1 Introduction

In a book about the logical foundations of knowledge bases, it is probably a good idea to review if only briefly how concepts like knowledge, representation, reasoning, knowledge bases, and so on are understood informally within Artificial Intelligence (AI), and why so many researchers feel that these notions are important to the AI enterprise.

1.1 Knowledge

Much of AI does indeed seem to be concerned with *knowledge*. There is knowledge representation, knowledge acquisition, knowledge engineering, knowledge bases and knowledge-based systems of various sorts. In the early eighties, during the heyday of commercial AI, there was even a slogan "Knowledge is power" used in advertisements. So what exactly is knowledge that people in AI should care so much about it? This is surely a philosophical issue, and the purpose of this chapter is not to cover in any detail what philosophers, logicians, and computer scientists have said about knowledge over the years, but only to glance at some of the issues involved and especially their bearings on AI.

1.1.1 Propositions

To get a rough sense of what knowledge is supposed to be, at least outside of AI, it is useful to look at how we talk about it informally. First, observe that when we say something like "John knows that ...," we fill in the blank with a simple *declarative sentence*. So we might say that "John knows that Mary will come to the party" or that "John knows that dinosaurs were warm blooded." This suggests that, among other things, knowledge is a relation between a knower (like John) and a *proposition*, that is, the idea expressed by a simple declarative sentence (like "Mary will come to the party").

Part of the mystery surrounding knowledge is due to the abstract nature of propositions. What can we say about them? As far as we are concerned, what matters about propositions is that they are abstract entities that can be *true* or *false*, right or wrong.[1] When we say that "John knows that p," we can just as well say that "John knows that it is true that p." Either way, to say that somebody knows something is to say that somebody has formed a judgement of some sort, and has come to realize that the world is one way and not another. In talking about this judgement, we use propositions to classify the two cases.

1 Strictly speaking, we might want to say that the *sentences* expressing the proposition are true or false, and that the propositions themselves are either factual or non-factual. Further, because of linguistic features such as indexicals (that is, words like "me" and "yesterday"), we more accurately say that it is actual tokens of sentences or their uses in contexts that are true or false, not the sentences themselves.

A similar story can be told about a sentence like "John hopes that Mary will come to the party." The same proposition is involved, but the relationship John has to it is different. Verbs like "knows," "hopes," "regrets," "fears," and "doubts" all denote *propositional attitudes*, relationships between agents and propositions. In all cases, what matters about the proposition is its truth: if John hopes that Mary will come to the party, then John is hoping that the world is one way and not another, as classified by the proposition.

Of course, there are sentences involving knowledge that do not mention a proposition. When we say "John knows who Mary is taking to the party," or "John knows how to get there," we can at least imagine the implicit propositions: "John knows that Mary is taking so-and-so to the party," or "John knows that to get to the party, you go two blocks past Main Street, turn left," and so on. On the other hand, when we say that John has a skill as in "John knows how to play piano," or a deep understanding of someone or something as in "John knows Bill well," it is not so clear that any useful proposition is involved. We will have nothing further to say about this latter form of knowledge in the book.

1.1.2 Belief

A related notion that we are concerned about, however, is the concept of *belief*. The sentence "John believes that p" is clearly related to "John knows that p." We use the former when we do not wish to claim that John's judgement about the world is necessarily accurate or held for appropriate reasons. We sometimes use it when we feel that John might not be completely convinced. In fact, we have a full range of propositional attitudes, expressed by sentences like "John is absolutely certain that p," "John is confident that p," "John is of the opinion that p," "John suspects that p," and so on, that differ only in the level of conviction they attribute. For now, we will not distinguish among *any* of them.[2] What matters is that they all share with knowledge a very basic idea: John takes the world to be one way and not another.

So when we talk about knowledge or any other propositional attitude, we are implicitly imagining a number of different ways the world could be. In some of these, Mary comes to the party; in others, she does not. When we say that John knows or believes or suspects that Mary will come to the party, we are saying that John takes it (with varying degrees of conviction) that those where Mary does not come to the party are fantasy only; they do not correspond to reality.

