

Knowledge Representation and Reasoning Logics for Artificial Intelligence

Stuart C. Shapiro
Department of Computer Science and Engineering
and Center for Cognitive Science
201 Bell Hall
University at Buffalo, The State University of New York
Buffalo, NY 14260-2000
shapiro@cse.buffalo.edu

January 13, 2004

Copyright © 2004 by Stuart C. Shapiro
All rights reserved.

Contents

Acknowledgments	v
1 Introduction	1
1.1 Knowledge Representation	1
1.2 Logic	3
2 Propositional Logic	5
2.1 Introduction	5
2.2 The “Standard” Logic	5
2.2.1 Syntax	5
2.2.2 Semantics	6
2.2.3 Proof Theory	10
2.3 Clause Form Logic	11
2.3.1 Syntax	11
2.3.2 Semantics	11
2.3.3 Proof Theory	11
3 Elementary Predicate Logic	13
3.1 The “Standard” Logic	13
3.1.1 Syntax	13
3.1.2 Semantics	13
3.1.3 Proof Theory	13
3.2 Clause Form Logic	13
3.2.1 Syntax	13
3.2.2 Semantics	13
3.2.3 Proof Theory	13
3.3 SNePS Logic	13
3.3.1 Syntax	13
3.3.2 Semantics	13
3.3.3 Proof Theory	13
4 Full First-Order Predicate Logic	15
4.1 The “Standard” Logic	15
4.1.1 Syntax	15

4.1.2	Semantics	15
4.1.3	Proof Theory	15
4.2	Clause Form Logic	15
4.2.1	Syntax	15
4.2.2	Semantics	15
4.2.3	Proof Theory	15
4.3	SNePS Logic	15
4.3.1	Syntax	15
4.3.2	Semantics	15
4.3.3	Proof Theory	15
	Bibliography	17

Acknowledgments

The introduction is based, in part, on (Shapiro, 2003). The material on Logic is based, in part, on (Shapiro, 2000).

Chapter 1

Introduction

Reports that say something hasn't happened are always interesting to me, because as we know, there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns — the ones we don't know we don't know. [Donald Rumsfeld, February 2002]

We think we know what he means. But we don't know if we really know. [John Lister, spokesman for Britain's Plain English Campaign, December 1, 2003]

1.1 Knowledge Representation

Artificial Intelligence (AI) is a field of computer science and engineering concerned with the computational understanding of what is commonly called intelligent behavior, and with the creation of artifacts that exhibit such behavior (Shapiro, 1992, p. 54).

Knowledge representation (KR) is a subarea of Artificial Intelligence concerned with understanding, designing, and implementing ways of representing information in computers so that programs can use this information

- to derive information that is implied by it,
- to converse with people in natural languages,
- to plan future activities,
- to solve problems in areas that normally require human expertise.

Deriving information that is implied by the information already present is a form of *reasoning*. Because knowledge representation schemes are useless without the ability to reason with them, the field is usually known as “knowledge representation and reasoning” (KRR). KRR can be seen to be both necessary and sufficient for producing general intelligence. That is, KRR is an AI-complete area (Shapiro, 1992, p. 56).

Sentence	How we understand it
<i>John likes ice cream.</i>	John likes to eat ice cream.
<i>Mary likes Asimov.</i>	Mary likes to read books written by Isaac Asimov.
<i>Bill flicked the switch. The room was flooded with light.</i>	Bill moved the switch to the “on” position, which caused a light to come on, which lit up the room Bill was in.
<i>Betty opened the blinds. The courtyard was flooded with light.</i>	Betty adjusted the blinds so that she could see through the window they were in front of, after which she could see that the courtyard on the other side of the window was bright.

Table 1.1: Some sentences and how we understand them.

Many philosophers consider knowledge to be justified true belief. Thus, if John believes that the world is flat, we would not say that John *knows* that the world is flat, because he is wrong—“the world is flat” is not true. Also, it may be that Sally believes that the first player in chess can always win, Betty believes that the second player can always win, and Mary believes that, with optimal play on both sides, chess will always end in a tie. One of them is correct, but we would still not say that any of them *knows* the answer, because their belief cannot have been justified by a complete analysis of the game. A computer system could not limit its information to knowledge in this strict sense, so it would be more accurate to say that the topic being discussed is *belief representation*, rather than knowledge representation. Nevertheless, we will continue to use “knowledge representation,” because that has become accepted as the name of this subject.

