition that "John is taller than Mary" would be represented as shown in Figure 6.10. Under this perspective, we can revise the representation shown in Figure 6.9, and represent the proposition that "John is taller than 5 feet" by the network shown in Figure 6.11. In this way, we have a unique meaning for the node where the arc points to, the value of the property associated with the arc.

Figure 6.10: John is taller than Mary.

Figure 6.11: John is taller than 5 feet (revised representation).

Figure 6.12: John kissed Mary.

Figure 6.13: John is 6 feet high and kissed Mary.

Semantic networks have been widely use by researchers working in natural language understanding, under this perspective, researchers have used arcs to represent verb usage. For example, according to what we have been discussing, the sentence "John kissed Mary" could be represented as shown in Figure 6.12. However, if we use this representation, the semantics of our notation changes again. An arc not only represents the attributes of a node, but also relationships between a node and other nodes. If we want to use both kinds of representations, to state, for example, that "John is 6 feet high and kissed Mary" (Figure 6.13) we had to tell the system that manipulates the network that the arcs used in the representation are of different nature, and thus have to be treated differently by the procedures that are responsible for the inference.

In order to avoid this situation, we have to find another meaning for arcs. The alternative most commonly used is based on the work of [Fillmore 68] on cases associated with verbs. According to Fillmore, a *case* is a particular role that a noun phrase or other participant takes in the action or activity expressed by the verb of a sentence. For example, the sentence "John gave Mary a book" has giving action, where John is the agent, Mary is the recipient, and the book is the object. This can be represented by the network of Figure 6.14.

We should notice, however, that the node labeled "gave" does not represent a general characteristic of the verb "to give", but rather a particular instance of the giving action. It is more correct to represent this instance by a new

Figure 6.14: John gave Mary a book.

Figure 6.15: Alternative representation for "John gave Mary a book".

node (that identifies the instance) and to make explicit that it corresponds to a giving action. In Figure 6.15, "S001" represents an instance of giving and "give" represents the verb, in general.

The use of cases in a semantic network gives rise to a new interpretation for the nodes and arcs in the network:

1. There are nodes that represent objects, as before. These nodes, in our example are John, Mary, and book;

2. There are nodes that represent propositions, in our example, S001.

The arcs that emanate from this new kind of nodes, are different, in nature, from the other arcs:

1. The arcs we had before represent statements. Their presence in the networks corresponds to a proposition about the two nodes that they connect;

2. New arcs represent "parts" of a proposition. For example, the meaning of S001 is given by *all* arcs that emanate from it, not only by a subset of these arcs. These new kind of arcs are called *structural.* We can argue that these arcs represent the statement that the agent of S001 is John,

the recipient of S001 is Mary, and so on. In this representation S001 has no meaning other that what is ascribed by the structural links to other nodes.

6.3 SNePS: A case study

In this section, we consider, in detail, a propositional semantic network, SNePS ("Semantic Network Processing System"), developed by Stuart C. Shapiro and his research group. The goal of building SNePS was to produce a network that would be able to represent virtually everything that is used in Natural Language. For this reason, SNePS offers a high expressive power. SNePS has undergone several versions, the most important of which are SNePS 1.0 [Shapiro 79a], [Shapiro and Rapaport 87] and SNePS 2.1 [Shapiro and Martins 90]. A detailed description of the historical evolution of SNePS is presented in [Shapiro and Rapaport 92].

SNePS uses structural relations, as described at the end of last section. SNePS is called a *propositional* semantic network because each proposition is represented by a node and not by an arc. In SNePS, nodes represent concepts, and arcs represent *non-conceptual* relations[1] between concepts. Arcs are only used to structure concepts. The fact that concepts are represented by nodes means that, in SNePS, we can only talk about the information represented by nodes and we cannot refer to the information represented by arcs. Arcs are considered to be part of the syntactic structure of the node from where they emanate.

SNePS imposes certain restrictions on the representation it uses. It is important to understand these restrictions, because they distinguish SNePS form a general labeled, directed graph, and from many other semantic network formalisms.

6.3.1 Nodes in SNePS

In SNePS, nodes represent concepts. There are two important aspects regarding the concepts represented in SNePS:

1. Nodes represent intensional concepts;

2. Each concept is represented by a unique node and each node represents a unique concept. This is known as the *uniqueness principle* [Maida and Shapiro 82].

The notion of a concept is vague and it corresponds to anything that we can talk about. Concepts, thus, represent, among other things, propositions. The benefit of representing propositions by nodes is that propositions about propositions can be represented with no limits.

[1] That is, information that does not correspond to a concept in the network.

If there is a direct arc from node m to node n, it is said that m *immediately dominates* n. If there is a path of direct arcs from node m to node n, it is said that m *dominates* n. There are no cycles composed only by direct arcs, so no node dominates itself. If there is a direct arc, r, from node m to node n, it is said that n "is the value of" $r(m)$.

According to their position in the network, nodes can be atomic or structured.

1. A node is said to be *atomic*, if it dominates no other node. Atomic nodes can be further divided into constant and variable nodes:

   (a) Atomic constant nodes, also called *base nodes*, represent non-structured concepts, and correspond, in a sense, to the constants of first-order logic. Atomic constant nodes are represented by a rectangle, inside of which there is the name of the node. These nodes are labeled by a string that stands for the name of an individual concept, for the name of a relation, for the name of a property, and so on.

   (b) Atomic non-constant nodes, also called *variable nodes*, represent variables, and correspond, in some sense, to the variables in first-order logic.[2] Variable nodes are represented by a circle, inside of which there is the name of the node, labeled in SNePS by an identifier of the form `V`n, where n is a positive integer.

2. A node is said to be *structured* if it dominates other node(s) (structured nodes are represented by a circle, inside of which there is the label of the node). Structured nodes can be further divided into constant and variable nodes:

   (a) Constant structured nodes represent either propositions or structured concepts. These nodes correspond either to well formed formulas in first-order logic without free variables or to complex terms. Constant structured nodes are labeled in SNePS by an identifier of the form `M`n, where n is a positive integer. This identifier may be followed by the character "`!`", meaning that the corresponding node is believed by SNePS. In this case, we say that the node is *asserted* in the network.

   (b) Non-constant structured nodes, also called *pattern nodes*, dominate at least a node corresponding to a free variable. These nodes correspond to well formed formulas in first-order logic with free variables. Non-constant structured nodes are labeled in SNePS by an identifier of the form `P`n, where n is a positive integer.

6.3.2 Arcs in SNePS

In SNePS there are two kinds of arcs, direct and inverse arcs. For each relation represented by a direct arc, there is a converse relation represented by an inverse

[2] These may also represent arbitrary propositions.

Figure 6.16: Case frame for class membership.

arc. Inverse arcs have the same name as the corresponding direct arcs, followed by a "-", and will not be shown in our diagrams.

The labels of the arcs used in SNePS are defined by the person who is using it to represent knowledge. However, in order to enable the inference engine to interpret the propositions involving logical connectives, there are some pre-defined arcs in SNePS.

6.3.3 Knowledge representation in SNePS

SNePS has no pre-defined arcs for representing knowledge, except for those related with the logical connectives (these are described in Section 6.3.4). Therefore, the first step towards representing knowledge in SNePS corresponds to the choice and definition of the arcs that will be used.

To define a piece of knowledge in SNePS we have to use a "case frame" of arcs that emanate from a node and correspond to the relation or proposition that the node represents. Although there are no fixed guidelines for the definition of these case frames (they depend on the intended application), there are some commonly used representations, some of which are described in this section. For a more complete description, refer to [Shapiro and Rapaport 87] and [Shapiro *et. al.* 93].

Two of the basic aspects that appear in most applications are the statements that an individual is member of a given class and that a class is a subclass of another class. The first statement is usually represented by the case frame "member/class" (Figure 6.16), where the arc labeled "member" points to the node that corresponds to the concept about which we are asserting membership and the arc labeled "class" points to the node that corresponds to the class. The second statement is usually represented by the case frame "subclass/superclass" (Figure 6.17), where the arc labeled "subclass" points to the node that corresponds to the subclass and the arc labeled "superclass" points to the node that corresponds to the super class.

Using these case frames, we show in Figure 6.18, the representation that John is a man (node `M1!`) and that the class of man is a subclass of the class of mammals (node `M2!`). It is important to notice that, from this information alone, SNePS is not able to know that John is a mammal because the "operational semantics" associated with these case frames are not part of the network yet.

Figure 6.17: Case frame for class containment.

Figure 6.18: John is a man and men are mammals.

These operational semantics can be specified trough the use of the connectives described in Section 6.3.4.

It is common to need to explicitly represent arbitrary relations in SNePS. This can be done using the case frame "rel/arg1/.../argn", where the arc labeled "rel" points to the node that corresponds to the relation and the arc labeled "argi" ($1 \leq i \leq n$) points to the node that corresponds to i-th argument of the relation (Figure 6.19). This case frame can be used to represent any relation, namely the membership and subset relations that we discussed before (see Figure 6.20). The representational difference between the use of the case frame of Figure 6.20 and the case frame presented in Figure 6.18 is the fact that using the case frame of Figure 6.20, the membership relation itself becomes a concept in the network, and thus something we can talk about, whereas with the representation of Figure 6.18, the membership relation is implicit in the structure

Figure 6.19: Case frame for a n-ary relation.

Figure 6.20: Alternative representation for “John is a man”.

Figure 6.21: Bad representation for “John is a man”.

of the arcs and thus cannot be referred by SNePS. We could still think of the representation shown in Figure 6.21 to represent that “John is a man”.[3] This representation is a step backwards when compared with the representation of Figure 6.18, since the membership to the class of men is no longer a concept, but rather a relation between two nodes. With the representation of Figure 6.21, we cannot talk about the class of men nor about the relationship that John bears with it.

From this discussion, it may seem that the explicit representation of relations, since it corresponds to a more fine grained representation, should be preferred over the representation of relations through case frames that do not use the explicit concept of relation. You should be aware, however, that a generalized use of the case frame ‘rel/arg1/.../argn” for the representation of relations has a negative effect on the efficiency of the inference process because the pattern matcher will be overloaded with information (basically, all the nodes that represent relations can match with a generic relation).

The idea behind the “rel/arg1/.../argn” case frame can be used to represent instance of verbs, in the same way as is presented in Figure 6.15. In Figure 6.22, we represent that “John knows that Mary believes that whales are fish” (node `M3!`) and that “Moby Dick is a whale” (node `M4!`). Besides the case frames member/class and subclass/superclass, we use the case frame agent/verb/object to represent instance of verbs. Notice that only nodes M3 and M4 are asserted in the network (their identifier is followed by “!”). The distinction between

[3] I am grateful to António Leitão for suggesting this representation.

Figure 6.22: Example of representation in SNePS.

asserted and non-asserted nodes enables to understand which nodes are believed by SNePS (the asserted nodes) and which nodes are part of the structure of propositions, but not believed by SNePS. The distinction between asserted and non-asserted nodes is made by the user of SNePS and also by the inference system while deducing new information.[4] With this network, if we ask SNePS whether "whales are fish" it answers "I don't know", but if we ask whether "John knows that Mary believes that whales are fish", it answers "yes".

The meaning of the nodes labeled "John", "Mary", "whale", "fish", "believe", and "Moby Dick" will be defined by the additional network structure that will be connected to them.

6.3.4 Logical connectives in SNePS

An important aspect regarding the research in semantic networks concerns the study of how to represent relations between propositions. One of the obvious alternatives is to use the traditional logical connectives, which has the advantage of having the guarantee that the inference produced corresponds to solid, well known, and well understood. However, the choice of logical connectives depends on other aspects, such as the ease of representation, the ease of proving properties about the system, the proof of equivalence with other logical systems, and so on. If the number of logical connectives is small, it is easy to prove properties *about* the system and it is difficult to prove properties *in* the system; if the number of logical connectives is large, the opposite holds.

In the choice of the logical connectives to use in SNePS, the decisions of and adequate expressive power and the naturalness of the representation were the guidelines. As [Shapiro 79b] argues, "the set of connectives used in traditional first-order logic presents some disadvantages from the network representation point of view. ... the main disadvantage relates to the fact that all the connectives, except negation, are binary and therefore expressing sentences about sets of propositions becomes cumbersome. ... having to choose between $(\alpha \vee \beta) \vee \gamma$ and $\alpha \vee (\beta \vee \gamma)$ is unnecessary and unfairly distinguishes one of the disjuncts".

On the other hand, suppose, for example, that given three propositions, α, β and γ, we wanted to express the fact that exactly one of them is true. Using the traditional logical connectives this would be done as $(((\alpha \wedge \neg\beta) \wedge \neg\gamma) \vee ((\neg\alpha \wedge \beta) \wedge \neg\gamma)) \vee ((\neg\alpha \wedge \neg\beta) \wedge \gamma)$,[5] which is a lengthy and difficult to read wff. Sentences involving more than three propositions (e.g., exactly five out of ten propositions are true) are even more complicated. Since this type of sentence often occurs in some of the intended applications of SNePS, the need was felt to find a more powerful and simpler way of defining such sentences (for a detailed discussion about the disadvantages of the standard connectives refer to [Shapiro 79b]).

[4] In fact, the distinction between asserted and non-asserted nodes is more complex because it resorts to an ATMS-based context shadowing operations.

[5] Again, in each of the conjunctions we had to make the choice of associations because logical connectives are binary.

Non-standard connectives

The problems that we just otlined, lead Stuart Shapiro to the definition of non-standard connectives [Shapiro 79b]. All logical connectives in SNePS take sets of propositions (represented by nodes) as arguments. The following connectives are defined in SNePS:

1. *And-or*, represented by $\bigvee\!\!\!\!\!\bigwedge$.

   The proposition

   $${}_n\bigvee\!\!\!\!\!\bigwedge_i^j \{\alpha_1, \ldots, \alpha_n\}$$

   asserts that at least i and at most j of its n arguments (propositions) are true. In this way:

   - ${}_1\bigvee\!\!\!\!\!\bigwedge_1^1\{\alpha\}$ represents the assertion of α;
   - ${}_1\bigvee\!\!\!\!\!\bigwedge_0^0\{\alpha\}$ represents the negation of α, $\neg\alpha$;
   - ${}_2\bigvee\!\!\!\!\!\bigwedge_2^2\{\alpha, \beta\}$ represents the conjunction of α and β, $\alpha \wedge \beta$;
   - ${}_2\bigvee\!\!\!\!\!\bigwedge_1^2\{\alpha, \beta\}$ represents the disjunction of α and β, $\alpha \vee \beta$.

   We have to consider now to represent a proposition whose main connective is an and-or. In order to do so, we have to address two questions: (1) How the individual propositions that compose the and-or are represented in the proposition that corresponds to the and-or; (2) How do we represent the and-or connective. The first question is answered by the introduction of case frames as defined in Section 6.2. A proposition corresponding to an and-or is represented by a structured node that relates the nodes corresponding to the individual propositions. In SNePS, there are pre-defined arcs that emanate from the node that corresponds to the and-or, and whose semantics is known by the inference engine. The second question is answered by the kinds of arcs that emanate from this structured node: arcs labeled `arg` pointing to the nodes corresponding to the propositions that are the arguments of the and-or, and arcs labeled `min` and `max` pointing to the parameters i and j, respectively. In Figure 6.23 we show the representation of and-or in SNePS. In this figure, a triangle below a node represents the network structure of the proposition that is written bellow it.