In this very abstract and informal picture, we can already see emerging two very different but related views of knowledge or belief. First, we can think of knowledge (or belief) as a collection of propositions held by an agent to be true. Second, we can think in terms

2 One way to understand (subjective) probability theory is as an attempt to deal in a principled way with these levels of conviction as numeric degrees of belief. This is the last we will say on this subject in the book.

different possible ways the world could be, and knowledge (or belief) as a classification of these into two groups, those that are considered incorrect, and those that are candidates for the way the world really is.

1.1.3 Representation

The interest of AI in knowledge is obviously that we want to design and build systems that know a lot about their world, enough, in fact, that they do not act unintelligently.[3] But there is more to it. Any system, AI-based or not, can be said to have knowledge about its world. Any Java compiler, for example, knows a lot about the details of the Java language. There's even the joke about a thermos "knowing" whether the liquid it contains is hot or cold, and making sure it preserves the correct one. This idea of attributing knowledge to a more-or-less complex system (or person) is what the philosopher Dennett calls "taking the intentional stance." But when people in AI talk about knowledge bases, knowledge engineering and so on, they mean more than this. They have in mind a system that not only knows a lot in the above sense, but also a system that does what it does using a representation of that knowledge.

The concept of representation is no doubt as philosophically problematic as that of knowledge. Very roughly speaking, *representation* is a relationship between two domains where the first is meant to "stand for" or take the place of the second. Usually, the first domain, the representer, is more concrete, immediate, or accessible in some way than the second. The type of representer that we will be most concerned with here is that of a formal *symbol*, that is, a character or group of them taken from some predetermined alphabet. The digit "7," for example, stands for the number 7, as does the group of letters "VII," and in other contexts, the words "sept," "sieben," and "shichi." As with all representation, it is assumed to be easier to deal with symbols (recognize them, distinguish them from each other, display them *etc*.) than with what the symbols represent. In some cases, a word like "John" might stand for something quite concrete; but many words, like "love" or "truth," stand for abstractions.

Of special concern to us is when a group of formal symbols stands for a proposition: "John loves Mary" stands for the proposition that John loves Mary. Again, the symbolic English sentence is concrete: it has distinguishable parts involving the 3 words, for example, and a recognizable syntax. The proposition, on the other hand, is abstract: it is something like a classification of the ways the world can be into two groups: those where John loves Mary, and those where he does not.

Knowledge Representation, then, is this: it is the field of study within AI concerned

3 What we call "commonsense" clearly involves considerable knowledge of a variety of sorts, at least in the sense of being able to form a judgement about different ways the world could be.

with using formal symbols to represent a collection of propositions believed by some putative agent. As we will see however, we would not want to insist that there be symbols to represent *each* of the propositions believed by the agent. There may very well be an infinite number of propositions believed, only a finite number of which are ever represented. It will be the role of reasoning to bridge the gap between what is represented and the full set of propositions believed.

1.1.4 Reasoning

So what is *reasoning*? In general, it is the formal manipulation of the symbols representing a collection of believed propositions to produce representations of new ones. It is here that we use the fact that symbols are more accessible than the propositions they represent: they must be concrete enough that we can manipulate them (move them around, take them apart, copy them, string them together) in such a way as to construct representations of new propositions.

The analogy here is with arithmetic. We can think of binary addition as being a certain formal manipulation: we start with symbols like "1011" and "10," for instance, and end up with "1101." The manipulation here is addition since the final symbol represents the sum of the numbers represented by the initial ones. Reasoning is similar: we might start with the sentences "John loves Mary" and "Mary is coming to the party," and after a certain amount of manipulation produce the sentence "Someone John loves is coming to the party." We would call this form of reasoning logical inference because the final sentence represents a logical entailment of the propositions represented by the initial ones. According to this view (first put forward, incidentally, by the philosopher Leibniz in the 17th century), reasoning is a form of calculation, not unlike arithmetic, but over symbols standing for propositions rather than numbers.

