The Logical Way to Be Artificially Intelligent

Robert Kowalski

Imperial College London
rak@doc.ic.ac.uk
http://www.doc.ic.ac.uk/~rak/

Abstract. Abductive logic programming (ALP) can be used to model reactive, proactive and pre-active thinking in intelligent agents. Reactive thinking assimilates observations of changes in the environment, whereas proactive thinking reduces goals to sub-goals and ultimately to candidate actions. Pre-active thinking generates logical consequences of candidate actions, to help in deciding between the alternatives. These different ways of thinking are compatible with any way of deciding between alternatives, including the use of both decision theory and heuristics.

The different forms of thinking can be performed as they are needed, or they can be performed in advance, transforming high-level goals and beliefs into lower-level condition-action rule form, which can be implemented in neural networks. Moreover, the higher-level and lower-level representations can operate in tandem, as they do in dual-process models of thinking. In dual process models, intuitive processes form judgements rapidly, sub-consciously and in parallel, while deliberative processes form and monitor judgements slowly, consciously and serially.

ALP used in this way can not only provide a framework for constructing artificial agents, but can also be used as a cognitive model of human agents. As a cognitive model, it combines both a descriptive model of how humans actually think with a normative model of humans can think more effectively.

1 Introduction

Symbolic logic is one of the main techniques used in Artificial Intelligence, to develop computer programs that display human intelligence. However, attempts to use symbolic logic for this purpose have identified a number of shortcomings of traditional logic and have necessitated the development of various improvements and extensions. This paper - and the draft book [6] on which it is based - aims to show that many of these developments can also be used for the original purpose of logic – to improve the quality of human thinking.

I have written the book informally, both to reach a wider audience and to demonstrate that the enhanced logic is in fact relevant and congenial for human thinking. However, in this paper, I will draw attention to some of the more technical issues, for the consideration of a more academic audience.

The logic used in the book is based on an extension of abductive logic programming (ALP) to logic-based agents [7]. In ALP agents, *beliefs* are represented by logic programs and *goals* are represented by integrity constraints. The agent's

F. Toni and P. Torroni (Eds.): CLIMA VI, LNAI 3900, pp. 1 – 22, 2006.

observations and *actions* are represented by abducible predicates. Beliefs and goals have both a declarative interpretation in logical terms, as well as a procedural interpretation in computational terms.

ALP agents are both *reactive* to changes they observe in the environment and *proactive* in planning ahead and reducing goals to sub-goals. In this paper I show that ALP agents can also be *pre-active* in thinking about the possible consequences of actions before deciding what to do.

In conventional ALP, the logical consequences of abductive hypotheses are checked to determine whether they violate any integrity constraints. However, in ALP agents, where abductive hypotheses include alternative, candidate actions, the pre-actively generated consequences of candidate actions are used to decide between the alternatives. This decision can be made in different ways. One way is to use conventional Decision Theory, judging the utilities and probabilities of the consequences of the alternative candidates and choosing an action that maximizes expected utility. However, other ways of deciding between actions are also compatible with ALP, including ways that compile decision making into heuristics.

The combination of reactive, proactive and pre-active thinking is obtained in ALP agents by combining forward and backward reasoning. This reasoning can be performed whenever the need arises, or it can be performed once and for all by reasoning in advance. Reasoning in advance transforms and compiles higher-level goals and beliefs into lower-level goals, which are similar to condition-action rules, which implement stimulus-response associations compiled into neural networks.

In modern computing, it is common to develop programs in a high-level representation and then to transform or compile them into a lower-level representation for the sake of efficiency. If it later becomes necessary to correct or enhance the resulting lower-level program, this is generally done by first modifying the higher-level representation and then recompiling it into a new lower-level form.

However, many existing computer systems are legacy systems developed before the existence of higher-level programming languages. It is often possible to decompile these lower-level programs into higher-level form, although, because of the undisciplined nature of lower-level languages, sometimes the relationship is only approximate.