The field of knowledge representation began, around 1958, with an investigation of how a computer might be able to represent and use the kind of commonsense knowledge we have when we decide that to get from our house to the airport, we should walk to our car and drive to the airport, rather than, for example drive to our car and then walk to the airport. The manifesto of KRR may be taken to be

a program has common sense if it automatically deduces for itself a sufficiently wide class of immediate consequences of anything it is told and what it already knows. . . In order for a program to be capable of learning something it must first be capable of being told it. (McCarthy, 1959)

In the 1960s and 1970s, much knowledge representation research was concerned with representing and using the kind of information we get from reading and from talking to other people; that is, the information that is often expressed in natural languages, and that underlies our ability to understand and to use natural languages. For example, we probably understand each of the sentences in the first column of Table 1.1 as shown in the second column, by adding our “background knowledge” to what the sentences explicitly say. Moreover, our understanding of English includes our being able to make the following inferences.

Every student studies hard. Therefore every smart student studies.

Tuesday evening, Jack either went to the movies, played bridge, or studied.

Tuesday evening, Jack played bridge. Therefore, Jack neither went to the movies nor studied Tuesday evening.

In the 1970s and 1980s, researchers became increasingly concerned with knowledge about specific domains in which human experts operate, such as medical diagnosis and the identification of chemical compounds from mass spectrometry data, and also with the other extreme—knowledge about the everyday world that everyone knows, such as the fact that when you tip over a glass of water, the water will spill on the floor.

In the 1980s and 1990s, these concerns focussed on the details of specific sub-domains of everyday knowledge, such as theories of time and space, and also on the general structure of our knowledge of everyday terms, leading to the construction of large and general purpose “ontologies.” For example, the Cyc Project has devoted many staff-years to the organization of a computer-useable representation of all the knowledge that is *not* contained in encyclopedias (Thus the name “Cyc,” from “encyclopedia.”) but is assumed to be already known by people who read them, and Lycos is using such an ontology to organize searches of the World-Wide Web.

1.2 Logic

In the late 1800s and early 1900s, various formal systems were developed by people who hoped, thereby, to turn human reasoning into a kind of calculation. From our perspective, we can now see that what these people were engaged in was research in knowledge representation. The formal systems they developed were systems of logic, a topic which has been studied since the days of Plato and Aristotle.

Logic is the study of correct reasoning. It is not a particular KRR language. Thus, it is not proper to say “We are using (or not using) logic as our KRR language.” There are, indeed, many different logics (*see* (Haack, 1978), (McCawley, 1981), and the various articles on Logic in (Shapiro, 1992) beginning with (Rapaport, 1992)). So KRR research may be regarded as the search for the right logic to use in AI system.

A logic consists of two parts, a language and a method of reasoning. The logical language, in turn, has two aspects, syntax and semantics. Thus, to specify or define a particular logic, one needs to specify three things:

syntax: the specification of a set of atomic symbols, and the grammatical rules for combining them into well-formed expressions;

semantics: the specification of the meaning of the atomic symbols, and the rules for determining the meanings of the well-formed expressions from the meanings of their parts;

proof theory: the specification of a set of rules, called **rules of inference**, which, given an initial collection, called a **proof**, of well-formed expressions, called **axioms**, specify what other well-formed expressions, called **theorems** can be added to the proof.

Chapter 2

Propositional Logic

2.1 Introduction

Propositional logics (sometimes called Sentential Logics) conceptualize domains at, but not below the level of sentences (or propositions).

We will use CarPool World as a simple example domain. In CarPool World, Tom and Betty carpool to work. On any day, either Tom drives Betty or Betty drives Tom. In the former case, Tom is the driver and Betty is the passenger. In the latter case, Betty is the driver and Tom is the passenger.

The finest analysis of CarPool World in Propositional Logic is that there are six sentences:

<i>Betty drives Tom.</i>	<i>Tom drives Betty.</i>
<i>Betty is the driver.</i>	<i>Tom is the driver.</i>
<i>Betty is the passenger.</i>	<i>Tom is the passenger.</i>

2.2 The “Standard” Logic

2.2.1 Syntax

The syntactic expressions of propositional logics consist of **atomic propositions** and nonatomic, **well-formed propositions (wfps)**.