1.2 Why knowledge representation and reasoning?

Let's talk motivation: why do people in AI who want their systems to know a lot, also want their systems to represent that knowledge symbolically? The intentional stance above says nothing about what is or is not represented within a system. We can say that a system knows that p without claiming that there is anything represented within the system corresponding to that proposition. The hypothesis underlying much (but not all) of the work in AI, however, is that we want to construct systems that do contain symbolic representations with two important properties. First is that we (from the outside) can understand them as standing for propositions. Second is that the system is designed to behave the way that it does *because* of these symbolic representations. This is what Brian Smith has called the

Knowledge Representation Hypothesis:

> Any mechanically embodied intelligent process will be comprised of structural ingredients that a) we as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and b) independent of such external semantic attribution, play a formal but causal and essential role in engendering the behaviour that manifests that knowledge.

In other words, the Knowledge Representation Hypothesis is that we will want to construct systems for which the intentional stance is grounded by design in symbolic representations. A system of this sort is called a *knowledge-based system* and the symbolic representation involved its *knowledge base* (or KB).

1.2.1 Knowledge-based systems

To see what a knowledge-based system amounts to, it is helpful to look at two very simple Prolog programs with identical behaviour.[4] The first is:

```
printColour(snow) :- !, write("It's white.").
printColour(grass) :- !, write("It's green.").
printColour(sky) :- !, write("It's yellow.").
printColour(X) :- write("Beats me.").
```

The second is:

```
printColour(X) :- colour(X,Y), !,
       write("It's "), write(Y), write(".").
printColour(X) :- write("Beats me.").
colour(snow,white).
colour(sky,yellow).
colour(X,Y) :- madeof(X,Z), colour(Z,Y).
madeof(grass,vegetation).
colour(vegetation,green).
```

Observe that both programs are able to print out the colour of various items (getting the sky wrong, as it turns out). Taking an intentional stance, both might be said to "know" that the colour of snow is white. The crucial point, however, is that only the second program is designed according to the Knowledge Representation Hypothesis.

Consider the clause `colour(snow,white)`, for example. This is a symbolic structure that we can understand as representing the proposition that snow is white, and moreover,

4 No further knowledge of Prolog is assumed beyond this motivating example.

we know, by virtue of knowing how the Prolog interpreter works, that the system prints out the appropriate colour of snow precisely *because* it bumps into this clause at just the right time. Remove the clause and the system would no longer do so.

There is no such clause in the first program. The one that comes closest is the first clause of the program which says what to print when asked about snow. But we would be hard-pressed to say that this clause literally represents a belief, except perhaps a belief about what ought to be written.

So what makes a system knowledge-based, as far as we are concerned, is not the use of a logical formalism (like Prolog), or the fact that it is complex enough to merit an intentional description involving knowledge, or the fact that what it believes is true; rather it is the presence of a KB, a collection of symbolic structures representing what it believes and reasons with during the operation of the system.

1.2.2 Why knowledge representation?

So an obvious question arises when we start thinking about the two Prolog programs of the previous section: what advantage, if any, does the knowledge-based one have? Would it not be better to "compile out" the KB and distribute this knowledge to the procedures that need it, as we did in the first program? The performance of the system would certainly be better. It can only slow a system down to have to look up facts in a KB and reason with them at runtime in order to decide what actions to take. Indeed advocates within AI of so-called "procedural knowledge" take pretty much this point of view.

When we think about the various skills we have, such as riding a bicycle or playing a piano, it certainly *feels* like we do not reason about the various actions to take (shifting our weight or moving our fingers); it seems much more like we just know what to do, and do it. In fact, if we try to think about what we are doing, we end up making a mess of it. Perhaps (the argument goes) this applies to most of our activities, making a meal, getting a job, staying alive, and so on.