2. *Thresh*, represented by Θ.

   The proposition

   $${}_n\Theta_i\{\alpha_1, \ldots, \alpha_n\}$$

   means that either less than i of its n arguments are true or all of them are true. In this way, ${}_n\Theta_1\{\alpha_1, \ldots, \alpha_n\}$ states that $\alpha_1, \ldots, \alpha_n$ are equivalent.

Figure 6.23: Representation of ${}_n\bigvee\!\!\!\!\bigwedge_i^j \{\alpha_1, \ldots, \alpha_n\}$.

Figure 6.24: Representation of ${}_n\Theta_i\{\alpha_1, \ldots, \alpha_n\}$.

Similarly to what we did for and-or, the representation of a proposition whose main connective is thresh is done by a structured node with arcs labeled `arg` pointing to the nodes that correspond to the arguments of the proposition and with arc labeled `thresh` pointing to the parameter i of the connective. In Figure 6.24 we show the representation of thresh in SNePS.

3. *And-entailment*, represented by $\wedge\!\!\rightarrow$.

   The proposition

   $$\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \wedge\!\!\rightarrow \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$$

   means that the conjunction of the antecedents $\alpha_1, \alpha_2, \ldots, \alpha_n$ entails the conjunction of the consequents, $\gamma_1, \gamma_2, \ldots, \gamma_m$.

   And-entailment is represented by a node, with arcs labeled `&ant` pointing to the nodes corresponding to the antecedents of the entailment and with arcs labeled `cq` pointing to the nodes corresponding to the consequents of the entailment (Figure 6.25).

4. *Or-entailment*, represented by $\vee\!\!\rightarrow$.

   The proposition

Figure 6.25: Representation of $\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \wedge\!\!\rightarrow \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$.

Figure 6.26: Representation of $\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \vee\!\!\rightarrow \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$.

$$\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \vee\!\!\rightarrow \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$$

states that the disjunction of the antecedents $\alpha_1, \alpha_2, \ldots, \alpha_n$ entails the conjunction of the consequents $\gamma_1, \gamma_2, \ldots, \gamma_m$.

Or-entailment is represented by a node with arcs labeled **`ant`**, pointing to the nodes corresponding to the antecedents, and arcs labeled **`cq`** pointing to the nodes corresponding to the consequents (Figure 6.26).

5. *Numerical entailment*, represented by $i\!\!\rightarrow$.

   The proposition

$$\{\alpha_1, \alpha_2, \ldots, \alpha_n\}\ i\!\!\rightarrow \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$$

   states that the conjunction of i antecedents $\alpha_1, \alpha_2, \ldots, \alpha_n$ entails the conjunction of the consequents $\gamma_1, \gamma_2, \ldots, \gamma_m$. Notice that numerical entailment generalizes both and-entailment (if $i = n$) and or-entailment (if $i = 1$), so, in fact, only numerical entailment is needed. However, since the most commonly used statements involve assertions corresponding to either or-entailment or and-entailment, these are kept.

   Numerical entailment is represented by a node with arcs labeled **`&ant`** pointing to the nodes corresponding to the antecedents of the entailment,

Figure 6.27: Representation of $\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \; i\!\!\rightarrow \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$.

arc labeled `thresh` pointing to the parameter i, and arcs labeled `cq` pointing to the nodes corresponding to the consequents (Figure 6.27).

Quantifiers in SNePS

There are three quantifiers in SNePS, all of them applicable to a set of variables:

1. *Universal quantifier*, represented by $\forall$. The proposition

$$\forall x[\alpha(x)]$$

means that every constant that can replace x makes the proposition $\alpha(x)$ true. The universal quantifier is represented in SNePS by an arc labeled `forall` that links the node where the quantification is stated to the nodes that correspond to the quantified variables.

2. *Existential quantifier*, represented by $\exists$. The proposition

$$\exists x[\alpha(x)]$$

means that there is at least one constant that can replace x and make the proposition $\alpha(x)$ true. The existential quantifier is represented in SNePS by an arc labeled `exists` that links the node where the quantification is stated to the nodes that correspond to the quantified variables.

3. *Numerical quantifier*, represented by ${}_n\exists_i^j$. The proposition

$${}_n\exists_i^j \; x \; [\{\alpha_1(x), \ldots, \alpha_k(x) : \beta(x)\}]$$

means that there are n constants that when replaced by x make $\alpha_1(x) \wedge \ldots \wedge \alpha_k(x)$ true. However, at least i and at most j of them also satisfy $\beta(x)$. Numerical quantifier is represented by an arc labeled `nexists` that links the node where the quantification is made to the nodes that correspond to the quantified variables. From the node where the quantification is made, there are also arcs labeled `etot`, `emax` and `emin`, pointing, respectively, to

n, i and j. There are also arcs labeled `&ant` pointing to the nodes that correspond to the antecedents ($\alpha_1(x)$, ..., $\alpha_k(x)$) and arcs labeled `cq` that point to the consequent ($\beta(x)$).

Numerical quantification enables to represent propositions such as "there are exactly two numbers such that $x^2 + 4 = 4x$"; "there are at least two numbers such that $z + 2 < 6$".

Another advantage of the numerical quantifier is to allow reasoning by exclusion. For example, knowing that every person has exactly one mother and that Betty is John's mother, we may conclude that Mary is not John's mother.

A node that represents a proposition and dominates one or more pattern nodes, may be arcs labeled `forall`, `exists`, or `nexists` pointing to one or more nodes. However, there is the restriction that only one of these types of arcs may emanate from a single node (so that an ordering may be imposed in the use of quantification). Therefore, if more that one quantifier is to be used node embedding is required.

6.3.5 Putting it all together

In this section we discuss the representation in SNePS involving both user-defined case frames and the pre-defined case frames used to represent logical connectives and quantifiers. We will use the case frames described in Section 6.3.3.

We will start by defining rules that specify some of the properties of the membership and subclass relations. The first statement that we represent asserts that if an element is member of a class, then it is also member of a superclass of that class. Using the non-standard connectives, this statement can be expressed as follows:

$$\forall(m, C_1, C_2)\{member(m, C_1), subclass(C_1, C_2)\} \wedge\!\!\rightarrow \{member(m, C_2)\}.$$

The representation of this statement in SNePS is shown in Figure 6.28. If we intend to use the class-subclass relation we should also state that the subclass relation is transitive. This statement can be expressed as:

$$\forall(C_1, C_2, C_3)\{subclass(C_1, C_2), subclass(C_2, C_3)\} \wedge\!\!\rightarrow \{subclass(C_1, C_3)\}.$$

The representation of this statement is SNePS is shown in Figure 6.29.

With these two rules and the network shown in Figure 6.18, SNePS is now capable knowing that John is a mammal.

An alternative way of representing the transitivity of the subclass relation is to explicitly state that the relation is transitive, and to have a rule that states what it means for a relation to be transitive (this representation is shown in Figure 6.30):

$$\begin{array}{ll} \forall(R) & \{Transitive(R)\} \\ & \vee\!\!\rightarrow \\ & \{\forall(C_1, C_2, C_3)\{R(C_1, C_2), R(C_2, C_3)\} \wedge\!\!\rightarrow \{(R(C_1, C_3)\}\} \end{array}$$

Figure 6.28: Transitivity of membership.

Figure 6.29: Transitivity of subclass.

Figure 6.30: Meaning of a transitive relation.

Stating that a relation is transitive, requires the possibility of talking *about* the relation. Since the case frames member/class and subclass/superclass do not allow to explicitly talk about the relation involved, the approach used in Figure 6.30 requires the use of the case frame rel/arg1/.../argn.

Notice that, due to the fact that, in SNePS nodes represent propositions (among other things), we are able to write propositions *about* propositions. In this way, the logic underlying the reasoning of SNePS is a higher-order logic. The advantage of this second approach of representation is that it can be applied to *any* transitive relation. Thus, a SNePS's user, after defining the network structure of Figure 6.30, just has to state that a certain relation is transitive to obtain the desired behavior. On the other hand, this second approach requires additional work from the inference system.

6.3.6 Inference in SNePS

Note: Rule-based and path-based inference. To be added.

6.3.7 Belief revision in SNePS

Note: relationship with TMS systems. To be added.

6.3.8 Conclusions

Note: To be added.

6.4 Other kinds of semantic networks

6.4.1 Semantic primitives

Associated with each representational formalism there is the problem of deciding the vocabulary to be used in the formalism. For example, if we use logic, we have to decide the names of the predicates, functions, and constants to use; if we use SNePS we have to decide the case frames to adopt for our representation.

The research in semantic primitives addresses this problem. We can define a *semantic primitive* as a representational symbol that is used without definition. The idea to use of semantic primitives is based on the atomic elements of Chemistry or a primitive basis in Mathematics (used for an axiomatization). A set of primitives should verify the following conditions:

1. *Finiteness.* A small number of primitives should be defined. The number of primitives should be much smaller than the number of entities that can be described through a combination of primitives.

2. *Comprehension.* The set of primitives must be able to describe all entities of the domain.

3. *Independence.* Primitives should not be definable in terms of other primitives of the same theory (the primitives correspond to a minimal set).

4. *Canonicity.* The analysis of an entity of the domain, in terms of the primitives, should be unique and not ambiguous (existence of a canonical form).

The most famous approaches to semantic primitives were done by Schank and his research group, at the Yale University [Schank 75], [Schank and Abelson 77], [Schank and Riesbeck 81], and by Wilks, in the University of Essex [Wilks 73, 77].

Schank, tried to supply a representation for actions, using a small number of primitives. Each primitive was associated with a certain number of inferences. Schank's work was used to rephrase sentences, translate sentences, produce inferences, and answering questions. The programs based on his formalism translate the input into an underlying representation, based on primitives, conceptual dependency. During the translation phase ambiguities are resolved and the same representation should be generated for sentences with the same meaning.

In order to achieve a canonical form, Schank uses two types of classifications: the classification of concepts into categories (for example, actions, location in space and time) and the grouping of concepts into conceptualizations, each one of them associated with a certain number of cases (in Fillmore's sense).

Concepts are classified in the following categories:

1. *PPs* (Picture Producers), that correspond to physical objects. for example, John, book, cat. Special cases of PPs correspond to forces of Nature, for example, the wind.

2. *ACTs* (Actions), that correspond to the primitive actions that can be performed by an agent. There are eleven primitive actions, classified into four groups, *Physical ACTs* (containing the actions PROPEL, MOVE, INGEST, EXPEL and GRASP), *ACTs characterized by changing states* (containing the actions, PTRANS and ATRANS), *ACTs used as instrument for others* (containing the actions SPEAK and ATTEND), and *mental ACTs* (containing the actions MTRANS and MBUILD). The meaning of each of the primitive actions is the following:

   (a) PROPEL. To apply a force to a physical object;
   (b) MOVE. To move a body part;
   (c) INGEST. To transfer something inside a living being;
   (d) EXPEL. To transfer something from inside a living being;
   (e) GRASP. To physically grasp an object;
   (f) PTRANS. To modify the location of a physical object;
   (g) ATRANS. To modify an abstract relation, for example, change of ownership;
   (h) SPEAK. To produce sounds;
   (i) ATTEND. To capture a stimuli;
   (j) MTRANS. To transfer information;
   (k) MBUILD. To construct information from previous information.

   Each ACT entails a certain number of inferences, formalized in [Schank 75]. For example, "X PTRANSed Y FROM W to Z", gives rise to the following three inferences: (1) Y is located in Z; (2) Y is no longer in W; (3) If (Z = X) or (Z is human and asked for the PTRANS) the Z is happy.

3. *LOCs* (Locations). Every physical action has a location that modifies the conceptualization of the action.

4. *Ts* (Times). Every action has an associated time.

5. *AAs* (Action Aiders). Modify the actions, there are few AAs. For example, to throw has an associated velocity.

$$\text{PP} \Longleftrightarrow \text{ACT}$$

Figure 6.31: PPs can act.

$$\text{PP} \Longleftrightarrow \text{PA}$$

Figure 6.32: PPs may be described by attributes.

6. *PAs* (Picture Aiders). Represent attributes of objects, for example, color and size. These attributes have associated values.

In order to represent knowledge, concepts are grouped into conceptualizations. Each conceptualization has an action, an actor (the agent that executes the action), an object (the thing over which the action applies), a recipient (that receives the result of the action), a direction, and a state. These concepts are nothing more than cases in the sense of Fillmore. Figures 6.31 through 6.35 show some of the ways of combining conceptual categories. The arrows used in these figure, correspond to labeled arcs in the semantic network. The legal ways of combining the concepts correspond to the syntax of the network; each conceptualization has a meaning that is part of the semantics of the network.

Figure 6.36 shows the conceptual dependency representation of "John threw a stone to the cat and killed him".

Schank believed that his system was enough to represent any action. However, soon he recognized that an additional mechanism was needed to represent quantification, commonsense and typical situations. Some of these problems were solved with the introduction of scripts (see Chapter 7).

Schank's theory influenced the field of natural language understanding, thought the introduction of the concept of conceptual primitive that is language-independent. The inferences associated with primitives contributed to the field of language processing based on expectation.

In reality, conceptual dependency is a form of a semantic network. Whereas in traditional semantic networks the primitives are structural and they are used to agglomerate concepts (in SNePS these primitives are defined by the user, with the exception of those needed to represent logical connectives and quantifiers),

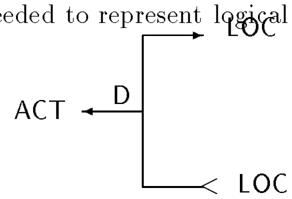

Figure 6.33: ACTs have a direction.

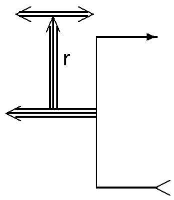

Figure 6.34: Conceptualizations may result from state changes.

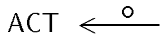

Figure 6.35: ACTs may have objects.

in the conceptual dependency the primitives are both structural and conceptual, and they are used to create more complex concepts.

6.4.2 Conceptual graphs

Note: To be added.

6.5 Evaluation of semantic networks

Semantic networks, as a knowledge representation formalism, present both advantages and disadvantages. Their main advantage is the possibility of explicitly stating associations between concepts. In this way, all the concepts that are related with another concepts are linked to it. In this way we can define structured objects. The main disadvantage of semantic networks is that, if care is not place of the semantics of the representation, the network may generate wrong inferences, since the interpretation of the semantic structure depends on the program that manipulates it.

Note: Additional text to be added.

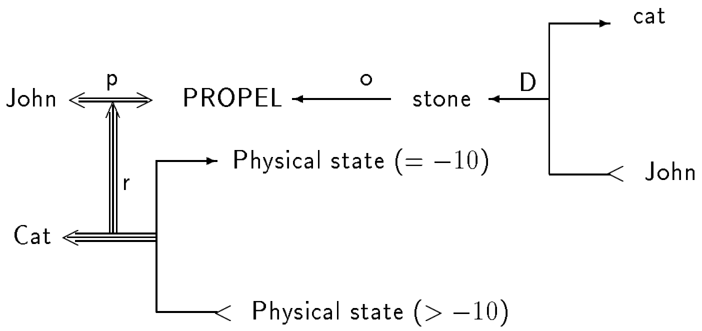

Figure 6.36: Example of representation.