The relationship between higher-level and lower-level computer programs is analogous to the relationship between higher-level and lower-level representations in ALP agents. It is also similar to the relationship between deliberative and intuitive thinking in the *dual process model* of human thinking [10]. In the dual process model, one system, which is older in evolutionary terms, is responsible for *intuitive thinking*. It is associative, automatic, unconscious, parallel, and fast. The other system, which is distinctively human, is responsible for *deliberative thinking*. It is rule-based, controlled, conscious, serial, and slow.

In computing, high-level and low level representations normally operate separately, but can be compiled or decompiled from one into the other. In the dual process model, however, intuitive and deliberative thinking can operate in tandem, as when the intuitive, subconscious level "quickly proposes intuitive answers to judgement problems as they arise", while the deliberative, conscious level "monitors the quality of these proposals, which it may endorse, correct, or override" [3]. This interaction between intuitive and deliberative thinking can be mimicked in part by the

use of pre-active thinking in ALP agents, to monitor and evaluate candidate actions generated by reactive thinking. In ALP agents both the deliberative level and the intuitive level are represented in logical form.

These topics are expanded upon in the remainder of the paper. Section 2 outlines the basic features of the ALP agent model, including reactive, proactive, and pre-active thinking. Section 3 investigates the relationship between thinking and deciding. Section 4 discusses the transformation of high-level representations into lower-level, more efficient form, and the way in which high-level and lower-level representations interact. Section 5 shows how low-level feed-forward neural networks can be represented in logical form and can be simulated by forward reasoning. Section 6 discusses some of the implications of this for the notion that logic can serve as a wide-spectrum language of thought. Section 7 addresses some of the arguments against logic as a model of human thinking, and section 8 is the conclusion.

2 The Basic ALP Agent Model

2.1 Putting Logic in its Place in the Agent Cycle

The logic used in the book is based on an extension of abductive logic programming (ALP) to logic-based agents [7]. The most important feature of the extension is that it embodies logic in the thinking component of an agent's observe-think-decide-act cycle:

To cycle,
observe the world,
think,
decide what actions to perform,
act,
cycle again.

The agent cycle can be viewed as a generalisation of production systems, in which thinking is performed by using condition-action rules of the form:

If conditions then candidate actions.

Condition-action rules look a lot like logical implications, but they do not have the declarative semantics of implications. Nonetheless, as we will later see, in ALP agents, condition-action rules are represented by goals expressed in logical form.

This view of logic in the mind of an agent embodied in the world is pictured in figure 1. In this picture, the agent uses logic to represent its goals and beliefs, and to help control its interactions with the world. It transforms its experience into observations in logical form and uses its goals and beliefs to generate candidate actions, to satisfy its goals and to maintain itself in a satisfactory relationship with the changing world.

The agent's body, in addition to being a part of the world, transforms both raw experience into observations and the will to act into physical changes in the world. This is analogous to the way in which hardware and software are related in a computer. The hardware transforms stimuli from the environment into inputs and

transforms outputs into physical changes in the environment. The internal processing of inputs into outputs is performed by the hardware, but is controlled conceptually by software. In this analogy, the brain and body of an agent are to the mind as hardware is to software.