Of course, when we first learn these skills, the case is not so clear: it seems like we need to think deliberately about what we are doing, even riding a bicycle. The philosopher Hubert Dreyfus first observed this paradox of "expert systems." These systems are claimed to be superior precisely because they are knowledge-based, that is, they reason over explicitly represented knowledge. But novices are the ones who think and reason, claims Dreyfus. Experts do not; they learn to recognize and to react. The difference between a chess master and a chess novice is that the novice needs to figure out what is happening and what to do, but the master just "sees" it. For this reason (among others), Dreyfus believes that the development of knowledge-based systems is completely wrong-headed, if it is attempting to duplicate human-level expertise.

So why even consider knowledge-based systems? Unfortunately, no definitive answer

can yet be given. We suspect, however, that the answer will emerge in our desire to build systems that deal with a set of tasks that is *open-ended*. For any fixed set of tasks, it might work to "compile out" what the system needs to know; but if the set of tasks is not determined in advance, the strategy will not work. The ability to make behaviour depend on explicitly represented knowledge seems to pay off when we cannot specify in advance how that knowledge will ever be used.

The best example of this, perhaps, is what happens when we read a book. Suppose we are reading about South American geography. When we find out for the first time that approximately half of the population of Peru lives in the Andes, we are in no position to distribute this piece of knowledge to the various routines that might eventually require it. Instead, it seems pretty clear that we are able to assimilate the fact in declarative form for a very wide variety of potential uses. This is the prototypical case of a knowledge-based system.

From a system design point of view, the knowledge-based approach seems to have a number of desirable features:

- We can add new tasks and easily make them depend on previous knowledge. In our Prolog program example, we can add the task of enumerating all objects of a given colour, or even of painting a picture, by making use of the KB to determine the colours.
- We can extend the existing behaviour by adding new beliefs. For example, by adding a clause saying that canaries are yellow, we automatically propagate this information to any routine that needs it.
- We can debug faulty behaviour by locating the erroneous belief of the system. In the Prolog example, by changing the clause for the colour of the sky, we automatically correct any routine that uses colour information.
- We can concisely explain and justify the behaviour of the system. Why did the program say that grass was green? It was because it believed that grass is a form of vegetation and that vegetation is green. Moreover, we are justified in saying "because" here since if we removed either of the two relevant clauses, the behaviour would indeed change.

Overall, then, the hallmark of a knowledge-based system is that by design it has the ability to be *told* facts about its world and adjust its behaviour correspondingly. We will take this up again below.

This ability to have some of our actions depend on what we believe is what the cognitive scientist Zenon Pylyshyn has called *cognitive penetrability*. Consider, for example, responding to a fire alarm. The normal response is to get up and leave the building. But we would not do so if we happened to believe that the alarm was being tested, say. There are any number of ways we might come to this belief, but they all lead to the same effect. So our response to a fire alarm is cognitively penetrable since it is conditioned on what we

can be made to believe. On the other hand, something like a blinking reflex as an object approaches your eye does not appear to be cognitively penetrable: even if you strongly believe the object will not touch you, you still blink.

1.2.3 Why reasoning?

To see the motivation behind reasoning in a knowledge-based system, it suffices to observe that we would like action to depend on what the system believes about the world, as opposed to *just* what the system has explicitly represented. In the Prolog example, there was no clause representing the belief that the colour of grass was green, but we still wanted the system to know this. In general, much of what we expect to put in a KB will involve quite general facts, which will then need to be applied to particular situations.

For example, we might represent the following two facts explicitly:

1. Patient x is allergic to medication m.
2. Anyone allergic to medication m is also allergic to medication m'.

In trying to decide if it is appropriate to prescribe medication m' for patient x, neither represented fact answers the question. Together, however, they paint a picture of a world where x is allergic to m', and this, together with other represented facts about allergies, might be sufficient to rule out the medication. So we do not want to condition behaviour only on the represented facts that we are able to *retrieve*, like in a database system. The beliefs of the system must go beyond these.