6.6 Summary

In this chapter we discussed the basic principles underlying semantic networks. We briefly described their historical evolution and discusses the options available to the definition of the "semantics" of the semantic network. We presented SNePS, as a case study, and discussed the approaches based on semantical primitives.

Bill:
I will add more text

6.7 Suggested readings

[Lehmann 92]
This book presents a good perspective about the work that is being carried out in semantic networks. It describes the main families of formalisms and some of the problems that the researchers in semantic networks face at present.

[Sowa 91]
It presents a discussion about important aspects that research in semantic networks has to face. It describes, from a formal perspective, semantic networks.

6.8 Exercises

1. Suppose that in SIR, besides the relations defined in Table 6.1, the relations in Table 6.2 were also defined. Sow the network generated by the following interaction with SIR. Explain the answers supplied by SIR.

```
* John owns a pocket-calculator
  I understand
* A pocket-calculator is a computing-device
  I understand
* Does John own a computing-device
  Yes
* Every engineering-student owns a computing-device
  I understand
* Peter is a tech-man
  I understand
* A tech-man is an engineering-student
  I understand
* Does Peter own a computing-device
  Yes
* Does an engineering-student own the pocket-calculator
  Insufficient information
```

Relation	*Representation of relation*	*Representation of inverse relation*	*Meaning (corresponding sentence)*
$own(x, y)$	$x \xrightarrow{\text{owned}} y$	$y \xrightarrow{\text{possess}} x$	y owns an x
$owng(x, y)$	$x \xrightarrow{\text{owned-by-each}} y$	$y \xrightarrow{\text{possess-by-each}} x$	every y owns a x
$part(x, y)$	$x \xrightarrow{\text{superpart}} y$	$y \xrightarrow{\text{subpart}} x$	An x is a part of y
$partg(x, y)$	$x \xrightarrow{\text{superpart-of-each}} y$	$y \xrightarrow{\text{subpart-of-each}} x$	An x is a part of a y

Table 6.2: Relations defined in SIR.

```
* Alfred is a tech-man
  I understand
* Does an engineering-student own the pocket-calculator
  Yes
```

2. Suppose that in SIR, besides the relations defined in Table 6.1, the relations in Table 6.2 were also defined. Show the network generated by the following interaction with SIR. Explain the answers supplied by SIR.

```
* A nose is a part of a person
  I understand
* A nostril is a part of a nose
  I understand
* A professor is a teacher
  I understand
* A teacher is a person
  I understand
* Is a nostril part of a professor
  Yes
* A person is a living-creature
  I understand
* Is a nostril part of a living-creature
  Sometimes
```

3. What does ${}_2\mathbb{X}_1^1\{\alpha, \beta\}$ represent?

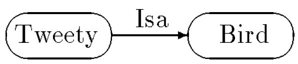

Figure 6.37: Impossible representation in SNePS.

4. What does ${}_n\mathbb{X}_0^0\ \{\alpha_1, \ldots, \alpha_n\}$ represent?

5. Using SNePS, represent, in a graphical form, the following statements: Anna knows that John bought a dog. The dog is a German Shepherd. John paid 300 dollars for the dog. Anna also knows that Stu sold his dog. Anna believes that John bought Stu's dog.

6. Using SNePS, represent, in a graphical form, the following statements: John and Mary are lawyers. Charles believes that all the lawyers are rich. Charles knows John's address but he does not know that he and Mary live together.

7. Using SNePS, represent, in a graphical form, the following statements: John is a professor and earns 50,000 dollars a year. His friends believe that he earns much more. They do not know that he earns as much as a police officer.

8. SNePS does not allow the representation shown in Figure 6.37 for the statement "Tweety is a bird". Explain the reason for this decision.

9. Explain the difference between the representations of figures 6.21 and 6.37. Why is the representation of Figure 6.21 legal?

10. Represent in SNePS the concept of a reflexive relation.

11. Represent in SNePS the concept of a symmetric relation.

12. Represent in SNePS the concept of an equivalence relation.

Chapter 7

Frames

1. Basic concepts
2. Reasoning with frames
3. KEE: A case study
4. Evaluation of frames
5. Summary
6. Suggested readings
7. Exercises

In this chapter we discuss frames as a knowledge representation formalism. We present a case study with a frame system: KEE. We evaluate frames as a knowledge representation system.

> *"When I have heard some slight indication of the course of events, I am able to guide myself by the thousand of other similar cases which occur to my mind"*
> Connan Doyle, *The Red-headed Leage* (Sherlock Holmes)

Frames, as a knowledge representation formalism in AI, appear in the mid 1970s. They were proposed by Marvin Minsky [Minsky 75], who suggested that our knowledge was organized in "chunks", called frames, that correspond to stereotyped situations. Related frames are linked together into frame systems. An similar idea had been proposed in psychology by [Bartlett 32].

A frame, as originally proposed by Minsky, corresponds to a chunk of knowledge that was activated in a specific situation and that was used to supply interpretations and to make predictions about that situation. Each frame is, thus, associated with a typical situation. For example, when we enter a room that corresponds to the office of a computer science scientist, we activate a frame that let us predict what we expect to find:

- Walls, floor, and ceiling, and, eventually, one or more windows. These predictions are made because the frame corresponds to a room. We would be very surprised, if, when opening the door, we would get into a beach or a room with no ceiling;
- A desk, chairs, and one of more bookshelves. These predictions are made because we are entering the room corresponding to an office. Again, we would be surprised in face of a set of sofas and a table with a TV. The existence of these later objects would not surprise us if we would be entering a room, whose type was unknown to us;
- A computer and computer science books. These predictions are made because we are entering the office of a computer scientist. We would be surprised to find medical instruments instead (these would be associated with a medical doctor).

In a similar way, we would make predictions about what to expect the walls, floor or ceiling to look like or to contain.

Minsky argues that each of the concepts that we described (room, office, office of a computer scientist, wall, floor, ceiling, and so on) corresponds to a frame, that, in turn, may refer other frames (for example, the frame corresponding to a room refers to the frames corresponding to wall, ceiling and floor).

The organization of knowledge using frames reduces the computational effort to process knowledge, because, once a frame is selected, all the relevant knowledge is associated to the frame itself. A frame, thus, is similar to an object on the abstract data types theory or in an object-oriented system, object that agglomerates all operations that manipulate it.

Figure 7.1: Classification of "animal".

Minsky proposed that frames would describe several viewing perspectives of an object, according to the position of the observer. He also proposed a language through which frames could be specified. Minsky's paper is very vague, raising more questions than answers, but was extremely influential in the area of knowledge representation.

In summary, frames are a structured knowledge representation formalism to represent stereotyped situations. Frames correspond to chunks of knowledge describing conceptual entities with associated descriptions. Minsky's paper presented applications in scene recognition and natural language understanding and relied on the notion of frame matching.

Frames are hierarchically organized and have a property inheritance mechanism as the basic inference procedure. Intuitively, a frame corresponds to the abstraction of a class whose characteristics are represented as attribute-value pairs.

Frames were the first knowledge representation formalism to combine procedural and declarative knowledge.

7.1 Basic concepts

There are several languages to represent knowledge using frames, for example, KRL [Bobrow and Winograd 77], FRL [Roberts and Goldstein 77], UNITS [Stefik 79], and KEE [Fikes and Kehler 85].

These languages represent knowledge, using the concept of a frame, that is typically called a *unit*. Each unit has a certain number of attributes, called *slots*, that describe the particular aspects of the unit. Units are hierarchically organized. There is a clear distinction between units that represent classes and units that represent instances. Figure 7.1 shows part of a hierarchy corresponding information pertaining animals. The solid line connections correspond to class/subclass relations and the dashed line connections correspond to class/member relations.

Each class of instance represented in this hierarchy corresponds to a frame. Each frame has a certain number of attributes. Attributes my be characterized by *facets* that define aspects of the attributes. For example, facets my characterize:

- The value of the attribute;
- The class of values that the attribute may have;
- The cardinality (either minimal, maximal, or both) of the values associated with the attribute. For example, if the maximal cardinality of an attribute is 1, it means that the attribute may have at most one value;
- The default value, that is the value of the attribute in case no value is explicitly stated;
- The type of inheritance associated with the attribute, that is the way values of the attribute are propagated to the frames bellow it in the hierarchy;
- Procedures associated with the attribute. There are two kinds of procedures that may be associated with an attribute, demons and servants.
  - *Demons* are procedures that are automatically executed when the value of the attribute changes;
  - *Servants* are procedures that are executed when the value of the attribute is inspected or queried.

7.2 Reasoning with frames

7.2.1 Property inheritance

Property inheritance is a form of inference, where the values of the attributes of a class are propagated to all of its subclasses and members. Property inheritance corresponds to default reasoning.

7.2.2 Reasoning based on rules

7.3 O KEE: A case study

7.4 Evaluation of frames

7.5 Summary

7.6 Suggested readings

[Minsky 75]
The original paper on frames.

[Lehman 92]
It presents a good discussion on the disadvantages of frames.

7.7 Exercises

Chapter 8

Hybrid Systems

1. KL-ONE
2. Terminological formalism
3. Assertional formalism
4. Operations with the formalisms
5. Summary
6. Suggested readings
7. Exercises

In this chapter, we discuss representation formalisms that use more than one of the traditional types of representation. We present the motivations for their creation and describe in detail one formalism of this kind, KL-ONE.

> 'Not very nice *alone*,' he interrupted, quite eagerly: 'but you've no idea what a difference it makes, mixing it with other things'
> Lewis Carroll, *Through the Looking Glass*

A system for knowledge representation is said to be a *hybrid system* if it incorporates more than one of the traditional representation formalisms. The idea behind the development of hybrid systems is to obtain the advantages of different representation systems through the joint use of some of their features. Hybrid systems evolved from the work of [Brachman 78], who developed Structural Inheritance Networks, a formalism in which there is a clear distinction between *structural links*, links that relate the components of a structured concept, and *assertional links*, links that represent assertions about concepts. The need to distinguish between these two kinds of links had been pointed out by [Woods 75]. A hybrid system is composed by, at least, two sub-systems:

1. A sub-system to represent concepts in terms of other concepts. This sub-system defines the terminology of concepts and is called the *terminological representation system* or *TBox*.

2. A sub-system to make assertions about the world. This sub-system defines the objects existing in the world and links them to the existing concepts. It is called the *assertional representation system* or *ABox*.

The development of the first hybrid system, was partially motivated by the arbitrary way in which frames deal with the representation of knowledge. When we represent knowledge in frames, we define a partial order on frames (which may either represent classes or instances). Each frame is characterized by a certain number of attributes that are propagated (together with their values) to the frames that are located lower down in the hierarchy. In a frame system, the value of a given attribute may be changed arbitrarily at any point. This approach presents two problems:

1. The partial order relations between frames are established by the person who builds the frame system, and there are no external criteria to define when these relations are correct. This makes the semantics of the links between frames purely operational; it defines the way attributes must be propagated without any external criteria to define when relations should be established.

2. The fact that an inherited attribute can be arbitrarily changed prevents keeping the properties (or characteristics) that one wants to assign to the classes that frames represent. For example, let us suppose that we want to define a frame "three legged dog", in which the attribute number-of-legs is equal to 3, and let us suppose, furthermore, that Fido is an instance of

the class represented by this frame. There is nothing in the frame system that prevents us from changing the number of legs of Fido to 4.

Based on these criticisms, [Brachman 78] proposes Structural Inheritance Networks, a kind of semantic network that is intended to distinguish between structural relations and assertional relations. In Structural Inheritance Networks, knowledge is represented as classes, sub-classes, and instances. Each class (representing a concept) is associated with the characteristics that distinguish it from its superclasses.

8.1 KL-ONE

Structural Inheritance Networks led to the development of KL-ONE [Brachman and Schmolze 85], the first system to deal with knowledge representation at the epistemological level [Brachman 79] (see Chapter 9). The primitives supplied by KL-ONE deal with basic relations between concepts, such as the definition that a given concept has a certain attribute, that a concepts satisfies a certain relation, that a concept is defined by a set of properties, or that a concept is more specific than another one. Another characteristic of KL-ONE is the concern for the non-ambiguous definition of the meaning (semantics) of the representation.

KL-ONE represents knowledge based on a taxonomy defined among concepts, but the semantics of the taxonomy is defined independently of the algorithms that manipulate the data structures that correspond to the taxonomy. The semantics defines when a concept is a generalization or a specialization of another concept. Concepts in KL-ONE are divided into generic and individual concepts:

1. *Generic concepts* represent classes of individuals. Generic concepts are, in turn, divided into defined concepts and primitive concepts:

   (a) *Defined concepts* are concepts whose meaning is fully defined by its description. These concepts are introduced specifying the necessary and sufficient conditions for a given entity to be classified as belonging to the class corresponding to the concept. For example, "Woman" may be defined as Human and Female

   (b) *Primitive concepts*[1] are concepts whose meaning cannot be defined in an exhaustive way; in other words, they are considered as associated with incomplete definitions. These concepts are introduced through the specification of the necessary conditions for a given entity to be considered as belonging to the class that the concept represents. These specifications constrain the class of the entities that satisfy the concept but do not fully define it. For example, "Dog" may be defined as Mammal that barks.

[1] Also known as natural concepts or natural kinds.

Figure 8.1: Taxonomy of Concepts.

2. *Individual concepts* represent individuals. Individual concepts may be considered as representing classes with a single element. These concepts have no assertional impact and are used to describe potential individuals. In their description, there is no claim about their existence.

In order to define a concept in KL-ONE, three kinds of specifications are used:

1. *Taxonomic specifications*: These relate a concept with other concepts that are either more general or more specific than the given concept. In Figure 8.1, we show a hierarchy that contains generic concepts and individual concepts (this figure was adapted from [Nebel 90], Figures 1.2 and 3.1). Concepts are graphically represented by labeled nodes, and the taxonomic relations between concepts are represented by directed unlabeled arcs that point from more specific concepts to more general concepts. Individual concepts are represented by shaded nodes. An important aspect of this taxonomy is that the meaning of a concept depends *only* on the meaning of the concepts that are above it in the taxonomy and on the information associated with that particular concept.

Figure 8.2: Roles relating concepts.

2. *Role specifications*: These correspond to potential relation between the members of the class corresponding to the concept and other concepts. Roles try to capture a common generalization of the notion of attribute, part, constituent, aspect, parameter, argument, grammatical case, and so on. Roles, together with structural descriptions, differentiate concepts and specify the ways a concept differs form the intersection of its immediate super-concepts in the taxonomy. Among other things, a role may be associated with restrictions on the possible values it may assume and the number of possible associations. Roles are graphically represented by circles with an inscribed square, and the restrictions to the values a role may assume are represented by an arc from the role to the possible concepts that may be values of the role. Figure 8.2 represents, among other things, that a group has a role called "elements", that the elements of a group are Humans, and that the number of elements is greater than or equal to 2. Roles may be hierarchically organized through two different relations:

   (a) A role may be more constrained version of another role. In Figure 8.2, the role "element" associated with the concept "Small group" has an additional restriction (that the number of elements be less than 5) on the role "element" associated with the concept "Group". In this case, the links between roles are labeled with the label "restr".