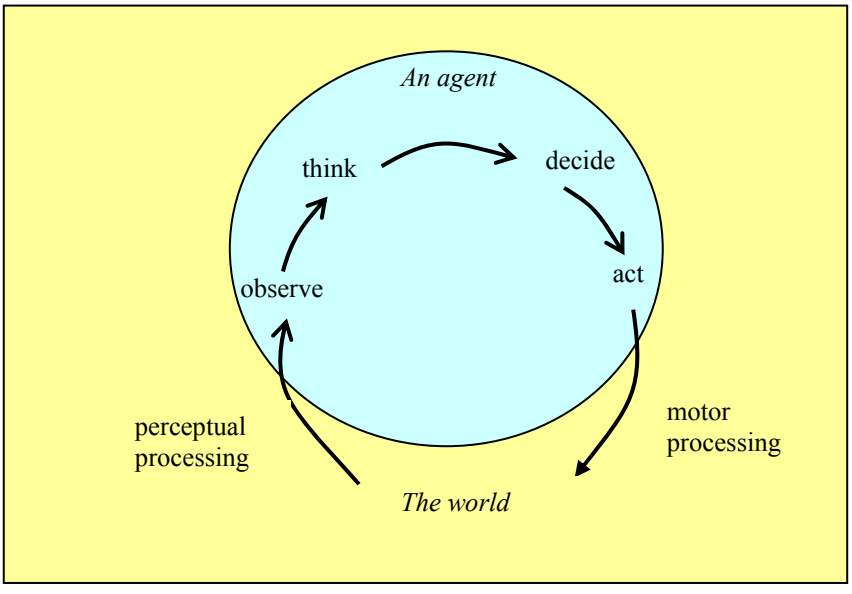

Fig. 1. The agent cycle

In general, the result of thinking is a set of candidate actions, which are the input to the decision-making component of the agent cycle. In the same way that Logic is only one way of thinking, there are many ways of deciding what to do. Decision theory, which combines judgements about the utility of the outcomes of actions with judgements about their probability, is one such way of deciding. As we will see in an example later, it is also possible to compile decision-making directly into lower-level goals and beliefs. In production systems, decision making is called "conflict resolution".

An agent's ultimate goal is to maintain itself in a satisfactory relationship with the surrounding world, and thinking and deciding are only one way of achieving that goal. An agent can also act to satisfy its goals instinctively, by means of *stimulus-response associations*, in a way that might be characterised as acting without thinking. Instinctive behaviour can be hardwired into an agent's body, without entering into its mind. Or it might be learned as the result of repeated performance and feedback. Instinctive behaviour is a near relation of intuitive thinking in the dual process model.

The agent cycle, as described above, concerns the real time behaviour of an agent, and does not address the processes involved in learning new behaviours and updating old ones. Suffice it to say that learning, belief revision and goal revision are essential activities for a real agent interacting with the real world. Because of the logical nature

of ALP agents, such techniques as inductive logic programming are especially suitable to model such activities. They are, however, beyond the scope of this paper.

2.2 ALP Combines Forward and Backward Reasoning

Abductive logic programming [4] comes in many forms and variations, both in terms of its semantics and in terms of its proof procedures. However, in all of these forms, abductive logic programs have two components: ordinary logic programs and integrity constraints. They also have two, corresponding kinds of predicates – *ordinary predicates* that are defined by logic programs and *abducible predicates* that are, directly or indirectly, constrained by integrity constraints.

In ALP agents, logic programs are used to represent an agent's *beliefs*, and integrity constraints to represent its *goals*. The abducible predicates are used to represent the agent's *observations* and *actions*. The integrity constraints are *active*, in the sense that they can generate representations of actions that the agent can perform, in order to maintain integrity.

Consider, for example, the goal of getting help in an emergency on the London underground.

Goal If there is an emergency then I get help.

Beliefs There is an emergency if there is a fire.
There is an emergency if one person attacks another.
There is an emergency if someone becomes seriously ill.
There is an emergency if there is an accident.

There is a fire if there are flames.[1]
There is a fire if there is smoke.

A person gets help
if the person alerts the driver.

A person alerts the driver
if the person presses the alarm signal button.

Here, for simplicity, the abducible predicates associated with observations are the predicates "there are flames", "there is smoke", "one person attacks another", "someone becomes seriously ill", and "there is an accident". The only abducible predicate associated with candidate actions is "the person presses the alarm signal button". All of these abducible predicates are indirectly constrained by the goal of getting help whenever there is an emergency. All the other predicates, including the higher-level actions of getting help and alerting the driver are ordinary predicates, defined by the agent's beliefs.