But beyond them to where? There is, as it turns out, a simple answer to this question, but one that we will argue in later chapters is too simplistic. The simple answer: the system should believe p if, according to the beliefs it has represented, the world it is imagining is one where p is true. In the above example, facts (1) and (2) are both represented. If we now imagine what the world would be like if (1) and (2) were both true, then this is a world where

3. Patient x is allergic to medication m'

is also true, even though this fact is only implicitly represented.

This is the concept of *entailment*: we say that the propositions represented by a set of sentences S entail the proposition represented by a sentence p when the truth of p is implicit in the truth of the sentences in S. In other words, if the world is such that every element of S comes out true, then p does as well. All that we require to get some notion of entailment is a language with an account of what it means for a sentence to be true or false. As we argued, if our representation language is to represent knowledge at all, it must come with such an account (again: to know p is to take p to be true). So any knowledge representation language, whatever other features it may have, whatever syntactic form it may take, whatever reasoning procedures we may define over it, ought to have a well-

defined notion of entailment.

The simple answer to what beliefs a knowledge-based system should exhibit, then, is that it should believe all and only the entailments of what it has explicitly represented. The job of reasoning, then, according to this account, is to compute the entailments of the KB.

What makes this account simplistic is that there are often quite good reasons not to calculate entailments. For one thing, it is too *difficult* computationally to decide if a sentence is entailed by the kind of KB we will want to use. Any procedure that gives us answers in a reasonable amount of time will occasionally either miss some entailments or return too many. In the former case, the reasoning process is said to be *incomplete*; in the latter case, the reasoning is said to be *unsound*.

But there are also conceptual reasons why we might consider unsound or incomplete reasoning. For example, suppose p is not entailed by a KB, but is a reasonable guess, given what is represented. We might still want to believe that p is true. To use a classic example, suppose all I know about an individual Tweety is that she is a bird. I might have a number of facts about birds in the KB, but likely they would not *entail* that Tweety flies. After all, Tweety might turn out to be an ostrich. Nonetheless, it is a reasonable assumption that Tweety flies. This is unsound reasoning since we can imagine a world where everything in the KB is true but where Tweety does not fly.

As another example, a knowledge-based system might come to believe a collection facts from various sources which, taken together, cannot all be true. In this case, it would be inappropriate to do logically complete reasoning, since *every* sentence would then be believed. This is because for any sentence p, any world where all the sentences in the set are true is one where p is also true, since there are no such worlds. An incomplete form of reasoning would clearly be more useful here until the contradictions are dealt with, if ever.

But despite all this, it remains the case that the simplistic answer is by far the best starting point for thinking about reasoning, even if we intend to diverge from it. So while it would be a mistake to *identify* reasoning in a knowledge-based system with logically sound and complete inference, it is the right place to begin.

1.3 Knowledge representation systems

The picture of a knowledge-based system that emerges from the above discussion is one where a system performs some problem-solving activity such as deciding what medicine to prescribe, and does so intelligently by appealing at various points to what it knows: is patient x allergic to medication m? what else is x allergic to? The mechanism used by the system to answer such questions involves reasoning from a stored KB of facts about the world. It makes sense in this scenario to separate the management of the KB from the rest

of the system. The data structures within a KB and the reasoning algorithms used are not really of concern to the problem-solving system. Ultimately, what a medical system needs to find out is whether or not x is allergic to m (and perhaps how certain we are of that fact), not whether or not a certain symbolic structure occurs somewhere or can be processed in a certain way.

We take the view that it is the role of a *knowledge representation system* to manage the KB within a larger knowledge-based system. Its job is to make various sorts of information about the world available to the rest of the system based on what information it has obtained perhaps from other parts of the system and whatever reasoning it can perform.[5] So its job is smaller than that of a full knowledge-based problem solver, but larger than that of a database management system which would merely retrieve the contents of the KB. According to this view, the contents of the KB and the reasoning algorithms used by the knowledge representation system are its own business; what the rest of knowledge-based problem solver gets to find out is just what is and is not known about the world.