Syntax of Atomic Propositions

- Any letter of the alphabet, e.g.: P
- Any letter of the alphabet with a numeric subscript, e.g.: Q_3
- Any alphanumeric string, e.g.: *Tom is the driver*

Tom is the driver is an atomic proposition.

Syntax of Well-Formed Propositions (WFPs)

1. Every atomic proposition is a wfp.
2. If P is a wfp, then so is $\neg P$.
3. If P and Q are wfps, then so are
 - (a) $(P \wedge Q)$
 - (b) $(P \vee Q)$
 - (c) $(P \Rightarrow Q)$
 - (d) $(P \Leftrightarrow Q)$
4. Nothing else is a wfp.

We will not bother using parentheses when there is no ambiguity, in which case \wedge and \vee will have higher priority than \Rightarrow , which, in turn will have higher priority than \Leftrightarrow . For example, we will write $P \wedge Q \Leftrightarrow \neg P \Rightarrow Q$ instead of $((P \wedge Q) \Leftrightarrow (\neg P \Rightarrow Q))$.

An example wfp in CarPool World is

$$Tom \text{ is the driver} \Leftrightarrow \neg Betty \text{ is the driver}$$

2.2.2 Semantics

To specify the semantics of a propositional logic, we must give the semantics of each atomic proposition and the rules for deriving the semantics of the wfps from their constituent propositions. There are actually two levels of semantics we must specify: **extensional semantics** and **intensional semantics**.

The **extensional semantics** (value or **denotation**) of the expressions of a logic are relative to a particular interpretation, model, or situation. The extensional semantics of CarPool World, for example, are relative to a particular day. The denotation of a proposition is either True or False. If P is an expression of some logic, we will use $\llbracket P \rrbracket$ to mean the denotation of P . If we need to make explicit that we mean the denotation relative to situation S , we will use $\llbracket P \rrbracket_S$.

The **intensional semantics** (or **intension**) of the expressions of a logic are independent of any specific interpretation, model, or situation, but are dependent only on the domain being conceptualized. If P is an expression of some logic, we will use $[P]$ to mean the intension of P . If we need to make explicit that we mean the intension relative to domain D , we will use $[P]_D$. Many formal people consider the intension of an expression to be a function from situations to denotations. For them, $[P]_D(S) = \llbracket P \rrbracket_S$. However, less formally, the intensional semantics of a wfp can be given as a statement in a previously understood language (for example, English) that allows the extensional value to be determined in any specific situation. Intensional semantics are often omitted when a logic is specified, but they shouldn't be.

Intensional Semantics of Atomic Propositions

The intensional semantics of atomic propositions must be specified for each particular propositional logic. For example, the intensional semantics of the atomic propositions of CarPool World are:

$[Betty\ drives\ Tom] =$ Betty drives Tom to work.

$[Tom\ drives\ Betty] =$ Tom drives Betty to work.

$[Betty\ is\ the\ driver] =$ Betty is the driver of the car.

$[Tom\ is\ the\ driver] =$ Tom is the driver of the car.

$[Betty\ is\ the\ passenger] =$ Betty is a passenger in the car.

$[Tom\ is\ the\ passenger] =$ Tom is a passenger in the car.

Note that each atomic proposition is a single indivisible symbol; the fact that the atomic propositions look like English sentences whose meanings are paraphrases of the intensional semantics is purely for mnemonic purposes. One should never rely on “pretend it’s English” semantics.

Intensional Semantics of WFPs

Since the **logical connectives** \neg , \wedge , \vee , \Rightarrow , and \Leftrightarrow are commonly used, the following clauses are the standard ones for deriving the intensional semantics of wfps from the intensional semantics of their constituents:

- $[\neg P] =$ It is not the case that $[P]$.
- $[P \wedge Q] =$ $[P]$ and $[Q]$.
- $[P \vee Q] =$ Either $[P]$ or $[Q]$ or both.
- $[P \Rightarrow Q] =$ If $[P]$ then $[Q]$.
- $[P \Leftrightarrow Q] =$ $[P]$ if and only if $[Q]$.