   (b) A role may be a differentiated version of another, more general, role. For example, "arm" and "leg" are differentiated versions of the role "member". In this case, links between roles are labeled with the label "diff". Figure 8.3 shows the roles that correspond to differentiation of "member".

3. *Structural conditions*: These define relationships between roles and constrain the potential values that a role may assume. The roles used in the definition of a concept are, in general, not independent of one another. These interdependencies are specified through structural relations.

Figure 8.3: Roles corresponding to the differentiation of member.

Figure 8.4: Example of structural relations.

In general, structural relations are represented as parameterized descriptions of the individual relations that have to be satisfied by instances of the concept. These relations are called parameterized entities, because there is an instantiation of the relation that is different for each instantiation of the concept. Graphically, these relations are represented by a double circle (enclosing the name of the relation) linked to a diamond-shaped lozenge, which is, in turn, linked to the concept. The label inside a double circle corresponds to a concept represented in the system and specifies a correspondence between the roles of that concept and the roles of the concept being described. This correspondence is done, graphically, through the indication of the name of the role (of the concept corresponding to the double circle) associated with a square. In Figure 8.4, we show the description of the concept "Arch". An arch has two roles, "lintel" (that has only one object that fills its role) and "upright" (for which there are two objects that fill their roles); both of these roles can be filled by Blocks. Given a particular arch, its uprights must satisfy the "NotTouch" relation, and the uprights must satisfy the relation "Support" in relation to the lintel.

There are two kinds of structural conditions:

(a) General conditions, or it structural relations, which enable the specification of the way roles and concepts interrelate with one another. The interdependencies between the components of an arc are examples of structural relations.

(b) Particular conditions, or it role value maps, which enable the specification of subset and equality relations between the possible fillers of a role.

In KL-ONE, roles and structural conditions associated with a concept are also applicable to all specializations of that concept. In this respect, concept inherit roles and structural conditions of the most general to which they are connected. This kind of inheritance is different from the inheritance found in frame systems, since the inherited attribute corresponds to a defining characteristic of a concept. More specific concepts may add restrictions to these characteristics but may not modify them.

KL-ONE had a profound influence on the field of knowledge representation and gave rise to about 20 knowledge representation systems [Woods and Schmolze 92].

The great success obtained by KL-ONE stems from the fact that it did not try to deal with all the necessary aspects in a knowledge representation scheme, but concentrated on the way of defining concepts. In this respect, KL-ONE is similar to the system developed by [Quillian 68]. For this reason, KL-ONE does not address certain aspects of a representation that clash with the notion of "definition", in particular the notion of default value and overriding inherited values, in other words, aspects that enable the suppression of parts of the description used to define a concept.

The stress placed on the definition of concepts is the main idea behind KL-ONE. The most important consequence of this aspect is the possibility of reasoning about relation between concepts, based on their description. This aspect of reasoning is called *terminological reasoning* and gave rise to a system that is able to automatically classify concepts in a hierarchy [Schmolze and Lipkis 83].

In Section 8.2 and 8.3, we present the definitions of languages for the terminological subsystem and the assertional subsystem of a hybrid formalism, respectivelly. In our presentation, we try show a large number of features of these formalisms, and we go beyond any existing system. In general, when the terminological language has the expressive power corresponding to what we present, the assertional language has a weaker expressive power, and viceversa.

8.2 Terminological formalism

In this section, we describe a language, called $\mathcal{TF}$, that corresponds to a superset of most languages of the KL-ONE family (our description is adapted from [Woods and Schmolze 92]). A specification in $\mathcal{TF}$ consists of a set of concepts, a set of roles, a set of assertions about the non-intersection between concepts and roles, a set of individual assertions about concepts, and a terminology that associates names with specifications of concepts and roles.

8.2.1 Syntax of $\mathcal{TF}$

In this section, we describe, using a BNF-like grammar, the syntax of the language $\mathcal{TF}$. A property that cannot be expressed in a BNF grammar and that is assumed in $\mathcal{TF}$ is that a term is only defined once.

A termonology consistst of one or more specifications
<terminology> ::= <specification>$^+$

<specification> ::= <term introduction> |
<concept> |
<role> |
<individual specification> |
<disjoint specification>

A term introduction can either correspond to the introduction of a concept of to the introduction of a role (these can, in turn, either be primitive or defined). A term is introduced by specifying its name (either <concept name> or <role name>), followed by the description of the term.

<term introduction> ::= <(**cprim** <concept name> <concept>) |
<(**cdef** <concept name> <concept>) |
<(**rprim** <role name> <role>) |
<(**rdef** <role name> <role>)

A description of a concept may correspond to the most general concept (**Thing**), to the composition of other concepts (conjunction, disjunction, or negation), or to the restriction on roles associated with it.

<concept> ::= **Thing** |
<concept name> |
(**and** <concept>$^+$) |
(**or** <concept>$^+$) |
(**not** <concept>) |
(**all** <role> <concept>) |
(**some** <role>) |
(**c-some** <role> <concept>) |
(**atleast** <minimum> <role>) |
(**c-atleast** <minimum><role> <concept>) |
(**atmost** <maximum><role>) |
(**c-atmost** <maximum><role> <concept>) |
(**rvm** <role> <crole>) |
(**rvm** <role> <crole>) |
(**sd** <concept> (<role> <crole>)$^+$)

The description of a role may either correspond to the most general role (`MGR`), to the composition of other roles (trough logical operators), or to the restriction on other roles.

<role>::= `MGR` |
<role name> |
(`and-role` <role>$^+$) |
(`or-role` <role>$^+$) |
(`not-role` <role>) |
(`restr` <role> <concept>) |
(`domain` <concept>) |
(`range` <concept>) |
(`self`) |
(`inv` <role>) |
(`chain` <role> <role>$^+$)

<individual specification> ::= (`individual`<concept>)
<disjoint specification> ::= (`disjoint`<concept> <concept>$^+$) |
`disjoint-role`<role> <role>$^+$)

<minimum> ::= "non-negative integer"

<maximum> ::= "non-negative integer"

<concept name> ::= "string"

<role name> ::= "string"

The `cprim` operator allows the introduction of a new term corresponding to a primitive concept and whose meaning is defined by the second argument of this operator. For example, (`cprim Human animal`), specifies that the concept `Human` is an `Animal`. Since `Human` is a primitive concept, we do not have enough criteria to determine when an `Animal` is a `Human`. Similarly, the operator `cdef` allows the introduction of a new term corresponding to a defined concept. The operators `rprim and rdef` have a similar behavior but they deal with the definition of roles.

give examples of both

The operators associated to the definition of the non-terminal symbol "concept" allow the specification of concepts. The symbol "`Thing`" corresponds to the most general primitive concept. The operators `and, or`, and `not` are used to specify, respectively, that a concept simultaneously satisfies the conditions imposed on two other concepts, the condition imposed on another concepts, or represents the complement of another concept. For example, (`and Female Human`) represents the concept of being simultaneously `Female` and `Human` can be used to define the concept.

`(cdef Woman (and Female Human))`.

The operator (`all` <role> <concept>) constrains the possible values that a role may have. For example, (`all element Woman`) states that the role `element` can only be filled by members of the concept `Woman`, in other words, that all the elements have to be women. This restriction may be used in the definition of the concept:

```
(cprim FeminineGroup (all element Woman)).
```

The operator (`some` <role>) states that the role has at least one value and corresponds to a way of defining an attribute of a concept. For example (`some children`) introduces the attribute `children` in a concept and can be used in the definition that a Parent is someone who has children:

```
(cprim Parent (some children)).
```

The operator (`c-some` <role> <concept>) is somewhat similar to the previous one, but restricts the values that the role may have. For example, (`c-some children Doctor`) requires that the attribute `children` has as value at least one entity that satisfies the concept `Doctor`, and can be used in the definition:

```
(cprim ProudParent (c-some children Doctor)).
```

The operators (`atleast`<minimum> <role>) and (`c-atleast` <minimum> <role> <concept>) allow the specification of the minimum number of elements that can be associated with a role. The difference between them is that `c-atleast` allows the specification of the class of values that the role may have. The operators `atmost` and `c-atmost` are similar to `atleast` and `c-atleast`, but they specify maximum number of elements that can be associated with the role.

The operators `rvm` and `rvm=` specify, respectively, subset and equality relations between sets that are associated with roles. For example, `rvm=` can be used to define that the GrandParents of a Person are exactly the Parents of the Parents of the Person. **give example showing how**

The operator (`sd` <concept> (<role> <role>)$^+$) specifies a structural description, by means of an association between the roles of the concept being defined and the roles of the concept in the structural description. For example, with respect to the concept "Arch" presented in Figure 8.4, we could write:

```
(sd Support (supporter lintel)(supported upright)).
```

The operators associated with the definition of the non-terminal symbol `MGR` enable the specification of roles. Symbol `MGR` corresponds to the most general role.

The operators `and-role, or-role`, and `not-role` enable the specification of conjunction, disjunction, and negations between roles, respectively.

The operator (`restr` <role> <concept>) allows the imposition of constraints on the values associated with the role of a given concept. You might think that since we have the operators `all` and `c-some`, this operator would be useless. However, we should keep in mind that these two operators are used in the definition of a concept, and they impose restrictions on the role of *that* concept, whereas **restr** is used in the definition of a role; thus, the imposed restriction is applicable to *all* uses of that role, no matter in which concept.

The operators (`domain` <concept>) and (`range` <concept>) are used to define, respectively, the concepts that belong to the domain and range of a role. The operator **self** corresponds to the concept itself. The operator **inv** corresponds to the converse operation. The operator (`chain` <role> <role>$^+$) enables the specification of an application of a sequence of roles.

The operators associated with the definition of the non-terminal symbol "disjoint specification" enable the specification that two or more concepts or roles are disjoint.

As a first example of a terminological specification, the following specifications define the concepts represented in Figures 8.1 and 8.2:

```
(cprim Intangible Thing)
(cprim Tangible Thing)
(cprim Set Intangible)
(cprim Group (and Set
                (all element Human)
                (atleast 2 element)))
(cdef SmallGroup (and Group
                     (atmost 5 element))
(cdef ModernGroup (and Group
                      (atmost 4 element)
                      (atleast 1 leader)
                      (atmost 1 leader)
                      (all leader Woman)))
(cprim Vegetable Tangible)
(cprim Animal Tangible)
(cprim Mineral Tangible)
(cprim  Male Animal)
(cprim Female Animal)
(cprim Human Animal)
(cdef Woman (and (Female Animal))

(rprim element MGR)
(rprim leader element)
```

As a second example, the concepts represented in Figure 8.4 can be described by the following specification:

```
(cdef Arc (and (atleast 1 lintel)
               (atmost 1 lintel)
               (all lintel Block)
               (atleast upright 2)
               (atmost upright 2)
               (all upright Block)
               (sd NotTouch (objects upright))
```

```
(sd Support (supported lintel)
            (supporter upright))))
```

8.2.2 Semantics of $\mathcal{TF}$

The idea behind the definition of a semantics for a terminological formalism is the existence of a certain number of entities that we are interested in (the universe of discourse, $\mathcal{D}$). Each concept represents a certain number of these entities (a subset of $\mathcal{D}$), and each role represents a certain number of pairs of these entities (a subset of $\mathcal{D} \times \mathcal{D}$). The sets associated with concepts and roles are called their *extensions*.

Using the relations (defined by the terminology) between concepts and roles, we can analyze the required relations between the entities that we are interested in. For example, the notion that a ModernGroup must be a Group must be reflected in the in the semantics through a subset relationship between the extension of ModernGroup and the extension of Group, independently of the extensions that really exist in the outside word.

In order to formalize these concepts, we introduce the notion of an extension function ($\Re$), a function that maps terms into extensions.

Let $\mathcal{D}$ be a set of entities, let $\mathcal{C}$ be a set of concepts and let $\mathcal{P}$ be a set of roles. Let $\Re$ be a function such that $\Re : \mathcal{C} \Rightarrow 2^{\mathcal{D}}$ (each concept is associated with a subset of $\mathcal{D}$), and $\Re : \mathcal{P} \Rightarrow 2^{\mathcal{D}^2}$ (each role is associated with a subset of ordered pairs of $\mathcal{D}$. We say that $\Re$ is an *extension function* if and only if it satisfies the following conditions on concepts and roles:

1. Concepts:
   - $\Re[\texttt{Thing}] = \mathcal{D}$
   - $\Re[(\texttt{and}\ c_1 \ldots c_n)] = \cap_{i=1}^{n} \Re[c_i]$
   - $\Re[(\texttt{or}\ c_1 \ldots c_n)] = \cup_{i=1}^{n} \Re[c_i]$
   - $\Re[(\texttt{not}\ c)] = \mathcal{D} - \Re[c]$
   - $\Re[(\texttt{all}\ p\ c)] = \{x \in \mathcal{D} : \forall y(<x, y> \in \Re[p] \rightarrow y \in \Re[c])\}$
   - $\Re[(\texttt{some}\ p)] = \{x \in \mathcal{D} : \exists y :<x, y>\in \Re[p]\}$
   - $\Re[(\texttt{c-some}\ p\ c)] = \{x \in \mathcal{D} : \exists y(<x, y>\in \Re[p] \land y \in \Re[c])\}$
   - $\Re[(\texttt{atleast}\ n\ p)] = \{x \in \mathcal{D} : \exists_n y \text{ (distinct) } (<x, y>\in \Re[p])\}$
   - $\Re[(\texttt{c-atleast}\ n\ p\ c)] =$
     $\{x \in \mathcal{D} : \exists_n y \text{ (distinct) } (<x, y>\in \Re[p] : y \in \Re[c])\}$
   - $\Re[(\texttt{atmost}\ n\ p)] =$
     $\{x \in \mathcal{D} : \neg(\exists_{n+1} y \text{ (distinct) } <x, y>\in \Re[p])\}$
   - $\Re[(\texttt{c-atmost}\ n\ p\ c)] =$
     $\{x \in \mathcal{D} : \neg(\exists_{n+1} y \text{ (distinct) } (<x, y>\in \Re[p] : y \in \Re[c]))\}$
   - $\Re[(\texttt{rvm}\ p_1\ p_2)] =$
     $\{x \in \mathcal{D} : \forall y<x, y>\in \Re[p_1] \rightarrow<x, y>\in \Re[p_2]\}$

- $\Re[(\texttt{rvm=}\ p_1\ p_2)] =$
  $\{x \in \mathcal{D} : \forall y {<}x, y{>} \in \Re[p_1] \leftrightarrow {<}x, y{>} \in \Re[p_2]\}$
- $\Re[(\texttt{sd}\ s\ (p_1^c\ p_1^s) \ldots (p_n^c\ p_n^s))] =$
  $\{x \in \mathcal{D} :$
  $\exists y(y \in \Re[s] \wedge$
  $\forall_{z_1}, \ldots, z_n\ (({<}x, z_1{>} \in \Re[p_1^c] \leftrightarrow {<}y, z_1{>} \in \Re[p_1^s]$
  $\wedge \ldots \wedge$
  $({<}x, z_n{>} \in \Re[p_n^c \leftrightarrow {<}x, z_n{>} \in \Re[p_n^s]))))\}$

2. Roles:

- $\Re[\texttt{MGR}] = \mathcal{D}^2$
- $\Re[(\texttt{and-role}\ p_1 \ldots p_n)] = \cap_{i=1}^n \Re[p_i]$
- $\Re[(\texttt{or-role}\ p_1 \ldots p_n)] = \cup_{i=1}^n \Re[p_i]$
- $\Re[(\texttt{not-role}\ p)] = \mathcal{D}^2 \Re[p]$
- $\Re[(\texttt{restr}\ p\ c)] = \{{<}x, y{>} \in \Re[p] : y \in \Re[c]\}$
- $\Re[(\texttt{domain})\ c)] = \{{<}x, y{>} \in \mathcal{D}^2 : x \in \Re[c]\}$
  item $\Re[(\texttt{range})\ c)] = \{{<}x, y{>} \in \mathcal{D}^2 : y \in \Re[c]\}$
- $\Re[(\texttt{self})] = \{{<}x, y{>} \in \mathcal{D}^2\}$
- $\Re[(\texttt{inv}\ p)] = \{{<}x, y{>} \in \mathcal{D}^2 : {<}x, y{>} \in \Re[p]\}$
- $\Re[(\texttt{chain}\ p_1 \ldots p_n)] =$
  $\{{<}x, y{>} \in \mathcal{D}^2 :$
  $\exists z_1, \ldots, z_{n-1}({<}x, z_1{>} \in \Re[p_1] \wedge {<}z_1, z_2{>} \in \Re[p_2]$
  $\wedge \ldots \wedge$
  ${<}z_{n-1}, y{>} \in \Re[p_n])\}$

An extension function $\Re$, defined over a domain $\mathcal{D}$, that respects all the term introductions, individual specifications, and disjoint specifications of a terminology is said to be a *semantic structure* of that terminology.