1.3.1 The knowledge and symbol levels

Allen Newell suggested that we can look at the knowledge in a knowledge-based system in at least two ways. At the *knowledge level*, we imagine a knowledge-based system as being in some sort of abstract epistemic state. It acquires knowledge over time, moving from state to state, and uses what it knows to carry out its activities and achieve its goals. At the *symbol level*, we also imagine that within the system somewhere there is a symbolic KB representing what the system knows, as well as reasoning procedures that make what is known available to the rest of the system. In our terms, the symbol level looks at knowledge from *within* a knowledge representation system where we deal with symbolic representations explicitly; the knowledge level looks at knowledge from *outside* the knowledge representation system, and is only concerned with what is true in the world according to this knowledge. So at the knowledge level, we are concerned with the logic of *what* a system knows; at the symbol level, within a knowledge representation system, we are concerned with *how* a system does it.

There are clearly issues of adequacy at each level. At the knowledge level, we deal with the expressive adequacy of a representation language, the characteristics of its entailment relation, including its computational complexity; at the symbol level, we ask questions about the computational architecture, the properties of the data structures and algorithms, including their algorithmic complexity.

This is similar in many ways to the specification/implementation distinction within

5 In this most general picture, we include the possibility of a knowledge representation system *learning* from what it has observed, as well as it having various levels of confidence in what it believes.

traditional computer science. The symbol level provides an implementation for the more abstract knowledge level specification. But what exactly does a knowledge level specify? What would a symbol level need to implement? In a sense, being precise about these is the topic of this book.

1.3.2 A functional view: TELL and ASK

We said that the role of a knowledge representation system was to make information available to the rest of the system based on what it had acquired and what reasoning it could perform. In other words, we imagine that there are two main classes of operations that a knowledge representation system needs to implement for the rest of the system: operations that absorb new information as it becomes available, and operations that provide information to the rest of the system as needed. In its simplest form, a knowledge representation system is passive: it is up to the rest of the system to *tell* it when there is something that should be remembered, and to *ask* it when it needs to know something. It is up to the knowledge representation system to decide what to do with what it is told, and in particular, how and when to reason so as to provide answers to questions as requested.

For a large section of the book, we will be concerned with a very simple instance of each of these operations: a **TELL** operation and an **ASK** operation each of which take as argument a sentence about the world. The idea is that the **TELL** operation informs the knowledge representation system that the sentence in question is true; the **ASK** operation asks the system whether the sentence in question is true. We can see immediately that any realistic knowledge representation system would need to do much more. At the very least, it should be possible to ask *who* or *what* satisfies a certain property (according to what is known). We will examine operations like these later; for now, we stick to the simple version.

So the idea at the knowledge level is that starting in some state of knowledge, the system can be told certain sentences and move through a sequence of states; at any point, the system believes the world is in a certain state, and can be asked if a certain sentence is true. In subsequent chapters, we will show how these two operations can be defined precisely but in a way that leaves open how they might be implemented. We will also discuss simple implementation techniques at the symbol level based on automated theorem-proving, and be able to prove that such implementations are correct with respect to the specification.

1.3.3 The interaction language

With this functional view of knowledge representation, we can see immediately that there is a difference at least conceptually between the *interaction language*, that is, the language used to tell the system or to ask it questions about the world, and the *representation lan-*

guage, the collection of symbolic structures used at the symbol level to represent what is known. There is no reason to suppose the two languages are identical, or even that what is stored constitutes a declarative language of any sort. Moreover, there are clear intuitive cases where simply storing what you have been told would be a bad thing to do.

Consider indexicals, for example, that is, terms like "I," "you," "here," and "now," that might appear in an interaction language. If a fact about the world you are told is that "*There is a treasure buried here*" for instance, it would be a bad idea to absorb this information by storing the sentence verbatim. Two weeks from now, when you decide to go looking for the treasure, it is likely no longer true that it is located "here." You need to resolve the "here" at the time you are told the fact into a description of a location that can be used in different contexts. If later the question "*Is there a treasure here?*" is asked, we would want to resolve the "here" differently. We need to distinguish between how information is communicated to or retrieved from the system and how it is represented for long-term storage.