[1] These two rules, relating fire, flames and smoke are the converse of the causal rules, which state that if there is a fire then there are flames and smoke. The causal rules are a higher-level representation, whereas the rules used here are a lower-level, more efficient representation. The higher-level, causal representation would need abduction to explain that an observation of smoke or flames can be caused by fire. In fact, the term "abduction" normally refers to such generation of hypotheses to explain observations. The lower-level representation used here replaces abduction by deduction.

The goal itself is a *maintenance goal*, which an agent can use to derive actions to maintain itself in a desired relationship with the changes that it observes in its environment. Maintenance goals can be viewed as a generalization of condition-action rules.

Maintenance goals are triggered as a consequence of observations, similarly to the way in which integrity constraints in a database are triggered as the result of updates. An agent reasons forwards from its beliefs, to derive consequences of its observations. Suppose, for example, that I am travelling as a passenger on the underground and that my body experiences a combination of sensations that my mind interprets as an observation of smoke. The observation triggers my beliefs, which I use to reason forward in two steps, to recognize that there is an emergency.

The conclusion that there is an emergency triggers the maintenance goal, which I then use to reason forward one more step, to derive the achievement goal of getting help. The achievement goal triggers other beliefs, which I use to reason backwards in two steps, to reduce the achievement goal to the action sub-goal of pressing the alarm signal button. Since there are no other candidate actions in this simple example, I decide to press the alarm signal button, which my body then transforms into a combination of motor activities that is intended to accomplish the desired action.

The fact that pure logic programs are declarative means that they can be used to reason in many different ways. In the procedural interpretation, they are used only to reason backwards, as procedures that reduce goals to sub-goals. However, in ALP they are used to reason both backwards and forwards.

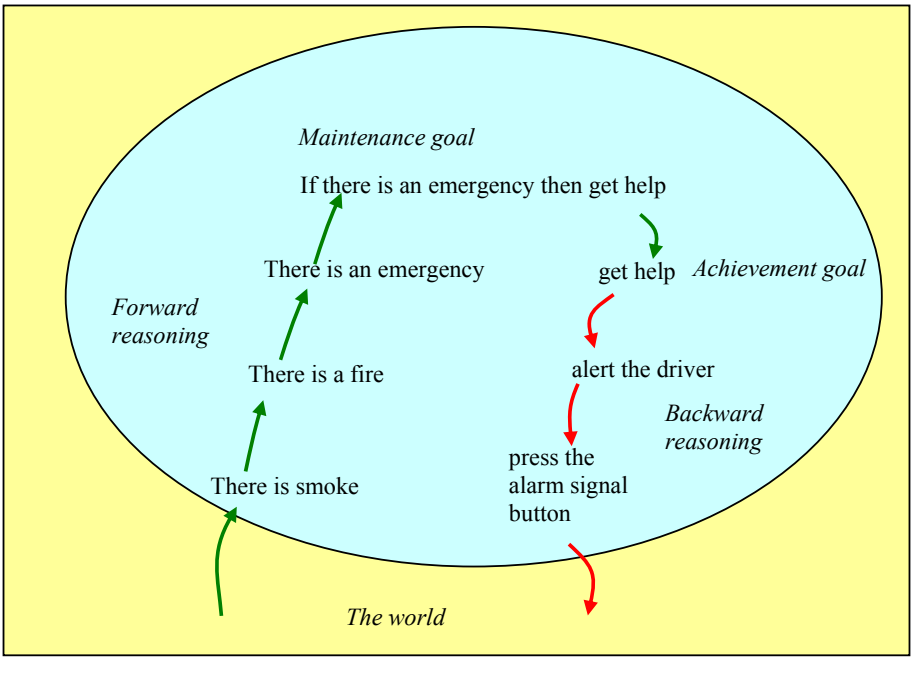

Fig. 2

This combination of forward and backward reasoning, together with the interface between the agent's mind and the world, is pictured in figure 2. Arguably, this treatment of maintenance goals as integrity constraints generalizes condition-action rules in production systems. Condition-action rules are the special case where no forward reasoning is needed to trigger the maintenance goal and no backward reasoning is needed to reduce the achievement goal to actions. Thus maintenance goals include condition-action rules as a special case, but in general are much higher-level.

