Introduction

Computational Logic has been developed in Artificial Intelligence over the past 50 years or so, in an attempt to program computers to display human levels of intelligence. It is based on Symbolic Logic, in which sentences are represented by symbols and reasoning is performed by manipulating symbols, like solving equations in algebra. However, attempts to use Symbolic Logic to solve practical problems by means of computers have led to many simplifications and enhancements. The resulting Computational Logic is not only more powerful for use by computers, but also more useful for the original purpose of logic, to improve human thinking.

Traditional Logic, Symbolic Logic and Computational Logic are all concerned with the abstract form of sentences and how their form affects the correctness of arguments. Although Traditional Logic goes back to Aristotle in the fourth century B.C., Symbolic Logic began primarily in the nineteenth century, with the mathematical forms of logic developed by George Boole and Gottlob Frege. It was enhanced considerably in the twentieth century by the work of Bertrand Russell, Alfred North Whitehead, Kurt Gödel and many others on its application to the Foundations of Mathematics. Computational Logic emerged in the latter half of the twentieth century, starting with attempts to mechanise the generation of proofs in mathematics, and was extended both to represent more general kinds of knowledge and to perform more general kinds of problem solving. The variety of Computational Logic presented in this book owes much to the contributions of John McCarthy and John Alan Robinson.

The achievements of Symbolic Logic in the past century have been considerable. But they have resulted in mainstream logic becoming a branch of mathematics and losing touch with its roots in human reasoning. Computational Logic also employs mathematical notation, which facilitates its computer implementation, but obscures its relevance to human thinking.

In this book, I will attempt to show that the practical benefits of Computational Logic are not limited to mathematics and Artificial Intelligence, but can also be enjoyed by ordinary people in everyday life, without the use of mathematical notation. Nonetheless, I include several additional, more technical chapters at the end of the book, which can safely be omitted by the casual reader.

The relationship between logic and thinking

Logic in all its varieties is concerned with formalising the laws of thought. Along with related fields such as Law and Management Science, it focuses on the formulation of *normative* theories, which prescribe how people ought to think. Cognitive Psychology is also concerned with thinking, but it focuses almost exclusively on *descriptive* theories, which study how people actually think in practice, whether correctly or not. For the most part, the two kinds of theories have been developed in isolation, and bear little relationship to one another.

However, in recent years, cognitive psychologists have developed *dual process* theories, which can be understood as combining descriptive and normative theories. Viewed from the perspective of dual process theories, traditional descriptive theories focus on *intuitive thinking*, which is associative, automatic, parallel and subconscious. Traditional normative theories, on the other hand, focus on *deliberative thinking*, which is rule-based, effortful, serial and conscious. In this book, I will argue that Computational Logic is a dual process theory, in which intuitive and deliberative thinking are combined.

But logic is concerned, not only with thinking in the abstract, but with thoughts represented in the form of sentences and with thinking treated as manipulating sentences to generate new thoughts. In Computational Logic, these logical manipulations of sentences also have a computational interpretation. Viewed in this way, Computational Logic can be regarded as a formalisation of the language of human thought.

Computational Logic and the language of thought

As used in Artificial Intelligence, Computational Logic functions first and foremost as an intelligent agent's *language of thought*. It includes a *syntax* (or grammar), which determines the *form* of the agent's thoughts, a *semantics*, which determines the *contents* (or meaning) of those thoughts, and an *inference engine* (or proof procedure), which generates (or derives or infers) new thoughts as consequences of existing thoughts. In this role, Computational Logic can be regarded as a *private language*, representing the agent's goals and beliefs, and helping the agent to regulate its behaviour. This private language is independent of, and more fundamental than, ordinary, natural languages like English.

However, in *multi-agent systems* in Artificial Intelligence, the private language of an individual agent also serves the secondary function of representing the meanings of its communications with other agents. These communications are expressed in a shared *public language*, which may differ from the private languages of individual agents. The task of a communicating agent is to translate thoughts from its private language into the public language, in such a way that the receiving agent can readily translate those public communications into appropriate thoughts in its own private language.

It would be easier if all agents shared the same private language, and if that private language were identical to the public language of the community of agents. This can be arranged by design in an artificial multi-agent system, but it can only be approximated in a society of human agents.

The distinction between private and public languages, which is so clear cut in Artificial Intelligence, has been proposed in the Philosophy of Language to explain the relationship between human thinking and communication. Many of these proposals, which for simplicity can be lumped together as "language of thought" (LOT) proposals, maintain that much human thinking can be understood as taking place in a language of thought. The most famous proposal along these lines is Fodor's hypothesis that the LOT is a private language, which is independent of the Babel of public languages (Fodor, 1975). Other proposals, notably Carruthers (2004), argue that a person's LOT is specific to the public language of the person's social community.