In this book, we will not emphasize indexicals like those above (although they are mentioned in an exercise). There is an important type of indexical that we *will* want to examine in considerable detail, however, and that is one that refers to the current state of knowledge.

Suppose, for example, we have a system that is attempting to solve a murder mystery, and that all it knows so far is that Tom and Dick were at the scene of the crime (and perhaps others). If the system is told that "*The murder was not committed by anyone you currently know to have been present*," the system learns that the murderer was neither Tom nor Dick. It is this "currently" that makes the expression indexical. As the knowledge of the system changes, so will what this expression refers to, just as "here" did. Suppose the system later finds out that Harry was also at the scene and was in fact the murderer. If it is now asked "*Was the murder committed by someone you currently know to have been present?*" the correct answer is *yes*, despite what it was told before. As we will see in Chapter 3, it is extremely useful to imagine an interaction language that can use indexicals like these to request information or provide new information. But it will require us to distinguish clearly between an interaction language and any language used at the symbol level to represent facts for later use.

1.4 The rest of the book

The remaining chapters of the book are divided into two broad sections: Chapters 2 to 8 cover the basics in sequential order; Chapters 9 to 14 cover advanced research-oriented topics.

- In Chapter 2, we start with a simple interaction language, a dialect of the language of first-order logic (with which we assume familiarity). However, there are good reasons to insist on some specific representational features, such as standard names and a special treatment of equality. With these, the semantic specification of the language ends up being clearer and more manageable than classical accounts. This will be especially significant when we incorporate epistemic features.
- In Chapter 3, we extend the first-order interaction language to include an epistemic operator, resulting in a language we call $\mathcal{KL}$. This involves being clear about what we mean by an epistemic state, distinct from a world state. This will allow us, among other things, to distinguish between questions about the world (e.g. the birds that do not fly) and questions about what is known (e.g. the birds that are known not to fly).
- Since what we mean by "knowledge" is so crucial to the enterprise here, in Chapter 4, we examine properties of knowledge in detail as reflected in the semantics of the language $\mathcal{KL}$. Among other things, we examine the interplay between quantifiers and knowledge, as well as the status of knowledge about knowledge.
- In Chapter 5, we define the **TELL** and **ASK** operations for the interaction language $\mathcal{KL}$. This provides a clear knowledge-level specification of the service to be provided by a knowledge representation system. We also include in this chapter a detailed example of the kind of questions and assertions that can be handled by our definition.
- In Chapter 6, we examine the relationship between the two views of knowledge mentioned above: knowledge in a KB, and knowledge in an abstract epistemic state. In other words, we look at the relationship between the symbol-level and knowledge-level views of knowledge. As it turns out, the correspondence between the two, as required by the semantics of the language $\mathcal{KL}$, is not exact.
- In Chapter 7, we prove that, despite the results of Chapter 6, it is possible to produce a symbol-level implementation of the interaction operations **TELL** and **ASK**, based on ordinary first-order reasoning. In particular, we show that the result of a **TELL** operation on a finitely represented state can itself always be properly represented, even if the sentence contains (indexical) references to what is currently known.
- In Chapter 8, we introduce a new concept called only-knowing that captures in a purely logical setting what is behind the **TELL** and **ASK** operations. The idea is to formalize using a new epistemic operator the assertion that a sentence is not only known, but all that is known.
- In Chapter 9, we relate only-knowing to Autoepistemic Logic, a special brand of so-called nonmonotonic logic which has been studied extensively in the literature. We are able to fully reconstruct Autoepistemic Logic using only-knowing and, in addition, extend it since we are using a more expressive language.