Vickers [12], in particular, championed the idea that human activity and organizations should be viewed as maintaining relationships with the changing environment. He characterized Simon's view of management and problem solving as the narrower view of only solving achievement goals. Vickers view-point has been taken up by in recent years by the soft systems school of management [2].

2.3 ALP Combines Reactive and Proactive Thinking

The combination of forward and backward reasoning enables ALP agents to be both reactive and proactive. They are reactive when they use forward reasoning to respond to changes in the environment, and they are proactive when they use backward reasoning to achieve goals by reducing them to sub-goals. In everyday life, human agents are both reactive and proactive to varying degrees.

Consider, as another example, a simplified ALP version of Aesop's fable of the fox and the crow. Suppose the fox has the following achievement goal and beliefs:

Goal I have the cheese.

Beliefs The crow has the cheese.

An animal has an object
if the animal is near the object
and the animal picks up the object.

I am near the cheese
if the crow has the cheese
and the crow sings.

The crow sings if I praise the crow.

The fox can use its beliefs as a logic program, to reason backwards, to reduce its goal to the actions of praising the crow and picking up the cheese.[2] The fox's reduction of its goal to sub-goals is pictured in figure 3.

In keeping with the view that the primary goals of an agent are all maintenance goals, the fox's achievement goal almost certainly derives from a maintenance goal, such as this:

If I become hungry, then I have food and I eat it.

[2] The story is simplified partly because the element of time has been ignored. Obviously, the fox needs to praise the crow before picking up the cheese.

Here the condition of being hungry is triggered by an observation of being hungry, which the fox receives from its body. Notice that the achievement goal of having the food is only half of the story. To satisfy the maintenance goal, the fox also needs to eat the food.

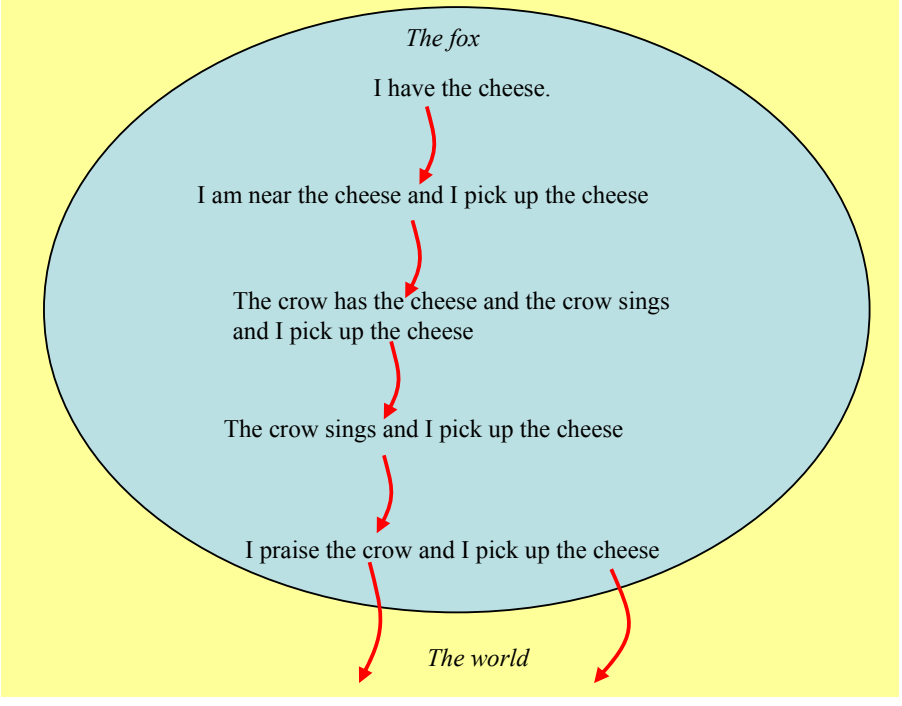

Fig. 3

In Aesop's fable, the fox's belief about the behaviour of the crow is true. The crow is a purely reactive agent, which responds to praise as the fox believes. The reactivity of the crow can be viewed as reasoning forwards in one step from an observation to derive an achievement goal, which is an action, from a maintenance goal. This is pictured in figure 4.