No matter where they stand on the relationship between private and public languages, most proposals seem to agree that the LOT has some kind of logical form. However, for the most part these proposals are remarkably shy about the details of that logical form. By comparison, the proposal that I present in this book – that Computational Logic can be regarded as a formalisation of the LOT – is shamelessly revealing. I draw the main support for my argument from the uses of Computational Logic in Artificial Intelligence. But I also draw support from the relationship between Computational Logic and normative theories of human communication.

Computational Logic and human communication

Much of the time, when we speak or write, we simply express ourselves in public, without making a conscious effort to communicate effectively. But when it really matters that we are understood – like when I am writing this book – we try to be as clear, coherent and convincing as possible. The difference is like the difference between descriptive and normative theories of thinking; and, as in the case of the two kinds of thinking, the two kinds of communication are studied mainly in different academic disciplines. Whereas linguistics is concerned with developing descriptive theories about how people use language in practice, rhetoric and allied disciplines such as English composition and critical thinking are concerned with *normative* theories about how people should use language to communicate more effectively.

In this book, I present a normative theory of intelligent thinking, communication and behaviour. But I pay attention to descriptive theories, because descriptive theories help us to understand where we are coming from, whereas normative theories show us where we are aiming to go.

The descriptive theory of communication that comes closest to a normative theory is probably relevance theory (Sperber and Wilson, 1986). It is based on a more general theory of cognition, which loosely speaking hypothesises that, given competing inputs from their environment, people direct their attention to those inputs that provide them with the most useful information for the least processing cost. Applied to communication, the theory hypothesises that, given a potentially ambiguous communication as input, readers or listeners translate the input into a logical form that maximises the amount of information it contains, while minimising the computational effort needed to generate that logical form.

Relevance theory is compatible with the hypothesis that Computational Logic, or something like it, is the logic of the language of thought. Like Computational Logic, relevance theory also has both logical and computational components. Moreover, it provides a link with such normative theories of communication as Joseph Williams' guides to English writing style (Williams, 1990, 1995).

One way to interpret Williams' guidance is to understand it in logical terms, as including the advice that writers should express themselves in a form that is as close as possible to the logical form of the thoughts they want to communicate. In other words, they should say what they mean, and they should say it in a way that makes it as easy as possible for readers to extract that meaning. Or to put it still differently, the public expression of our private thoughts should be as close as possible to the logical form of those thoughts.

If our private language and public language were the same, we could literally just say what we think. But even that wouldn't be good enough; because we would still need to organise our thoughts coherently, so that one thought is logically connected to another, and so that our readers or listeners can relate our thoughts to thoughts of their own.

Williams' guidance for achieving coherence includes the advice of placing old, familiar ideas at the beginning of a sentence and placing new ideas at its end. In a succession of sentences, a new idea at the end of a sentence becomes an old idea that can be put at the beginning of the next sentence.

Here is an example of his advice, which uses an informal version of the syntax of Computational Logic, and which incidentally shows how Computational Logic can be used to represent an agent's goals and beliefs to guide its behaviour:

You want to be more intelligent.
You will be more intelligent if you are more logical.
You will be more logical if you study this book.
So (given no other alternatives) you should study this book.

It may not be poetry, and you might not agree with it, but at least it's clear, coherent and to the point.

What is Computational Logic?

The version of Computational Logic presented in this book combines a simplified form of language for representing information with mechanical (or automatic) ways of using information to infer its consequences. Sentences in this language have the simple form of *conditionals*: *if conditions then conclusion* (or equivalently *conclusion if conditions*). The basic rules of inference are forward and backward reasoning.

Forward reasoning is the classical rule of inference (also called *modus ponens*) used to derive *conclusions* from *conditions*. For example, given the belief that in general *a person will be more logical if the person studies this book*, forward reasoning derives the *conclusion* that *Mary will be more logical* from the *condition* that *Mary studies this book*. Forward reasoning includes the special case in which an agent derives consequences of its observations, to determine how those consequences might affect its goals.

Backward reasoning works in the opposite direction, to derive *conditions* from *conclusions*. For example, given the belief that in general *a person will be more intelligent if the person is more logical* as the *only* way of concluding that a person will be more intelligent, backward reasoning derives the condition that *John should be more logical* from the conclusion *John will be more intelligent*. Backward reasoning can be regarded as a form of goal reduction, in which the *conclusion* is a goal, and the *conditions* are subgoals. Backward reasoning includes the special case in which an agent derives subgoals that are actions, which the agent can perform in the world.