- In Chapter 10, we consider a proof theory for the logic of only-knowing. We show soundness and completeness in the propositional case and discuss why it is incomplete in the first-order case. Nevertheless, the axiom system allows us to obtain nontrivial first-order derivations using examples from Autoepistemic Logic.
- In Chapter 11, motivated by the fact that only-knowing is not that interesting when used as part of a query, we introduce a more focussed version, namely that of only-knowing something *about* a certain subject matter. We study, in a propositional setting, how **ASK** and **TELL** need to be adapted, how the new concept relates to Autoepistemic Logic, and how it gives rise to certain forms of reasoning about relevance.
- In Chapter 12, we begin our excursion into limited belief and a more tractable form of reasoning. Here we lay out the basic concepts of our approach, first in the propositional case and then moving to first-order. The main idea is to switch from a traditional two-valued semantics to a four-valued one. To keep matters as simple as possible, nested beliefs are first ignored altogether.
- In Chapter 13, we drop all these restrictions. We give new definitions of **TELL** and **ASK** based on the four-valued semantics and provide symbol-level implementations for them in a way quite similar to Chapter 7. Most importantly, we show that in many cases these routines now become tractable or at least decidable.
- In Chapter 14, we consider knowledge bases for reasoning about knowledge and action. For that, we amalgamate the logic of only-knowing with the situation calculus, a popular formalism in AI for reasoning about action. Besides a semantics which naturally extends the one we have developed for only-knowing, we provide a sound and complete second-order proof theory.

There are different ways of approaching this book. The first eight chapters are the core, but the remaining ones can be read more or less independently of each other. Those interested in default reasoning or nonmonotonic reasoning should read Chapter 9; those interested in proof systems and questions involving the logic of only-knowing should read Chapter 10; those interested in the issue of relevance and how it applies to default reasoning should read Chapter 11; those interested in tractable logical reasoning and the problem of logical omniscience should read Chapters 12 and 13 (in that order); finally, those interested in how knowledge relates to action (including perceptual action), for robotic applications, for example, should read Chapter 14.

1.5 Bibliographic notes

Much of the material in this chapter is shared with a forthcoming textbook on knowledge representation [10]. For a collection of readings on knowledge representation, see [9]. For

a more philosophical discussion on knowledge and belief see [45, 12, 38], and [41] on the difference between the two. The notes at the end of Chapters 3 and 4 discuss attempts to formalize these notions. A general discussion of propositions, declarative sentences, and sentence tokens, as bearers of truth values, including the role played by indexicals, can be found in [4]. The connection between knowledge and commonsense is discussed in [101], one of the first papers on AI. The intentional stance is presented in [21], and critically examined in [22]. On Leibniz' views about thinking as a form of calculation see, for example, [28], vol. 3, p. 422. The Knowledge Representation Hypothesis is from Brian Smith's doctoral thesis [134], the Prologue of which appears in [9]. Procedural representations of knowledge are discussed in [143], and the criticism of AI by Hubert Dreyfus can be found in [26]. Zenon Pylyshyn discusses cognitive penetrability in [117], making a strong case for propositional representations to account for human-level competence. For Newell's knowledge and symbol levels, as well as the **TELL** and **ASK** functional interface, see the notes in Chapters 5 and 6. For general references on logic and entailment, see the notes in Chapter 2. Why reasoning needs to diverge from logic is discussed in [17] and [84]. For a review of the research in knowledge representation and reasoning in terms of this divergence, see [83]. For references on default (and logically unsound) reasoning, see the notes in Chapter 9.

1.6 Exercises

1. Consider a task requiring knowledge like baking a cake. Examine a recipe and state what needs to be known to follow the recipe.
2. In considering the distinction between knowledge and belief in this book, we take the view that belief is fundamental, and that knowledge is simply belief where the outside world happens to be cooperating (the belief is true, is arrived at by appropriate means, is held for the right reasons, and so on.). Describe an interpretation of the terms where knowledge is taken to be basic, and belief is understood in terms of it.
3. Explain in what sense reacting to a loud noise is and is not cognitively penetrable.
4. It has become fashionable to attempt to achieve intelligent behaviour in AI systems without using propositional representations. Speculate on what such a system should do when reading a book on South American geography.
5. Describe some ways in which the first-hand knowledge we have of some topic goes beyond what we are able to write down in a language. What accounts for our inability to express this knowledge?