Extensional Semantics of Atomic Propositions

The denotation of an atomic proposition is a truth value, True or False. Each way of assigning a truth value to each atomic proposition forms one situation. For example, each column of the following table gives one situation of CarPool World.

Proposition	Denotation in Situation				
	1	2	3	4	5
<i>Betty drives Tom</i>	True	True	True	False	False
<i>Tom drives Betty</i>	True	True	False	True	False
<i>Betty is the driver</i>	True	True	True	False	False
<i>Tom is the driver</i>	True	False	False	True	False
<i>Betty is the passenger</i>	True	False	False	True	False
<i>Tom is the passenger</i>	True	False	True	False	False

This shows 5 situations. Since there are 6 propositions, and each one can have either of 2 truth values, there are $2^6 = 64$ different situations in CarPool World. We will see below how to limit these to the two that “make sense.”

Extensional Semantics of WFPs

Just as there is a standard way to derive the intensional semantics of wfps from their constituents, so is there a standard way to compute the denotations of wfps from their constituents. These are:

- $\llbracket \neg P \rrbracket$ is True if $\llbracket P \rrbracket$ is False. Otherwise, it is False.
- $\llbracket P \wedge Q \rrbracket$ is True if $\llbracket P \rrbracket$ is True and $\llbracket Q \rrbracket$ is True. Otherwise, it is False.
- $\llbracket P \vee Q \rrbracket$ is False if $\llbracket P \rrbracket$ is False and $\llbracket Q \rrbracket$ is False. Otherwise, it is True.
- $\llbracket P \Rightarrow Q \rrbracket$ is False if $\llbracket P \rrbracket$ is True and $\llbracket Q \rrbracket$ is False. Otherwise, it is True.
- $\llbracket P \Leftrightarrow Q \rrbracket$ is True if $\llbracket P \rrbracket$ and $\llbracket Q \rrbracket$ are both True, or both False. Otherwise, it is False.

These can also be shown in the following **truth tables**.

P	True	False
$\neg P$	False	True

P	True	True	False	False
Q	True	False	True	False
$P \wedge Q$	True	False	False	False
$P \vee Q$	True	True	True	False
$P \Rightarrow Q$	True	False	True	True
$P \Leftrightarrow Q$	True	False	False	True

Notice that each column of these tables represents a different situation.

Semantic Properties of WFPs

A wfp is either **satisfiable**, **contingent**, **valid**, or **contradictory** according to the situations in which it is True. A wfp is **satisfiable** if it is True in at least one situation, **contingent** if it is True in at least one situation and False in at least one situation, **valid** if it is True in every situation, and **contradictory** if it is False in every situation. For example, as the following table shows, $\neg P$, $Q \Rightarrow P$, and $P \Rightarrow (Q \Rightarrow P)$ are satisfiable, $\neg P$ and $Q \Rightarrow P$ are contingent, $P \Rightarrow (Q \Rightarrow P)$ is valid, and $P \wedge \neg P$ is contradictory.

P	True	True	False	False
Q	True	False	True	False
$\neg P$	False	False	True	True
$Q \Rightarrow P$	True	True	False	True
$P \Rightarrow (Q \Rightarrow P)$	True	True	True	True
$P \wedge \neg P$	False	False	False	False

If A is a well-formed expression of a logic \mathcal{L} , it is standard to write $\models_{\mathcal{L}} A$ (The symbol “ \models ” is called a “double turnstyle”.) to indicate that A is valid in logic \mathcal{L} . The subscript may be omitted if it is clear from context. Thus, the above truth table shows that $\models P \Rightarrow (Q \Rightarrow P)$. Valid wfps are also called **tautologies**.

Related to the notion of validity is the notion of **logical implication**. The set of wfps $\{A_1, \dots, A_n\}$ logically implies the wfp B in logic \mathcal{L} (written $A_1, \dots, A_n \models_{\mathcal{L}} B$) if and only if B is True in every situation in which every A_i is True. This is how domain knowledge can be used to reduce the set of situations to only those that “make sense.” For example, in CarPool World, we want to specify that:

- Betty is the driver or the passenger, but not both:

$$\textit{Betty is the driver} \Leftrightarrow \neg \textit{Betty is the passenger}$$

- Tom is the driver or the passenger, but not both:

$$\textit{Tom is the driver} \Leftrightarrow \neg \textit{Tom is the passenger}$$