Backward reasoning gives Computational Logic the power of a high-level computer programming language, in which all programs consist of goal-reduction procedures. Indeed, the programming language Prolog, which stands for Programming in Logic, exploits this form of computation mainly for applications in Artificial Intelligence.

Computational Logic, in the more general form that we investigate in this book, also includes the use of inference to help an agent choose between alternative courses of action. For example, having used backward reasoning to derive two alternative subgoals, say *John is more logical* or *John takes intelligence-enhancing drugs*, for achieving the goal *John is more intelligent*, John can use forward reasoning to infer the possible consequences of the alternatives before deciding what to do. In particular, if John infers the consequence that *John may suffer irreversible brain damage* if John chooses the second alternative, *John takes intelligence-enhancing drugs*, then it will encourage John to choose the first alternative, *John is more logical*, instead.

What is Artificial Intelligence?

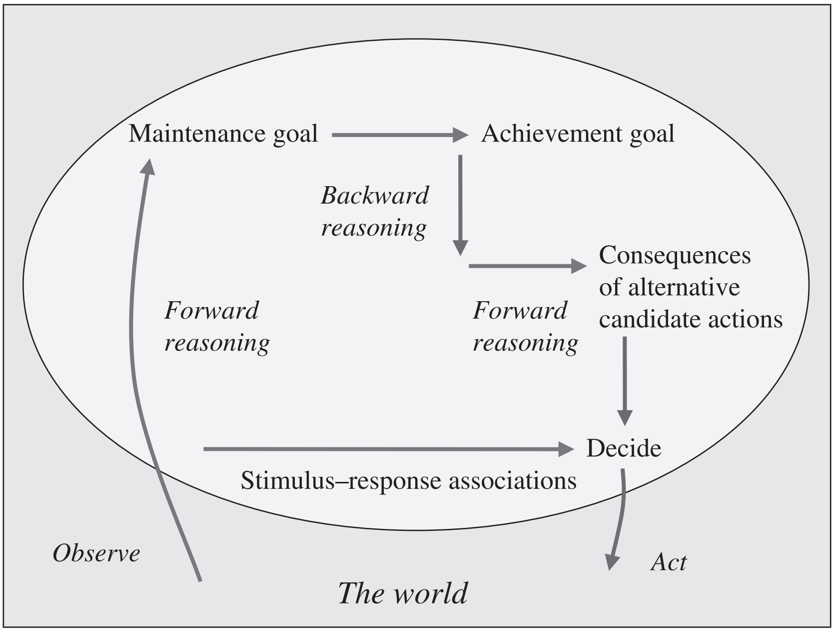
Artificial Intelligence (AI) is the attempt to program computers to behave intelligently, as judged by human standards. Applications of AI include such problem areas as English speech recognition, expert systems for medical and engineering fault diagnosis, and the formalisation of legal reasoning.

The tools of AI include such techniques as search, Symbolic Logic, artificial neural networks and reasoning with uncertainty. Many of these tools have contributed to the development of the Computational Logic we investigate in this book. However, instead of concerning ourselves with Artificial Intelligence applications, we will focus on the use of Computational Logic to help ordinary people think and behave more intelligently.

Thinking of people in computational terms might suggest that people can be treated as though they were merely machines. On the contrary, I believe instead that thinking of other people as computing agents can help us to better appreciate our common nature and our individual differences. It highlights our common need to deal with the cycle of life in an ever-changing world; and it draws attention to the fact that other people may have other experiences, goals and beliefs, which are different from our own, but which are equally worthy of understanding, tolerance and respect.

Computational Logic and the cycle of life

The role of Computational Logic in the mind of an intelligent agent can be pictured approximately like this:



In this way of looking at the relationship between an agent and the world, the mind of the agent is a syntactic structure, which represents the agent's beliefs about the world as it is and its goals for the way it would like the world to be. These beliefs and goals are represented in the agent's private language of thought, whose sentences have the syntactic form of conditionals.

The world, on the other hand, is a semantic structure, which includes the agent's body, and gives meaning to the agent's thoughts. It is a dynamic structure, which is continuously changing, and exists only in the here and now. However, the agent can record its changing experiences in its language of thought, and formulate general beliefs about the causal relationships between its experiences. It can then use these beliefs, which explain its past experiences, to help it achieve its goals in the future.

The agent observes events that take place in the world and the properties that those events initiate and terminate. It uses forward reasoning to derive conclusions of its observations. In many cases, these conclusions are actions, triggered by instinctive or intuitive stimulus–response associations, which can also be expressed in the logical form of conditionals. The agent may execute these actions by reflex, automatically and immediately. Or it may monitor them by performing higher-level reasoning, as in dual process models of human thinking.