This view of the crow's behaviour is a logical abstraction of behaviour that might be hardwired into the crow as a system of lower-level stimulus-response associations. The relationship between such a logical abstraction and the stimulus-response associations is, arguably, like the relationship between software and hardware.

Notice the difference between the sentence

If the fox praises me, then I sing.

which is a goal for the crow, and the sentence

The crow sings if I praise the crow.

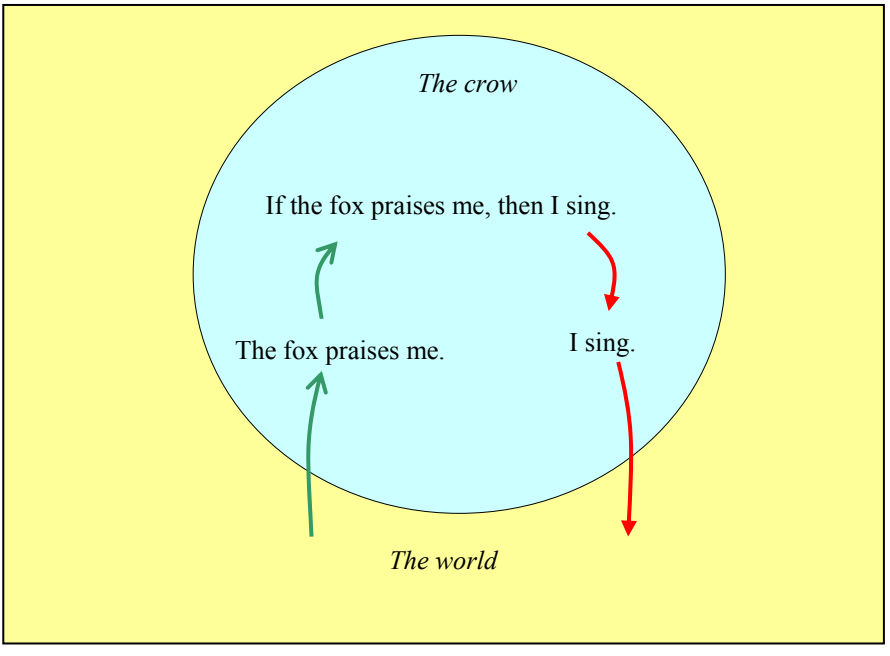

Fig. 4

which is a belief for the fox. Both sentences are implications. However, for the crow, the implication is used as a goal, to generate its behaviour. But for the fox, the implication is used as a belief, to describe the crow's behaviour and to reduce goals to sub-goals.

The difference between the two sentences has nothing to do with the order in which the conclusion and condition of the implication is written, because there is no semantic difference between writing an implication forwards in the form

If conditions then conclusion.

and writing it backwards in the form

Conclusion if conditions.

Semantically both implications have the same declarative meaning. (In the same way that both inequalities $1 < 2$ and $2 > 1$ have the same meaning.)

However, no matter how implications are written, there is an important distinction between them depending upon whether they are used as goals or as beliefs. When they are used as beliefs, they represent the world as it actually is. When they are used as goals, they represent the world as the agent would like it to be. When a goal is an implication, the agent performs actions to make the implication true. It only needs to perform these actions to make the conclusion of the implication true when the world makes the conditions of the implication true. It need not worry about performing actions when the world makes the conditions of the implication false. The analogous distinction in deductive databases between implications used as integrity constraints and implications used as rules was first investigated by Nicolas and Gallaire [17].