Let $\mathcal{T}$ be a terminology defined using the BNF grammar of Section 8.2.1. Let $\mathcal{D}$ be a universe of discourse, and let $\Re$ be an extension function. We say that $\Re$ is an *extension function to* $\mathcal{T}$ and conditions are satisfied:

1. Term introduction

- $\Re[n_C] \subseteq \Re[c]$ if $(\texttt{cprim}\ n_c\ c) \in \mathcal{T}$
- $\Re[n_C] = \Re[c]$ if $(\texttt{cdef}\ n_c\ c) \in \mathcal{T}$
- $\Re[n_P] \subseteq \Re[p]$ if $(\texttt{cprim}\ n_P\ p) \in \mathcal{T}$
- $\Re[n_P] \subseteq \Re[p]$ if $(\texttt{cdef}\ n_P\ p) \in \mathcal{T}$

2. Assertions about individuality: [2].

   - $\| c \| \leq 1$ if (`individual`c) $\in \mathcal{T}$

3. Assertions about disjunctions:

   - $\Re[c] \cap \Re[c'] = \{\}$ if (`disjoint` $c_1 \ldots c_n$) $\in T$ e $\exists\, i, j$ with $1 \leq i, j \leq n$, $i \neq j$, $c = c_i$ and $c' = c_j$.
   - $\Re[p] \cap \Re[p'] = \{\}$ if (`disjoint-role` $p_1 \ldots p_n$) $\in T$ e $\exists\, i, j$ with $1 \leq i$, $j \leq n$, $i \neq j$, $p = p_i$ and $p' = p_j$.

8.3 Assertional formalism

In this section, we describe a language, called $\mathcal{AF}$, that corresponds to one of the assertional languages used in hybrid systems. The language we present adapted from [Nebel 90].

8.3.1 Syntax of $\mathcal{AF}$

The syntax of AF is defined by the following BNF grammar:

<world description> ::= <description>$^+$

<description> ::= <description of object>|
<description of relation>

<description of object> ::= (<name of concept> <object>)

<description of relation> ::=
(<name of role> <object> <object>) |
(<name of role> <object> (`atleast` <integer>) |
(<name of role> <object> (`atmost` <integer>)

<name of concept> ::= "string"

<name of role> ::= "string"

<object> ::= "string"

<minimum>::= "integer"

<maximum>::= "integer"

A description of the world is a set of assertions about objects and their interrelations. A description of object is the assertion that a given object belongs

[2] $\| c \|$ represents the cardinality of the set c

to an extension of an atomic concept, in other words, that the object is an instance of the concept. With a description of relation, we can assert that an object has certain values as fillers for its roles or that it has a certain number of values as fillers for these roles without having to specify those values. As an example, let us define the following assertions concerning the concepts represented in Figure 8.2; the result of the assertions, together with the links for the terminological representation are shown in the top part of Figure 8.5:

```
(ModernGroup G05)
(Human John)
(Woman Tracy)
(element G05 Mary)
(element G05 Peter)
(element G05 Tracy)
(leader G05 Mary)
(element G05 (atmost 3))
```

8.3.2 Semantics of $\mathcal{AF}$

Let $\mathcal{D}$, be a universe of discourse, $\mathcal{E}$ a set of entities, $\mathcal{C}$ a set of concepts, and $\mathcal{P}$ a set of roles. Let $\mathcal{I}$ be an interpretation function such that $\mathcal{I} : \mathcal{E} \Rightarrow \mathcal{D}$ (each entity is associated with an object of the universe of discourse), $\mathcal{I} : \mathcal{C} \Rightarrow 2^{\mathcal{D}}$ (each role is associated with a subset of ordered pairs of $\mathcal{D}$).

The pair $\mathcal{M} = [\mathcal{D}, \mathcal{I}]$, called an *interpretation*, satisfies the description δ if and only if:[3]

$$\models_{\mathcal{M}} (c\ o) \text{ iff } I[o] \in I[c]$$
$$\models_{\mathcal{M}} (p\ o_1\ o_2) \text{ iff } \langle I[o_1], I[o_2]\rangle \in I[p]$$
$$\models_{\mathcal{M}} (p\ o\ (\texttt{atleast}n)) \text{ iff } \|\ \{\langle\ I[o], x\rangle \in I[p]\}\ \| \geq n$$
$$\models_{\mathcal{M}} (p\ o\ (\texttt{atmost}n)) \text{ iff } \|\ \{\langle I[o], x\rangle \in I[p]\}\ \| \leq n$$

An interpretation is a model of a description of the world, Δ, written as $\models_{\mathcal{M}} \Delta$, if and only if it satisfies all descriptions of Δ. A joint interpretation of a terminology and a description of the word requires a relationship between the models of the description of the world with the semantic structure of the terminology. Whereas the semantic structure of the terminology can be considered as a specification of the *possible* relations introduced by the way we organize our knowledge, world descriptions correspond to partial descriptions of the way the world in fact is, and must respect the relations required by the terminology.

A model $[\mathcal{D}, I]$ of a description, Δ, of the world *respects the terminology* $\mathcal{T}$ if and only if there is a semantic structure $[\mathcal{D}, \Re]$ of $\mathcal{T}$ such that the restrictions of the functions I and $\Re$ with respect to the individual concepts and roles defined in Δ are identical.

[3] In this description, c represents a concept, o, o_1 e o_2 represent objects, and n is an integer number.

Figure 8.5: Connection between t erminological and assertional representations.

8.4 Operations with the formalisms

8.4.1 Operations in the terminological system

In general, the terminological system of a hybrid system allows the following operations:

1. *Subsumption.* We say that a term subsumes another if and only if the set of entities that it represents is contained in the set of entities represented by the other. In other words, A is subsumed by B if and only if for all the semantic structures $[\mathcal{D}, \Re]$ we have $\Re[A] \subseteq \Re[B]$. *Classification* is the process of determining subsumption relations between sets of terms.

2. *Inheritance.* Inheritance is the process through which we determine the conditions that should be applied to a given term because it is subsumed by other terms.

3. *Completion.* Completion is the process through we identify and record all the conditions that are applicable to a given term.

4. *Coherence.* A term is said to be coherent if and only if there is a semantic structure for which the term is not associated with the empty set. In other words, A is coherent if there is a semantic structure $[\mathcal{D}, \Re]$ in which $\Re[A] \neq \{\}$.

All the system in the KL-ONE family compute inheritance and subsumption. Most of them also perform classification. One of the research areas in hybrid systems corresponds to the study of the balance between the expressive power of the representation scheme and the soundness, completeness, and computing tractability of its algorithms.

8.4.2 Operations in the assertional system

In the assertional system, we query the knowledge of *RepA* about the properties of the represented entities. This interaction may correspond to finding out all entities that can be described belonging to a given concept. In some systems, for example, LOOM [MacGregor 91], in which the assertional language corresponds to first-order logic, the system is associated with a belief-revision system that detects contradictions and supplies operations to remove assertions. For example, given the world description represented in Figure 8.5, we can infer that Mary is a Woman, since she fills the role "leader" of Group G05 (a Modern Group). The following assumptions may exist in the assertional system:

1. *Unique name assumption.* Each concept is only represented by one name. This assumption exists in many hybrid systems.

2. *Closed world assumption.* All the positive statements about the world are explicitly represented in the assertional system. This means that if something is not stated in the assertional system then it is false. This assumption is used by some of the hybrid systems.

8.5 Summary

In this chapter, we presented a description of hybrid systems. We presented the motivations for their introduction and we described the syntax and semantics of the two components of a hybrid system: the terminological sub-system and the assertional sub-system. We also described the operations associated with each of these sub-systems.

8.6 Suggested readings

[Brachman 79]
A paper on semantic networks that argues for the need of knowledge representation at the epistemological level. The concepts underlying KL-ONE are presented in an initial phase of development.

[Brachman and Schmolze 85]
A good paper on KL-ONE. Reading this paper allows a better understanding of recent papers on hybrid systems.

[Woods and Schmolze 92]
A paper that describes the motivations underlying the development of hybrid systems, the research themes under consideration, and the main problems that this approach faces. It also describes a common language associated with systems of the KL-ONE family. It also presents a very complete description of the several systems originating from KL-ONE with their distinguishing characteristics.

8.7 Exercises

Still to be added.

Chapter 9

Levels of a Representation

In this chapter, we discuss the different abstraction levels that can be distinguished in a knowledge representation formalism. We also discuss generic evaluation measures that should be kept in mind when evaluation a knowledge representation formalism.

Note: This chapter will be further extened with a deeper evaluation of the formalisms discussed in Chapters 6, 7, and 8.

Knowledge is of two kinds. We know a subject ourselves, or we know where we can find information upon it.
Samuel Johnson, *Boswell's Life of Johnson*

After discussing three knowledge representation formalisms, we now turn our attention into the abstraction levels that can be considered in a formalism. For each of these abstraction levels, we discuss what are the important issues and what should be considered in order to evaluate the formalism at that level.

A possible view of a knowledge representation formalism considers the formalism under a set of abstraction layers, each one of them respects a particular aspect pertaining the representation. In this chapter we discuss four abstraction levels that can be considered in a knowledge representation formalism: the implementation level, the logical level, the epistemological level, and the conceptual level. These levels were introduced by [Brachman 79], associated with semantic networks, but may be considered in any of the representation formalisms.[1]

Independently of these abstraction levels, any knowledge representation formalism should be considered under two different perspectives, the notational perspective and the inferential perspective.

- The *notational perspective* is concerned with the way of representing knowledge in a particular format. This perspective looks into the characteristics of the formal language that is used to represent knowledge: what structures belong to the language and how can these structures be combined.

  The evaluation of a representation formalism under the notational perspective should consider the naturalness and ease that the language allows the representation of the essential aspects to be considered, the expressive power of the language, and the capability of assigning a semantics to the structures of the language.

- The *inferential perspective* is concerned with the way of generating inferences from the syntactic structures of the language. This perspective is concerned with the generation of explicit knowledge from the knowledge that is implicit in the representation.

  In the evaluation of a formalism under the inferential perspective, the choice of the rules of inference and the control structures allowed to guide the inference are the essential aspects to consider. The concerns, under this perspective, are the soundness of the inference procedure and the mechanisms that are available to control the inference.

In this chapter, for each of the abstraction levels, we discuss the primitive elements that under consideration (that is, the basic building blocks) and the

[1] In [Brachman 79], a fifth level is discussed, *the linguistic level*, in which the primitives in which the primitive elements are language-specific. We will not consider this level.

criteria that should be used to evaluate the formalism at that level, both from the notational and inferential perspectives.

9.1 The implementation level

The implementation level is the lower level that can be considered in a representation, in the sense that this is the level that is closest to the programming language used to implement the formalism. This level must necessarily exist in any representation formalism, since we are interested in creating programs that use the structures of the knowledge representation language.

The primitives that we should consider at this level, are atoms, pointers, and data structures (in what concerns the representational perspective) and procedures to manipulate these data structures (in what concerns the inferential perspective).

The main concerns that we should have when we consider a knowledge representation formalism at this level respect the time and space efficiency of the implementation. We should try to answer the following questions: What are the most adequate data structures? Which is the best indexing technique between them in order to allow a fast and efficient access? What is the best technique to write an interpreter for these structures?

9.1.1 Notational perspective

The notational perspective is concerned with the definition of data structures that are used to implement the knowledge representation structures of the formalism. These structures must take into account both space and time considerations. The space considerations relate to the amount of memory that the data structures will require and the time considerations will be concerned with the efficiency of the access to these data structures. The data structures are typically linked, through pointers, to facilitate access.

9.1.2 Inferential perspective

The inferential perspective is concerned with the development and implementation of algorithms that carry out the desired inferences in an efficient way. Again, under this perspective space and time considerations should be taken into account.

9.2 The logical level

The logical level is concerned with the aspects of the representation from the stand point of the logic. At this level, we consider the representation formalism as if it were a logic. Recall, that a logic is composed of a declarative language,

Figure 9.1: Representation of "John does not love Mary".

a deduction system that is responsible for generating new sentences of the language from existing sentences, and a semantics that tells how to evaluate the meaning of each one of the sentences of the language.

At this level, we consider the representation as a set of logical primitives, eventually with an indexing scheme on top of it, and study the properties of the formalism under the perspective of the logic.

The primitives to consider, at this level, are propositions, predicate letters, constants, variables, logical connectives and quantifiers (which respect the notational perspective) and rules of inference (which respect the inferential perspective).

The concerns to have a this level relate to the decision of the logical operators to use, to the type of logic that underlies the representation (Are we just interested in first-order logic or shall we consider other aspects such as knowledge and belief? Shall we use classical logic or one of its extensions, or shall we forbid certain types of inference allowed in classical logic?), to the types of arguments that are allowed in the logic,[2] and to the properties of the logic (Is it sound? Is it complete?).

There are two adequacy criteria to be considered at this level. Fist of all the language must have a semantics. In other words, there should be a way of assigning meanings to the structures of the representation, in such a way that the meaning of a complex structure is completely defined by the meanings of the structures that compose it and by the structuring operators. On the other hand, the inference system should not "make mistakes", that is, the rules of inference should be sound in relation with the semantics of the representation.

[2] Note that this issue is different from the choice of the underlying logic. We may have a language that corresponds to the language of first-order logic but lack, in the inference system, some inference rules, for example disjunction elimination, which leads to a logic that is not complete.

Figure 9.2: Example of representation.