But whether an agent is tempted to react immediately with stimulus–response associations or not, the agent can reason forwards to determine whether the observation affects any higher-level goals that need to be maintained to keep it in a harmonious relationship with its environment. Forward reasoning with higher-level maintenance goals of this kind generates achievement goals for the future. The agent can reason backwards, to reduce these achievement goals to subgoals and to search in its mind for plans of actions to achieve these goals.

The agent may find that there are several, alternative plans all of which achieve the same goal; and, if there are, then the agent needs to decide between them. In classical decision theory, the agent uses the expected consequences of its candidate plans to help it make this decision. With its beliefs represented in the logical form of conditionals, these consequences can be derived by reasoning forwards from conditions that represent the hypothetical performance of alternative candidate actions. The agent can evaluate the consequences, reject actions that have unintended and undesirable consequences, and choose actions that have the most desirable expected outcomes (or utility).

However, the consequences of an agent's actions may depend, not only on its own actions, but also on the actions of other agents or on other conditions that are outside the agent's control. The agent may not be able to determine for certain whether these conditions hold in advance, but it may be able to judge their likelihood (or probability). In such cases, the agent can use the techniques of decision theory, to combine its judgements of probability and utility, and choose a course of actions having the highest expected utility. Alternatively, the agent may use more pragmatic, precompiled plans of action that approximate the decision-theoretic ideal.

Among the criteria that an agent can use to decide between alternative ways of accomplishing its goals, is their likely impact on the goals of other agents. Alternatives that help other agents achieve their goals, or that do not hinder the achievement of their goals, can be given preference over other alternatives. In this way, by helping the agent to understand and appreciate that other agents have their own experiences, goals and beliefs, Computational Logic can help the agent avoid conflict and cooperate with other agents.

This book aims to show that these benefits of Computational Logic, which have had some success in the field of Artificial Intelligence, also have great potential for improving human thinking and behaviour.

1

Logic on the Underground

If some form of Computational Logic is the language of human thought, then the best place to look for it would seem to be inside our heads. But if we simply look at the structure and activity of our brains, it would be like looking at the hardware of a computer when we want to learn about its software. Or it would be like trying to do sociology by studying the movement of atomic particles instead of studying human interactions. Better, it might seem, just to use common sense and rely on introspection.

But introspection is notoriously unreliable. Wishful thinking can trick us into seeing what we want to see, instead of seeing what is actually there. The behavioural psychologists of the first half of the twentieth century were so suspicious of introspection that they banned it altogether.

Artificial Intelligence offers us an alternative approach to discovering the language of thought, by constructing computer programs whose input–output behaviour simulates the externally visible manifestations of human mental processes. To the extent that we succeed in the simulation, we can regard the structure of those computer programs as analogous to the structure of the human mind, and we can regard the activity of those programs as analogous to the activity of human thinking.

But different programs with different structures and different modes of operation can display similar behaviour. As we will see later, many of these differences can be understood as differences between levels of abstraction. Some programs are closer to the lower and more concrete level of the hardware, and consequently are more efficient; others are closer to the higher and more abstract level of the application domain, and consequently are easier to understand. We will explore some of the relationships between the different levels later in the book, when we explore dual process theories of thinking in Chapter 9. In the meanwhile, we can get an inkling of what is to come by first looking closer to home.

If human thoughts have the structure of language, then we should be able to get an idea of that structure by looking at natural languages such as English. Better than that, we can look at English communication in situations where we do our best to express ourselves as clearly, coherently and effectively as possible. Moreover, we can be guided in this by the advice we find in books on English writing style.

For the purpose of revealing the language of thought, the most important advice is undoubtedly the recommendation that we express ourselves as clearly as possible – making it as easy as we can for the people we are addressing to translate our communications into thoughts of their own. Everything else being equal, the form of our communications should be as close as possible to the form of the thoughts that they aim to convey.

What better place to look than at communications designed to guide people on how to behave in emergencies, in situations where it can be a matter of life or death that the recipient understands the communication as intended and with as little effort as possible.

Imagine, for example, that you are travelling on the London Underground and you hear a suspicious ticking in the rucksack on the back of the person standing next to you. Fortunately, you see a notice explaining exactly what to do in such an emergency:

> *Emergencies*
>
> *Press the alarm signal button*
> *to alert the driver.*
>
> *The driver will stop*
> *if any part of the train is in a station.*
>
> *If not, the train will continue to the next station,*
> *where help can more easily be given.*
>
> *There is a fifty pound penalty*
> *for improper use.*

The public notice is designed to be as clear as possible, so that you can translate its English sentences into your own thoughts with as little effort as possible. The closer the form of the English sentences to the form in which you structure your thoughts, the more readily you will be able to understand the sentences and to make use of the thoughts that they communicate.