- If Betty drives Tom, then Betty is the driver and Tom is the passenger:

$$\textit{Betty drives Tom} \Rightarrow \textit{Betty is the driver} \wedge \textit{Tom is the passenger}$$

- If Tom drives Betty, then Tom is the driver and Betty is the passenger:

$$\textit{Tom drives Betty} \Rightarrow \textit{Tom is the driver} \wedge \textit{Betty is the passenger}$$

- Finally, either Tom drives Betty or Betty drives Tom:

$$\textit{Tom drives Betty} \vee \textit{Betty drives Tom}$$

The following table shows the only two situations of CarPool World (numbered as in the previous table) in which all five of these wfps are True.

Proposition	Denotation in Situation	
	3	4
<i>Betty drives Tom</i>	True	False
<i>Tom drives Betty</i>	False	True
<i>Betty is the driver</i>	True	False
<i>Tom is the driver</i>	False	True
<i>Betty is the passenger</i>	False	True
<i>Tom is the passenger</i>	True	False

Notice that these are precisely the two commonsense situations.

Logical implication and logical validity are related by the following

Metatheorem 1: $A_1, \dots, A_n \models_{\mathcal{L}} B$ if and only if $\models_{\mathcal{L}} A_1 \wedge \dots \wedge A_n \Rightarrow B$

The significance of this is that if one is interested in determining either logical validity or logical implication, one may solve the other problem instead.

2.2.3 Proof Theory

There are two basic varieties of proof theory (or syntactic inference methods) in propositional logics, **Hilbert-style** methods, and **natural deduction** methods. Hilbert-style inference methods use a large number of (**logical**) **axioms** and a small number of **rules of inference**, whereas natural deduction methods use a small number of (logical) axioms (or even none at all) and a large number of rules of inference. Usually there are two rules of inference for each **logical connective**, \neg , \wedge , \vee , \Rightarrow , and \Leftrightarrow , an **introduction rule**, and an **elimination rule**. These are usually abbreviated by writing the logical connective before “I” or “E”, respectively. For example \neg I is the “negation introduction” rule, and \wedge E is the “and elimination” rule. The rule \Rightarrow E is also often called **modus ponens**.

Hilbert-Style Methods

In Hilbert-style methods, a **derivation** of a wfp, A , from a set of assumptions (or **non-logical axioms**.) Γ , is a list of wfps in which each wfp in the list is either a logical axiom, or a non-logical axiom, or follows from previous wfps in the proof according to one of the rules of inference. A Hilbert-style **proof** of a wfp, A , is a derivation of A from an empty set of assumptions. If A can be derived from Γ in the logic \mathcal{L} , we write $\Gamma \vdash_{\mathcal{L}} A$, (The symbol “ \vdash ” is called a “turnstyle”.) while if A can be proved in \mathcal{L} , we write $\vdash_{\mathcal{L}} A$. If A can be proved in \mathcal{L} , A is called a **theorem** of \mathcal{L} .

One set of axioms for the standard propositional calculus is¹:

$$(A1). (A \Rightarrow (B \Rightarrow A))$$

$$(A2). ((A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C)))$$

$$(A3). ((\neg B \Rightarrow \neg A) \Rightarrow ((\neg B \Rightarrow A) \Rightarrow B))$$

Note that these axioms use only the connectives \Rightarrow and \neg . The others may be defined as abbreviations of these, or additional axioms may be given for them. For our purposes, we will not bother about them.

There are two attitudes taken about axioms such as these:

1. They are *axiom schemata*. The actual axioms (There are an infinite number of them.) may be derived from them by taking any one axiom schema, and substituting any wfp whatsoever for all occurrences of each of the letters, A , B , and C . We will take this attitude in the example below.
2. They are actual axioms, and the rule of Substitution, sketched above, is a Rule of Inference.