Although the logical level should be considered by any representation formalism, there have been efforts to study structured formalisms under the logical level perspective. Among these, are the semantic networks of [Hendrix 79], [Schubert 76], and [Schubert, Goebel, and Cercone 79], that can be considered as a programming-oriented logic, with associative links between concepts and propositions. In Figure 9.1 we show the representation of "John does not love Mary" ($\neg Loves(John, Mary)$) in the network of [Schubert, Goebel, and Cercone 77].[3]

9.2.1 Notational perspective

The notational perspective is concerned with the expressive power of the representation at the logical level, that is, with what is representable. The following are some of the possible questions that may be raised when evaluating the logical level under the notational perspective: Are disjunctions allowed? Are assertions about sets of entities allowed? Are rules with exceptions allowed? Which kind of entities does the language allow us to consider: Statements about the world? Time? Knowledge and belief?

Another concern, under the notational perspective, is the evaluation of the meanings of the representation structures, in the case pictorial representations are used. For example, given the representation of Figure 9.2, does it mean that:

- x belongs to the class y? In first-order logic, $Member(x, y)$;
- all x are y? In first-order logic, $\forall(w)\ [Member(w, x) \rightarrow Member(w, y)]$;
- x has the same properties as y? In first-order logic, $\forall(w)\ [HasProp(y, w) \rightarrow HasProp(x, w)]$.

Under the notational perspective, we should also consider what type of logic the representation formalism corresponds to. First-order logic, higher-order logic, modal logic, non-monotonic logic, and so on.

Out of the three formalisms that we have discussed, frames are clearly the formalism which are the weakest at the logical level. The hierarchy of frames specifies both a class/sub-class relation or a member/class relation (depending on the kinds of frames that are involved in the hierarchy). The inheritance of values corresponds to an implicit statement about how the properties of classes propagate to their sub-classes or members. There is no possibility to express statements involving disjunctions.

[3] Figure from [Schubert, Goebel, and Cercone 79, p. 135].

Figure 9.3: Role structure.

SNePS, on the other hand, is the formalism that offers the richest description at the logical level.

9.2.2 Inferential perspective

Under the inferential perspective, we should consider the properties of the deductive system. Does it have adequate rules on inference (in the sense that the resulting logical system is sound and/or complete)?

9.3 The epistemological level

The epistemological level is concerned with the definition of the kinds of structuring primitives that are necessary in a knowledge representation language and with the definition of the types of inference that should be supplied. At this level, there are no decisions taken about *which* particular primitives should be used, neither about *what* particular inference mechanisms should be used. The concerns are just about their kinds.

The primitives to consider at this level, in what concerns the notational perspective, are the types of concepts to be represented (for example, generic and individual concepts), the kinds of concept decompositions, and the structural description of concepts. The inferential perspective, on the other hand, considers, as primitives, the inference mechanisms.

KL-ONE was the first representation formalism to explicitly deal with the epistemological level. In Figure 9.3 we show some one the initial concerns of the description of a role associated with a concept.[4]

The concerns, at this level, respect the kinds of primitives to be used rather than the actual primitives used. For example, the decision, in SNePS, to use

[4] Figure from [Brachman 79, p.36].

nodes to represent concepts and to use arcs to represent non-conceptual relations between concepts, is an example of a decision at the epistemological level.

The adequacy criteria aim at evaluating whether the representation reflects the natural way of organizing the information about the domain. The language used to represent knowledge should be modular in the sense that if a particular aspect of the representation changes, only part of the representation should be changed. As part of the adequacy criteria one should look into the granularity of the representation, that is how easy it is to define particular primitives and inference procedures.

9.3.1 Notational perspective

This level, under the notational perspective, is concerned with the way to structure concepts, with the distinction between generic concepts (i.e., concepts that represent classes of individuals) and specific concepts (i.e., concepts that represent particular individuals), as well as with the specifications of what are the characteristics that define a concept.

The natural way in which frames represent certain domains (domains where there is a natural hierarchy of classes) is an advantage under the notational perspective of the epistemological level since it relates with the way to structure concepts and to the types of primitives allowed.

9.3.2 Inferential perspective

Under the inferential perspective, this level is concerned with the choice of the inference mechanisms to be used in the representation, for example, inheritance, pattern-based reasoning, path-based reasoning, and so on.

Under this perspective, frames use inheritance as the basic inference mechanism; SNePS uses rule-based inference and path-based inference; KL-ONE uses inheritance in the terminological formalism, and rule-based inference on the assertional formalism, together with property inheritance.

9.4 The conceptual level

The conceptual level is concerned with the definition of the primitives that should be included in the representation and with the study of the inference procedures associated with them. In other words, having selected a knowledge representation formalism, what are the representation primitives? Some of these primitives may be system-defined, while other primitives may be user-defined.

The primitives to be considered at this level are the conceptual relations (cases), primitive objects, and primitive actions, as well as the inference mechanisms associated with them.

The concerns to have at this level should address the question of whether the number of primitives defined are enough to represent the range of knowledge

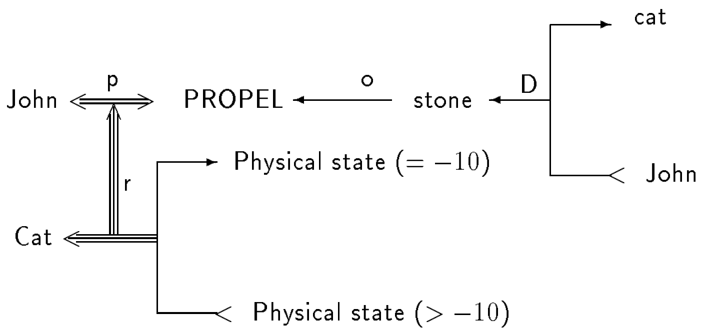

Figure 9.4: Example of representation.

that one is aiming at, and the evaluation of whether the prescribed inferences are sufficient for the type of behavior that one expects from the reasoning system.

As examples of formalisms that explicitly address knowledge representation at the conceptual level are conceptual dependency [Schank 75] and the networks of [Simmons 73]. In Figure 9.4[5] we show, using conceptual dependency, the representation of "John threw a stone to the cat and killed him".

9.4.1 Notational perspective

At this level, the notational perspective is concerned with the available primitives and the way of combining them. The adequacy criteria relating this perspective addresses the study of the precision and ease with which particular types of knowledge can be represented.

For example, the choice of pre-defined arcs to define non-standard connectives in SNePS is a decision taken at the conceptual level. Another aspect pertaining the notational perspective at the conceptual level is the decision, in SNePS, to allow the user to define its own arcs (case frames) for representing knowledge. When one is choosing a set of case frames for a particular representation in SNePS is taking a decision at the conceptual level.

9.4.2 Inferential perspective

The inferential perspective defines the available inferences associated with each of the the primitives or combination between them.

In the case of conceptual dependency, this decision was taken by the designers of the system when they defined the inferences associated with each of the primitives. In the case of SNePS, this decision is taken by the user of the system when rules are defined to express the meanings of the representations. For example, the rules in Figures 6.28 and 6.29.

[5]This is the same as Figure 6.36.

9.5 Summary

In this chapter we discussed the different level that can be distinguished in a knowledge representation formalism.

Note: This section will be extended

9.6 Suggested readings

[Brachman 79]
The paper that introduces the distinction between the several levels of a representation. It also presents, the historical evolution of the work in knowledge representation up to 1979.

[Delgrande and Mylopolos 87]
Describes several aspects about the nature of knowledge and its representation in AI. It discusses aspects that, typically, cause problems in the representation.

[Woods 87]
It discusses important aspects in the evaluation of a representation.

9.7 Exercises

1. Provide an analysis of SNePS, at the logical, epistemological and conceptual levels.
2. Provide an analysis of KEE, at the logical, epistemological and conceptual levels.
3. Provide an analysis of KL-ONE, at the logical, epistemological and conceptual levels.
4. Which knowledge representation formalism do you prefer. Based on the aspects discussed in the chapter, provide arguments for using it, face of other formalisms.
5. Which knowledge representation formalism do you dislike most. Based on the aspects discussed in the chapter, provide arguments for not using it, face of other formalisms.
6. Discuss one of the knowledge representation formalisms that were presented in this book, by evaluating its advantages and drawbacks when compared with logic.

Chapter 10

Ontologies

1.
2.
3.
4.
5.
6. Summary
7. Recommended readings
8. Exercises

...

...quote...
author, *title*

...

10.1 Summary

10.2 Recommended readings

[ref]
text

10.3 Exercises

1. ...

Chapter 11

Knowledge Sharing

1.
2.
3.
4.
5.
6. Summary
7. Recommended readings
8. Exercises

...

...quote...
author, *title*

...

11.1 Summary

11.2 Recommended readings

[ref]
text

11.3 Exercises

1. ...

References

Allen J., Fikes R. and Sandewall E., *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, San Mateo, CA: Morgan Kaufmann Publishers Inc., 1991.

Allen J.F. and Kautz H.A., "Logicism is alive and well", *Computational Intelligence 3*, pp. 161-162, 1987.

Anderson A.R. and Belnap N.D., *Entailment, The Logic of Relevance and Necessity*, Vol. 1, Princeton, N.J.: Princeton University Press, 1975.

Audi R., *Belief, Justification, and Knowledge*, Belmont, CA: Wadsworth Publishing Co., 1988.

Bartlett F., *Remembering*, Cambridge, UK: Cambridge University Press, 1932.

Besnard P., *An Introduction to Default Logic*, Heidelberg, Germany: Springer-Verlag, 1989.

Birnbaum L., "Rigor Mortis: A Response to Nilsson's Logic and Artificial Intelligence", *Artificial Intelligence 47*, No. 1-3, pp. 57-77, 1991.

Bobrow D. (ed.), "Special Issue on Non-Monotonic Reasoning", *Artificial Intelligence 13*, No. 1-2, 1980.

Bobrow D.G. and Collins A., *Representation and Understanding: Studies in Cognitive Science*, New York: Academic Press, 1975.

Bobrow D.G. and Winograd T., "An Overview of KRL, a Knowledge Representation Language", *Cognitive Science 1*, pp. 3-46, 1977. Also in *Readings in Knowledge Representation*, Brachman and Levesque (eds.), Los Altos, CA: Morgan Kaufmann, pp. 263-285, 1985.

Brachman R. and Smith B., "Special Issue on Knowledge Representation", *SIGART Newsletter 70*, 1980.

Brachman R., Fikes R. and Levesque H., "KRYPTON: Integrating Terminology and Assertion", *Proc. AAAI-83*, Los Altos, CA: Morgan Kaufmann, pp. 31-35, 1983.

Brachman R.J and Levesque H.J. (eds.), *Readings in Knowledge Representation*, Los Altos, CA: Morgan Kaufmann Publishers Inc., 1985.

Brachman R.J. and Schmolze J.G., "An Overview of the KL-ONE Knowledge Representation System", *Cognitive Science 9*, No. 2, pp. 171-216, 1985.

Brachman R.J., Fikes R.E. and Levesque H.J., "Krypton: A Functional Approach to Knowledge Representation", *IEEE Computer 16*, No. 10, pp. 67-73, 1983. Also in *Readings in Knowledge Representation*, Brachman and Levesque (eds.), Los Altos, CA: Morgan Kaufmann, pp. 412-429, 1985.

Brachman R.J., Levesque H.J. and Reiter R., *Principles of Knowledge Representation and Reasoning: Proceedings of the First International Conference*, San Mateo, CA: Morgan Kaufmann Publishers Inc., 1989.

Brachman R.J., "A Structural Paradigm for Representing Knowledge", BBN Report 3605, Cambridge, MA: Bolt Beranek & Newman, 1978.

Brachman R.J., "On the Epistemological Status of Semantic Networks", in *Associative Networks: Representation and Use of Knowledge by Computers*, Findler (ed.), New York: Academic Press, pp. 3-50, 1979. Also in *Readings in Knowledge Representation*, Brachman and Levesque (eds.), Los Altos, CA: Morgan Kaufmann, pp. 191-215, 1985.

Brachman R.J., "The Future of Knowledge Representation", *Proc. AAAI-90*, Boston, MA: AAAI Press, pp. 1082-1092, 1990.

Carpenter B. and Thomason R., "Inheritance Theory and Path-Based Reasoning: An Introduction", in *Knowledge Representation and Defeasible Reasoning*, Kyburg et al. (eds.), Dordrecht, The Netherlands: Kluwer Academic Publishers, pp. 309-343, 1990.

Cercone N. and McCalla G., "What is Knowledge Representation", in *The Knowledge Frontier: Essays in the Representation of Knowledge*, Cercone and McCalla (eds.), New York: Springer-Verlag, pp. 1-43, 1987.

Chapman D., "Planning for Conjuntive Goals", Technical Report 802, Cambridge, MA: Department of Computer Science, MIT, 1987.

Charniak E., Riesbeck C., McDermott D.V., and Meehan J.R., *Artificial Intelligence Programming*, Hillsdale, N.J.: Lawrence Erlbaum Associates, 2nd Edition, 1987.

Cheeseman P., "In Defense of Probability", *Proc. IJCAI-85*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 1002-1009, 1985.

Chellas B.F., *Modal Logic. An Introduction*, Cambridge, UK: Cambridge University Press, 1980.

Church A., *Introduction to Mathematical Logic*, Vol. 1, Princeton, N.J.: Princeton University Press, 1956.

Collins A.M. and Quillian M.R., "Facilitating Retrieval from Semantic Memory: The Effect of Repeating Part of an Inference", in *Acta Psychologica 33 Attention and Performance III*, Sanders (ed.), Amsterdam, The Netherlands: North-Holland Publishing Co, pp. 304-314, 1970.

Collins A.M. and Quillian M.R., "How to Make a Language User", in *Organization of Memory*, Tulving and Donaldson (eds.), New York: Academic Press, pp. 309-351, 1972.

Corcoran J. (ed.), *Logic, Semantics, Metamatemathics*, Indianapolis, IN: Hackett Publishing Co., 2nd Edition, 1983.

Corcoran J., "Conceptual Structure of Classical Logic", *Philosophy and Phenomenological Research 33*, pp. 25-47, 1972.

Cravo M.R. and Martins J.P., "Being Aware of Assumptions", *Proc. of the Seventh Austrian Conference on Artificial Intelligence*, Kaindl (ed), Informatik-Fachberichte 287, Heidelberg, Germany: Springer-Verlag, pp. 137-146, 1991.

Cravo M.R. and Martins J.P., "SNePSwD: A Newcomer to the SNePS Family", *Journal of Experimental and Theoretical Artificial Intelligence 5*, pp. 135-148, 1993.

Cravo M.R., *Personal Communication*, 1994.

de Kleer J. and Williams B.C., "Diagnosing Multiple Faults", *Artificial Intelligence 32*, No. 1, pp. 97-130, 1987.

de Kleer J., Doyle J., Steele G., and Sussman G., "AMORD: Explicit Control of Reasoning", *Proc. Symposium on Artificial Intelligence and Programming Languages, SIGART Newsletter 64*, pp. 116-125, 1977. Also in *Readings in Knowledge Representation*, Brachman and Levesque (eds.), Los Altos, CA: Morgan Kaufmann Publishers, Inc., pp. 345-355, 1985.

de Kleer J., "An Assumption-based TMS", *Artificial Intelligence 28*, No. 2, pp. 127-162, 1986. Also in *Readings in Nonmonotonic Reasoning*, Ginsberg (ed.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 280-297, 1987.

de Kleer J., "Choices without Backtracking", *Proc. AAAI-84*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 79-85, 1984.

Delgrande J.P. and Mylopoulos J., "Knowledge Representation: Features of Knowledge", in *Fundamentals of Artificial Intelligence: An Advanced Course*, Bibel and Jorrand (eds.), Heidelberg, Germany: Springer-Verlag, pp. 3-36, 1987.