Whether (A1) – (A3) are axioms or axiom schemata, the (other) Rule of Inference is *modus ponens*, which says that if any wfp A is a line of a proof, and a wfp of the

¹These axioms, and the following example, come from E. Mendelson, *Introduction to Mathematical Logic*, (Princeton: D. Van Nostrand) 1964, pp. 31–32.

form $A \Rightarrow B$ is also a line in the proof, for any wfp B , then B can be added as a line in the proof. This rule of inference can be summarized as

$$\frac{A, A \Rightarrow B}{B}$$

An example proof of the theorem $A \Rightarrow A$ is:

- | | | |
|-----|---|-----------------|
| (1) | $(A \Rightarrow ((A \Rightarrow A) \Rightarrow A)) \Rightarrow ((A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A))$ | Instance of A2 |
| (2) | $A \Rightarrow ((A \Rightarrow A) \Rightarrow A)$ | Instance of A1 |
| (3) | $(A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A)$ | From 1, 2 by MP |
| (4) | $A \Rightarrow (A \Rightarrow A)$ | Instance of A1 |
| (5) | $A \Rightarrow A$ | From 3, 4 by MP |

As a second example, let's derive

\neg *Betty is the passenger*

from

John is the passenger \Rightarrow *Betty is the driver*,
Betty is the driver \Rightarrow \neg *Betty is the passenger*,
 and *John is the passenger*

The derivation is

- | | | |
|-----|---|-----------------|
| (1) | <i>John is the passenger</i> | Assumption |
| (2) | <i>John is the passenger</i> \Rightarrow <i>Betty is the driver</i> | Assumption |
| (3) | <i>Betty is the driver</i> | From 1, 2 by MP |
| (4) | <i>Betty is the driver</i> \Rightarrow \neg <i>Betty is the passenger</i> | Assumption |
| (5) | \neg <i>Betty is the passenger</i> | From 3, 4 by MP |

Notice that “rules” in the sense of “rule-based systems” correspond not to rules of inference, but to non-atomic assumptions. Rules of inference are used by the underlying “inference engine” of rule-based systems. We sometimes call assumptions *non-logical axioms* because they are used in proofs the same way axioms are, but they are not actually axioms of any logical system. The rules of rule-based systems are actually non-logical axioms. Sometimes, they are also called *domain rules* to distinguish them from rules of inference.

2.3 Clause Form Logic

2.3.1 Syntax

2.3.2 Semantics

2.3.3 Proof Theory

Chapter 3

Elementary Predicate Logic

3.1 The “Standard” Logic

3.1.1 Syntax

3.1.2 Semantics

3.1.3 Proof Theory

3.2 Clause Form Logic

3.2.1 Syntax

3.2.2 Semantics

3.2.3 Proof Theory

3.3 SNePS Logic

3.3.1 Syntax

3.3.2 Semantics

3.3.3 Proof Theory

Chapter 4

Full First-Order Predicate Logic

4.1 The “Standard” Logic

4.1.1 Syntax

4.1.2 Semantics

4.1.3 Proof Theory

4.2 Clause Form Logic

4.2.1 Syntax

4.2.2 Semantics

4.2.3 Proof Theory

4.3 SNePS Logic

4.3.1 Syntax

4.3.2 Semantics

4.3.3 Proof Theory

Bibliography

- Haack, S. (1978). *Philosophy of Logics*. Cambridge University Press, New York.
- Lifschitz, V., editor (1990). *Formalizing Common Sense: Papers by John McCarthy*. Ablex Publishing Corporation, Norwood, NJ.
- McCarthy, J. (1959). Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 75–91, London. Her Majesty's Stationery Office. Reprinted in (Lifschitz, 1990, 9–16).
- McCawley, J. D. (1981). *Everything that Linguists have Always Wanted to Know about Logic* *but were ashamed to ask*. The University of Chicago Press, Chicago, IL.
- Rapaport, W. J. (1992). Logic. In Shapiro, S. C., editor, *Encyclopedia of Artificial Intelligence*, pages 851–853. John Wiley & Sons, New York, second edition.
- Shapiro, S. C. (1992). Artificial intelligence. In Shapiro, S. C., editor, *Encyclopedia of Artificial Intelligence*, pages 54–57. John Wiley & Sons, New York, second edition.
- Shapiro, S. C. (2000). Propositional, first-order and higher-order logics: Basic definitions, rules of inference, and examples. In Iwańska, Ł. M. and Shapiro, S. C., editors, *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*, pages 379–395. AAAI Press/The MIT Press, Menlo Park, CA.
- Shapiro, S. C. (2003). Knowledge representation. In Nadel, L., editor, *Encyclopedia of Cognitive Science*, volume 2, pages 671–680. Macmillan Publishers Ltd.