Dhar V. and Ranganathan N., "Integer Programming vs. Expert Systems: An Experimental Comparison", *Communications of the ACM 33*, No. 3, pp. 323-336, 1990.

Doyle J. and London P., "A Selected Description-Indexed Bibliography to the Literature on Belief Revision", *SIGART Newsletter 71*, pp. 7- 23, 1980.

Doyle J., "A Truth Maintenance System", *Artificial Intelligence 12*, No. 3, pp. 231-272, 1979. Also in *Readings in Artificial Intelligence*, Webber and Nilsson (eds.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 496-516, 1981. Also in *Readings in Nonmonotonic Reasoning*, Ginsberg (ed.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 259-279, 1987.

Doyle J., "The Ins and Outs of Reason Maintenance", *Proc. IJCAI-83*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 349-351, 1983.

Doyle J., Sandewall E., and Torasso P., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference*, San Francisco, CA: Morgan Kaufmann Publishers Inc., 1994.

Dressler O., "An Extended Basic ATMS", *Proc. of The Second Internatinal Workshop on Non-Monotonic Reasoning*, Reinfrank, de Kleer, Ginsberg and Sandewall (eds), Lecture Notes in Computer Science 346, Heidelberg, Germany: Springer-Verlag, pp. 143-163, 1988.

Drummond M.E., "A Representation of Action and Belief for Automatic Planning Systems", *Proc. 1986 Workshop on Reasoning about Actions and Plans*, Georgeff and Lansky (eds.) Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 189-211, 1987.

Eklund P., *An Epistemic Approach to Interactive Design in Multiple Inheritance Hierarquies*, Linköping Studies in Science and Technology. Dissertation 252. Linköping, Sweden: Department of Computer and Information Science, Linköping University, 1991.

Ernst G.W. and Newell A., *GPS: A Case Study in Generality and Problem-solving*, New York: Academic Press, 1969.

Etherington D.W. e Reiter R., "On Inheritance Hierarchies With Exceptions", *Proc. AAAI-83*, Los Altos, CA: Morgan Kaufmann, pp. 104-108, 1983.

Etherington D.W., "A Semantics for Default Logic", *Proc. IJCAI-87*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 495-498, 1987.

Etherington D.W., "Formalizing Nonmonotonic Reasoning Systems", *Artificial Intelligence 31*, No.1, pp. 41-85, 1987a.

Etherington D.W., "More on Inheritance Hierarquies with Exceptions: Default Theories and Inferential Distance", *Proc. AAAI-87*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 352-357, 1987b.

Etherington D.W., *Reasoning with Incomplete Information*, London, UK: Pitman Publishing, 1988.

Fahlmann S., *NETL: A System for Representing and Using Real World Knowledge*, Cambridge, MA: The MIT Press, 1979.

Fikes R. and Nilsson N., "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", *Artificial Intelligence 2*, No. 3-4, pp. 189-208, 1971

Fikes R., "Deductive Retrieval Mechanisms for State Description Models", *Proc. IJCAI-75*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 99-106, 1975.

Fikes R.E. and Kehler T., "The Role of Frame-Based Representation in Reasoning", *Communications of the ACM 28*, pp. 904-920, 1985.

Fillmore C., "The Case for Case", in *Universals in Linguistic Theory*, Bach and Harms (eds.), Chicago, IL: Holt Rinehart and Winston, pp. 1-88, 1968.

Filman R.E., "Reasoning with Worlds and Truth Maintenance in a Knowledge-Based Programming Environment", *Communications of the ACM 31*, pp. 382-401, 1988.

Fitch F., *Symbolic Logic: An Introduction*, New York: The Ronald Press Company, 1952.

Forbus K.D. and de Kleer J., "Focusing the ATMS", *Proc. AAAI-88*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 193-198, 1988.

Grdenfors P., *Knowledge in Flux: Modeling the Dynamics of Epistemic States*, Cambridge, MA: The MIT Press, 1988.

Gärdenfors P. and Makison D., "Revisions of Knowledge Systems using Epistemic Entrenchment", *Proc. of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, Vardi (ed.), Los Altos, CA: Morgan Kaufmann, pp.83-95, 1988.

Genesereth M.R. and Nilsson N.J., *Logical Foundations of Artificial Intelligence*, Los Altos, CA: Morgan Kaufmann Publishers Inc., 1987.

Gentzen G., *Collected Papers of Gerhard Gentzen*, Amsterdam, The Netherlands: North-Holland, 1969.

Ginsberg M.L., "Multiple-Valued Logics", *Proc. AAAI-86*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 243-247, 1986. Also in *Readings in Non-monotonic Reasoning*, Ginsberg (ed.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 251-255, 1987.

Ginsberg M.L., *Readings in Nonmonotonic Reasoning*, Los Altos, CA: Morgan Kaufmann Publishers Inc., 1987.

Goodwin J., *An Improved Algorithm for Nonmonotonic Dependency Network Update*, Technical Report LITH-MAT-R-82-23, Linköping, Sweden: Department of Computer and Information Science, Linköping University, 1982.

Haack S., *Philosophy of Logics*, Cambridge, UK: Cambridge University Press, 1978.

Halpern J.Y. and Moses Y., "Towards a Theory of Knowledge and Ignorance: Preliminary Report", *Proc. of the AAAI Workshop on Non-Monotonic Reasoning*, Menlo Park, CA: American Association for Artificial Intelligence, pp. 125-143, 1984.

Hanks S. and McDermott D., "Default Reasoning, Nonmonotonic Logics, and the Frame Problem", *Proc. AAAI-86*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 328-333, 1986. Also in *Readings in Non-monotonic Reasoning*, Ginsberg (ed.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 390-395, 1987.

Harman G., *Change in View: Principles of Reasoning*, Cambridge, MA: The MIT Press, 1986.

Hayes P.J., "A Representation for Robot Plans", *Proc. IJCAI-75*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 181-188, 1975.

Hayes P.J., "In Defense of Logic", *Proc. IJCAI-77*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 559-565, 1977.

Hayes P.J., "The Logic of Frames", in *Frame Conceptions and Text Understanding*, Metzing (ed,), Berlin, Germany: Walter de Gruyter and Co., pp. 46-61, 1979. Also in *Readings in Artificial Intelligence*, Webber and Nilsson (eds.), Palo Alto, CA: Tioga, pp. 451-458, 1979. Also in *Readings in Knowledge Representation*, Brachman and Levesque (eds.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 287-295, 1985.

Hawkins J.M. and Allen R., *The Oxford Encyclopedic English Dictionary*, Oxford, UK: Clarendon Press, 1991.

Hendrix G.G., "Encoding Knowledge in Partitioned Networks", in *Associative Networks: Representation and Use of Knowledge by Computers*, Findler (ed.), New York: Academic Press, pp. 51-92, 1979.

Hewitt C.E., Bishop P. and Steiger R., "A Universal Modular ACTOR Formalism for Artificial intelligence", *Proc. IJCAI-73*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 235-245, 1973.

Hewitt C.E., "A Language for Proving Theorems in Robots", *Proc. IJCAI-69*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 295-301, 1969.

Horty J.F. and Thomason R., "Mixing Strict and Defeasible Inheritance", *Proc. AAAI-88*, San Mateo, CA: Morgan Kaufmann, pp. 427-432, 1988.

Horty J.F., Thomason R. and Touretzky D.S., "A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks", *Artificial Intelligence 42*, No. 2-3, pp. 311-348, 1990.

Horty J.F., "A Credulous Theory of Mixed Inheritance", in *Inheritance Hierarquies in Knowledge Representation and Programming Languages*, Lenzerini, Nardi and Simi (eds.), pp. 13-28, New York: John Wiley and Sons Ltd, 1991.

Hughes G.E. and Cresswell M.J., *An Introduction to Modal Logic*, London, UK: Methuen and Co., 1968.

Israel D., "Some Remarks on the Place of Logic in Knowledge Representation", in *The Knowledge Frontier: Essays in the Representation of Knowledge*, Cercone and McCalla (eds.), New York: Springer-Verlag, pp. 80-91, 1987.

Israel D., "What's Wrong With Non-monotonic Logic?", *Proc. AAAI- 80*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 99-101, 1980. Also in *Readings in Non-monotonic Reasoning*, Ginsberg (ed.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 53-55, 1987.

Janlert L.-E., "Modeling Change: The Frame Problem", in *The Robot's Dilemma: The Frame Problem in Artificial Intelligence*, Pylyshyn (ed.), Norwood N.J.: Ablex Publishing Corporation, pp. 1-40, 1987.

Jaskowski S., "On the Rules of Supposition in Formal Logic", *Studia Logica 1*, 1934.

Jeffreys H., "Does a Contradiction Entail Every Proposition?", *Mind 51*, 1942.

Keene S.E., *Object-Oriented Programming in COMMON LISP: A Programmers Guide to CLOS*, Reading, MA: Addison-Wesley Publishing Co., 1989.

Kirsh D., "Foundations of AI: The Big Issues", *Artificial Intelligence 47*, pp. 3-30, 1991.

Kleene S.C., *Introduction to Metamathematics*, New York: American Elsevier Publishing Co, Inc., 1952.

Kline M., *Mathematical Thought: From Ancient to Modern Times*, Oxford, UK: Oxford University Press, 1972.

Kneale W. and Kneale M., *The Development of Logic*, Oxford, UK: Clarendon Press, 1988.

Konolige K., *A Deduction Model of Belief*, London, UK: Pitman Publishing, 1986.

Kowalski R., "A Proof Procedure Using Connection Graphs", *Journal of the ACM 22*, pp. 573-595, 1975.

Kramosil I., "A Note on Deduction Rules with Negative Premises", *Proc. IJCAI-75*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 53-56, 1975.

Kripke S.A., "Semantical Analysis of Modal Logic I, Normal Propositional Calculi", *Zeitschrift für Mathematische Logik und Grundlagen den Mathematik 9*, pp. 67-96, 1963.

Léa Sombé, *Reasoning under Imcomplete Information in Artificial Intelligence*, New York: John Wiley and Sons, Inc., 1990.

Lehmann F. (ed.), *Semantic Networks in Artificial Intelligence*, N.Y.: Pergamon Press, 1992.

Lehrer K., *Theory of Knowledge*, London, UK: Routledge, 1990.

Lemmon E.J., *Beginning Logic*, Indianapolis, IN: Hackett Publishing Co., 1978.

Lenzerini M., Nardi D. and Simi M., *Inheritance Hierarchies in Knowledge Representation and Programming Languages*, New York: John Wiley and Sons, 1991.

Levesque H.J., "Foundations of a Functional Approach to Knowledge Representation", *Artificial Intelligence 23*, No. 2, pp. 155-212, 1984.

London P., "Dependency Networks as Representation for Modelling in General Problem Solvers", Ph.D. Dissertation, Technical Report 698, College Park, MD: Department of Computer Science, University of Maryland, 1978.

Lukaszewicz W., *Non-Monotonic Reasoning: Formalization of Commonsense Reasoning*, Chichester, UK: Ellis Horwood Limited, 1990.

MacGregor R., "The Evolving Technology of Classification-Based Knowledge Representation Systems", in *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Sowa (ed.), San Mateo: CA, Morgan Kaufmann, pp. 385-400, 1991.

Machtey M. and Young P., *An Introduction to the General Theory of Algorithms*, New York: Elsevier North Holland Inc., 1978.

Maida A.S. and Shapiro S.C., "Intensional Concepts in Propositional Semantic Networks", *Cognitive Science 6*, No. 4, pp. 291-330, 1982. Also in *Readings in Knowledge Representation*, Brachman and Levesque (eds.), Los Altos, CA: Morgan Kaufmann, pp. 169-189, 1985.

Martins J., "Reasoning in Multiple Belief Spaces", Ph.D. Dissertation, Technical Report 203, Buffalo, N.Y.: Department of Computer Science, State University of New York at Buffalo, 1983.

Martins J.P., "Belief Revision", in *Encyclopedia of Artificial Intelligence*, Shapiro (ed.), pp.58-62, New York, N.Y.: John Wiley and Sons, 1987.

Martins J., "The Truth, The Whole Truth, and Nothing But The Truth: An Indexed Bibliography to the Literature on Truth Maintenance Systems", *AI Magazine 11*, No. 5, pp. 7-25, 1990.

Martins J.P. and Shapiro S.C., "A Model for Belief Revision", *Artificial Intelligence 35*, No. 1, pp. 25-79, 1988.

Martins J.P. and Shapiro S.C., "Reasoning in Multiple Belief Spaces", *Proc. IJCAI-83*, Los Altos, CA: William Kaufmann, pp. 370-373, 1983.

Martins J.P., "A Structure for Epistemic States", in *New Directions for Intelligent Tutoring Systems*, Costa (ed.), NATO ASI Series F, Heidelberg, Germany: Springer-Verlag, pp. 198-212, 1992.

McAllester D., "A three-valued truth maintenance system", Technical Report Memo 473, Cambridge, MA: AI-Lab, Massachussetts Institute of Technology, 1978.

McCarthy J. and Hayes P., "Some Philosophical Problems from the Standpoint of Artificial Intelligence", in *Machine Intelligence 4*, Meltzer and Michie (eds.), Edinburgh, UK: Edinburgh University Press, pp. 463-502, 1969.

McCarthy J., "Applications of Circumscription to Formalizing Common- Sense Knowledge", *Proc. of the AAAI Workshop on Non-monotonic Reasoning*, Menlo Park, CA: American Association for Artificial Intelligence, pp. 295-324, 1984. Also in *Artificial Intelligence 28*, No. 1, pp. 89-116, 1986. Also in *Readings in Non-monotonic Reasoning*, Ginsberg (ed.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 153-166, 1987.

McCarthy J., "Circumscription – A Form of Non-Monotonic Reasoning", *Artificial Intelligence 13*, No. 1-2, pp. 27-39, 1980. Also in *Readings in Non-monotonic Reasoning*, Ginsberg (ed.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 145-152, 1987.

McCarthy J., "Mathematical Logic in Artificial Intelligence", in *The Artificial Intelligence Debate–False Starts, Real Foundations*, Grabaud (ed.), pp. 297-311, Cambridge, MA: The MIT Press, 1988. Also in *Formalizing Commonsense: Papers by John McCarthy*, Lifschitz (ed.), Norwood, N.J.: Ablex Publishing Co., pp. 237-249, 1990.

McCarthy J., "Programs with Common Sense", in *Semantic Information Processing*, Minsky (ed.), Cambridge, MA: The MIT Press, pp. 403-410, 1968. Also in *Readings in Knowledge Representation*, Brachman and Levesque (eds.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 299-307, 1985.

McCawley J.D., *Everything that Linguists have Always Wanted to Know about Logic (but were ashamed to ask)*, Chicago, IL: The University of Chicago Press, 1981.

Mendelson E., *Introduction to Mathematical Logic*, Belmont, CA: Wadsworth Publishing Co., 1987.

Minsky M., "A Framework for Representing Knowledge", in *The Psychology of Computer Vision*, Winston (ed.), New York: McGraw-Hill Book Co., pp. 211-277, 1975. Also in *Readings in Knowledge Representation*, Brachman and Levesque (eds.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 245-262, 1985.

Moore R.C. "Possible - World Semantics for Autoepistemic Logic", *Proc. of the AAAI Workshop on Non-monotonic Reasoning*, Menlo Park, CA: American Association for Artificial Intelligence, pp. 344-354, 1984. Also in *Readings in Non-monotonic Reasoning*, Ginsberg (ed.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 137-142, 1987.

Moore R.C. "Semantical Considerations on Non-monotonic Logic", *Artificial Intelligence 25*, No. 1, pp. 75-94, 1985. Also in *Readings in Non-monotonic Reasoning*, Ginsberg (ed.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 127-136, 1987.

Moore R.C., "Autoepistemic Logic", in *Non-Standard Logics for Automated Reasoning*, Smets et. al. (eds.), New York: Academic Press, pp. 105-127, 1988.

Moore R.C., "Semantical Considerations on Non-monotonic Logic", *Proc. IJCAI-83*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 272-279, 1983.

Moore R.C., "The Role of Logic in Artificial Intelligence", Technical Note 335, Stanford, CA: SRI International, Artificial Intelligence Center, 1984.

Moore R.C., "The Role of Logic in Knowledge Representation and Commonsense Reasoning", *Proc. AAAI-82*, pp. 428-433, 1982. Also in *Readings in Knowledge Representation*, Brachman and Levesque (eds.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 335-341, 1985.

Morris P.H., "A Truth Maintenance Based Approach to the Frame Problem", *Proc. of the 1987 Workshop on the Frame Problem in Artificial Intelligence*, Brown (ed.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 297-307, 1987b.

Morris P.H., "Curing Anomalous Extensions", *Proc. AAAI-87*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 437-442, 1987a.

Nado R. and Fikes R., "Semantically Sound Inheritance for a Formally Defined Frame Language with Defaults", *Proc. AAAI-87*, Los Altos, CA: Morgan Kaufmann, pp. 443-449, 1987.

Nebel B., *Reasoning and Revision in Hybrid Representation Systems*, Lecture Notes in Artificial Intelligence 422, Heidelberg, Germany: Springer-Verlag, 1990.

Nebel B., Rich C. and Swartout W., *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*, San Mateo, CA: Morgan Kaufmann Publishers Inc., 1992.

Newell A., "The Knowledge Level", *AI Magazine 2*, No. 2, pp. 1-20, 1980. Also in *Artificial Intelligence 18*, No. 1, pp. 87-127, 1980.

Newell A., "Physical Symbol Systems", *Cognitive Science 4*, No. 2, pp. 135-183, 1982.

Newell A and Simon H.A., "Computer Science as Empirical Inquiry: Symbols and Search", *Communications of the ACM 19*, No. 3, pp. 113-126, 1976. Also in *The Phylosophy of Artificial Intelligence*, Boden (ed.), Oxford, UK: Oxford University Press, pp. 105-132, 1990.

Nilsson N., "Logic and Artificial Intelligence", *Artificial Intelligence 47*, No. 1-3, pp. 31-56, 1991.

Nilsson N., *Problem Solving Methods in Artificial Intelligence*, New York: McGraw-Hill Book Co., 1971.

Nutter J.T., "What Else is Wrong with Non-Monotonic Logics? Representational and Informational Shortcomings", *Proc. of the Fifth Annual Conference of the Cognitive Science Society*, 1983.

Perlis D., "On the Consistency of Commonsense Reasoning", *Computational Intelligence 2*, pp. 180-190, 1987.

Pinto-Ferreira C. and Martins J.P., "A Formal System for Reasoning about Change", *Proc. ECAI-90*, Aiello (ed.), London, UK: Pitman Publishing, pp. 503-508, 1990.

Pinto-Ferreira C., *Personal Communication*, 1991.

Quillian M.R., "Semantic Memory" in *Semantic Information Processing*, Minsky (ed.), Cambridge, MA: The MIT Press, pp. 227-270, 1968.

Quine W.V., *Philosophy of Logic*, Cambridge, MA: Harvard University Press, 2nd Edition, 1986.

Ramsay A., *Formal Methods in Artificial Intelligence*, Cambridge, UK: Cambridge University Press, 1988.

Rapaport W.J., "Logic", in *Encyclopedia of Artificial Intelligence*, Shapiro (ed.), pp. 536-538, New York: John Wiley and Sons, 1987a.

Rapaport W.J., "Predicate Logic", in *Encyclopedia of Artificial Intelligence*, Shapiro (ed.), pp. 538-544, New York: John Wiley and Sons, 1987b.

Raphael B., "SIR: Semantic Information Retrieval", in *Semantic Information Processing*, Minsky (ed.), Cambridge, MA: The MIT Press, pp. 33-145, 1968.

Rautenberg W. (ed.), *Ω-Bibliography of Mathematical Logic*, Volume II: Non-Classical Logics, Berlim, Germany: Springer-Verlag, 1987.

Read S., *Relevant Logic: A Philosophical Examination of Inference*, Oxford, UK: Basil Blackwell Ltd, 1988.

Reinfrank M., *Fundamentals and Logical Foundations of Truth Maintenance*, Linköping Studies in Science and Technology. Dissertation 221. Linköping, Sweden: Department of Computer and Information Science, Linköping University, 1989.

Reiter R. and Criscuolo G., "On Interacting Defaults", *Proc. IJCAI-81*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 270-276, 1981. Also in *Readings in Monmonotonic Reasoning*, Ginsberg (ed.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 94-100, 1987.

Reiter R., "A Logic for Default Reasoning", *Artificial Intelligence 13*, No. 1-2, pp. 81-132, 1980. Also in *Readings in Monmonotonic Reasoning*, Ginsberg (ed.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 68-93, 1987.

Reiter R., "Non-monotonic Reasoning", in *Exploring Artificial Intelligence*, Shrobe (ed.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 439-481, 1988. Also in *Annual Review of Computer Science 2*, Tramb (ed.), Palo Alto, CA: Annual Reviews Inc., pp. 947-986, 1987.

Reiter R., "On Closed Word Data Bases", in *Logic and Data Bases*, Gallaire and Minker (eds.), New York: Plenum Press, pp. 55-76, 1978b. Also in *Readings in Artificial Intelligence*, Webber and Nilsson (eds.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 119-140, 1981.

Reiter R., "On Reasoning by Default", *Proc. Theoretical Issues in Natural Language Processing 2*, pp. 210-218, 1978a. Also in *Readings in Knowledge Representation*, Brachman and Levesque (eds.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 401-410, 1985.

Roberts D.D., *The Existential Graphs of Charles S. Pierce*, The Hage, The Netherlands: Mouton, 1973.

Roberts R.B. and Goldstein I.P., "The FRL Manual", AI Memo 409, Artificial Intelligence Laboratory, Cambridge, MA: Massachusetts Institute of Technology, 1977.

Sandewall E., "An Approach to the Frame Problem and its Implementation", *Machine Intelligence 7*, Meltzer and Michie (eds.), Edinburgh, Scotland: Edinburgh University Press, pp. 195-204, 1972.

Schank R.C. and Abelson R.P., *Scripts, Plans, Goals and Understanding*, Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers, 1977.

Schank R.C. and Riesbeck C.K., *Inside Computer Understanding: Five Programs plus Miniatures*, Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers, 1981.

Schank R.C., *Conceptual Information Processing*, Amsterdam, The Netherlands: North-Holland Publishing Co., 1975.

Schmolze J.G. and Lipkis T., "Classification in the KL-ONE Knowledge Representation System", *Proc. IJCAI-83*, Los Altos, CA: Morgan Kaufmann, pp. 330-332, 1983.

Schubert L.K., Goebel R.G., and Cercone N.J., "The Structure and Organization of a Semantic Net for Compreension and Inference", in *Associative Networks: Representation and Use of Knowledge by Computers*, Findler (ed.), New York: Academic Press, pp. 121-175, 1979.

Schubert L.K., "Extending the Expressive Power of Semantic Networks", *Artificial Intelligence 7*, No. 2, pp. 163-198, 1976.

Shapiro S.C., "A Net Structure for Semantic Information Storage, Deduction, and Retrieval", *Proc. IJCAI-71*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 512-523, 1971.

Shapiro S.C., "The SNePS Semantic Network Processing System", in *Associative Networks: Representation and Use of Knowledge by Computers*, Findler (ed.), New York, N.Y.: Academic Press Inc., pp. 179-203, 1979a.

Shapiro S.C., "Using Non-standard Connectives and Quantifiers for Representing Deduction Rules in a Semantic Network", presented at "Current Aspects of AI research", a seminar held at the Electrotechnical Laboratory, Tokyo, August 27-28, 1979b.

Shapiro S.C. and Martins J.P., "Recent Advances and Developments: The SNePS 2.1 Report", in *Current Trends in SNePS - Semantic Network Processing System*, Kumar (ed.), Lecture Notes in Artificial Intelligence 437, Heidelberg, Germany: Springer-Verlag, pp. 1-13, 1990.

Shapiro S.C. and Rapaport W.J., "SNePS Considered as a Fully Intensional Propositional Semantic Network", in *The Knowledge Frontier: Essays in the Representation of Knowledge*, Cercone and McCalla (eds.), New York: Springer Verlag, pp. 262-315, 1987.

Shapiro S.C. and Rapaport W.J., "The SNePS Family", in *Semantic Networks in Artificial Intelligence*, Lehmann (ed.), Oxford, UK: Pergamon Press Ltd., pp. 243-275, 1992.

Shapiro S.C., Rapaport W.J., Cho S-H., Choi J., Feit E., Haller S., and Kumar D., "A Dictionary of SNePS Case Frames", Buffalo, N.Y.: Department of Computer Sciemce. State University of New York at Buffalo, 1993.

Shapiro S. and Wand M., "The Relevance of Relevance", Technical Report No.46, Bloomington, IN: Computer Science Department, Indiana University, 1976.

Sickel S., "A Search Technique for Clause Interconnectivity Graphs", *IEEE Transactions on Computers*, Vol. C-25, No. 8, pp. 823-835, 1976.

Simmons R.F., "Semantic Networks: Their Computation and Use for Understanding English Sentences", in *Computer Models of Though and Language*, Schank and Colby (eds.), San Francisco, CA: W.H. Freeman and Co., pp. 63-113, 1973.

Smith B. and Kelleher G., *Reason Maintenance Systems and Their Applications*, Chichester, UK: Ellis Horwood Limited, 1988.

Smith B.C., "Reflection and Semantics in a Procedural Language", Technical Report MIT/LCS/TR-272, Cambridge, MA: Massachusetts Institute of Technology, 1982. Partially published in *Readings in Knowledge Representation*, Brachman and Levesque (eds.), Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 31-39, 1985.

Snyder A., "Inheritance in Object-Orientated Programming", in *Inheritance Hierarchies in Knowledge Representation and Programming Languages*, Lenzerini, Nardi and Simi (eds.), New York: John Wiley and Sons, pp. 153-172, 1991.

Sowa J.F.(ed.), *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, San Mateo, CA: Morgan Kaufmann Publishers Inc., 1991.

Sowa J.F., *Conceptual Structures: Information Processing in Mind and Machine*, Reading, MA: Addisson-Wesley, 1984.

Stallman R. and Sussman G., "Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis", *Artificial Intelligence 9*, No. 2, pp. 135-196, 1977.

Stalnaker R., *Inquiry*, Cambridge, MA: The MIT Press, 1984.

Stefik M., "An Examination of a Frame-structured Representation System", *Proc. IJCAI-79*, Los Altos, CA: Morgan Kaufmann, pp. 845-852, 1979.

Struss P., "Extensions to ATMS-based Diagnosis", in *Artificial Intelligence in Engineering: Diagnosis and Learning*, Gero (ed.), Amsterdam, The Netherlands: Elsevier, pp. 3-28, 1988.

Sussman G., Winograd T. and Charniak E., "Micro-Planner Reference Manual", Technical Report 203A, Cambridge, MA: AI Lab., Massachussets Institute of Technology, 1971.

Tarski A., *Introduction to Logic and to the Methodology of Deductive Sciences*, New York: Oxford University Press, 1965.

Tennant N., "Perfect Validity, Entailment and Paraconsistency", *Studia Logica 43*, pp. 181-200, 1984.

Thayse A., *From Standard Logic to Logic Programming*, Chichester, UK: John Wiley and Sons, 1988.

Thomason R., Horty J. and Touretsky D., "A Calculus for Inheritance in Monotonic Semantic Nets", *Proc. Second International Symposium on Methodologies for Intelligent Systems*, Ras and Zemankova (eds.), North-Holland, p. 280-287, 1987.

Tourestzky D.S., *The Methematics of Inheritance Systems*, London, UK: Pitman Publishing, 1986.

Touretzky D.S., Horty J.F. and Thomason R.H., "A Clash of Intuitions: The Current State of Nonmonotonic Multiple Inheritance Systems", *Proc. IJCAI-87*, Los Altos: Morgan Kaufmann, pp. 476-482, 1987.

Touretzky D.S., "Inheritance Hierarchy", in *Encyclopedia of Artifical Intelligence*, 2nd Edition, Shapiro (ed.), New York: John Wiley and Sons, pp. 690-701, 1992.

Turner R., *Logics for Artificial Intelligence*, Chicester, UK: Ellis Horwood Ltd., 1984.

Turner R., *Truth and Modality for Knowledge Representation*, London, UK: Pitman Publishing, 1990.

Whitehead A.N. and Russell B., *Principia Mathematica*, Cambridge, UK: Cambridge University Press, 1910 (2nd Edition, 1987).

Wilks Y., "An Artificial Intelligence Approach to Machine Translation", in *Computer Models of Thought and Language*, Schank and Colby (eds.), San Francisco, CA: W.H. Freeman and Co., pp. 114-151, 1973.

Wilks Y., "Good and Bad Arguments about Semantic Primitives", D.A.I. Research Report N. 42, University of Essex, UK, 1977.

Winograd T., "Frame Representation and the Declarative/Procedural Controversy", in *Representation and Understanding: Studies in Cognitive Science*, Bobrow and Collins (eds.), New York, NY: Academic Press Inc., pp. 185-210, 1975.

Winograd T., *Understanding Natural Language*, N.Y.: Academic Press, 1972.

Wolfram S., *Philosophical Logic, An Introduction*, London, UK: Routledge, 1989.

Woods W.A., "What's in a Link: Foundations for Semantic Networks", in *Representation and Understanding, Studies in Cognitive Science*, Bobrow and Collins (eds.), New York: Academic Press, pp. 35-82, 1975. Also in *Readings in Knowledge Representation*, Brachman and Levesque (eds.), Los Altos, CA: Morgan Kaufmann, pp. 217-241, 1985.

Woods W.A., "Knowledge Representation: What's Important About It?", in *The Knowledge Frontier: Essays in the Representation of Knowledge*, Cercone and McCalla (eds.), New York: Springer-Verlag, pp. 44-79, 1987.

Woods W.A. and Schmolze J.G., "The KL-ONE Family", in *Semantic Networks in Artificial Intelligence*, Lehmann (ed.), Oxford, UK: Pergamon Press Ltd., pp. 133-177, 1992.

Zadeh L., "Syllogistic Reasoning as a Basis for Combination of Evidence in Expert Systems", *Proc. IJCAI-85*, Los Altos, CA: Morgan Kaufmann Publishers Inc., pp. 417-419, 1985.