2.4 ALP Includes Pre-active Thinking

Aesop's fable shows how a proactive fox outwits a reactive crow. But there is an even more important moral to the story - namely that an intelligent agent should think before it acts. Thinking before acting is more than just proactive thinking. It is thinking about the possible consequences of candidate actions - *pre-actively* – before deciding what to do. Pre-active thinking is obtained in ALP by reasoning forward from candidate actions, whether derived proactively or reactively, and whether generated by symbolic reasoning or by instinctive stimulus-response associations.

Suppose, for example, that the crow not only has the maintenance goal of singing whenever it is praised, but also has the achievement goal (for whatever reason) of having the cheese. If the crow also has the same beliefs as the fox, then the crow would be able to reason forward, pre-actively, to deduce the possible consequences of singing:

> I want to sing.
>
> But if I sing,
> then the fox will be near the cheese.
>
> *Perhaps* the fox will pick up the cheese.
> Then the fox will have the cheese,
> and I will not have the cheese.
>
> Since I want to have the cheese,
> I will not sing.

Notice that the crow can not consistently achieve the goal of having the cheese and also maintain the goal of singing whenever it is praised. In real life, an agent needs to weigh its goals, trading one goal off against another.[3]

Notice too that the outcome of an agent's actions typically depends also on external events, over which the agent may have little or no control. In the story of the fox and crow, the outcome of the crow's singing depends on whether or not the fox decides to pick up the cheese.

3 Thinking Needs to be Combined with Deciding What to Do

In ALP, pre-active thinking simply checks whether candidate actions satisfy the integrity constraints. However, in real life, we also have to choose between actions, taking into consideration the relative utilities and probabilities of their possible consequences. In Decision Theory, the agent uses these considerations to choose an action that has maximum expected utility.

[3] Alternatively, if the crow wants to have the cheese in order to eat it, then the crow could satisfy both goals by first eating the cheese and then singing.

3.1 Combining ALP with Decision Theory

Suppose, for example, that I have the following beliefs:

I get wet if it rains and I do not carry an umbrella.
I stay dry if I carry an umbrella.
I stay dry if it doesn't rain.

Assume also that I am about to leave home, and that as a sub-goal of leaving home I have to decide what to take with me, and in particular whether or not to take an umbrella. I can control whether to take an umbrella, but I can not control whether it will rain. At best I can only judge the probability of rain.

Reasoning forward from the assumption that I take an umbrella and then have to carry it, I can derive the certain outcome that I will stay dry. However, reasoning forward from the assumption that I do not carry an umbrella, I derive the uncertain outcome that I will get wet or I will stay dry, depending on whether or not it will rain.

In classical logic, that would be the end of the story. But, in Decision Theory, I can judge the likelihood that it is going to rain, judge the positive utility of staying dry compared with the negative utility of having to carry the umbrella, weigh the utilities by their associated probabilities, and then choose an action that has the maximum expected utility.

For the record, here is a simple, example calculation:

Utility of getting wet = – 8.
Utility of staying dry = 2.
Utility of carrying an umbrella = – 3
Utility of not carrying an umbrella = 0
Probability of raining = .1
Probability of not raining = .9

Assume I take an umbrella.
Then Probability of staying dry = 1
Expected utility = 2 – 3 = - 1

Assume I do not take an umbrella .
Then Probability of staying dry = .9
Probability of getting wet =.1
Expected utility = .9 ·2 - .1 ·8 = 1.8 - .8 = 1

Decide I do not take an umbrella!

Given the same utilities, the probability of rain would have to be greater than .3 before I would decide to take an umbrella.

Because thinking and deciding are separate components of the agent cycle, any way of thinking is compatible with any way of deciding. Thus the use of ALP for thinking can be combined with Decision Theory or any other way of deciding what to do. This combination of thinking and deciding in ALP agents is pictured in figure 5.

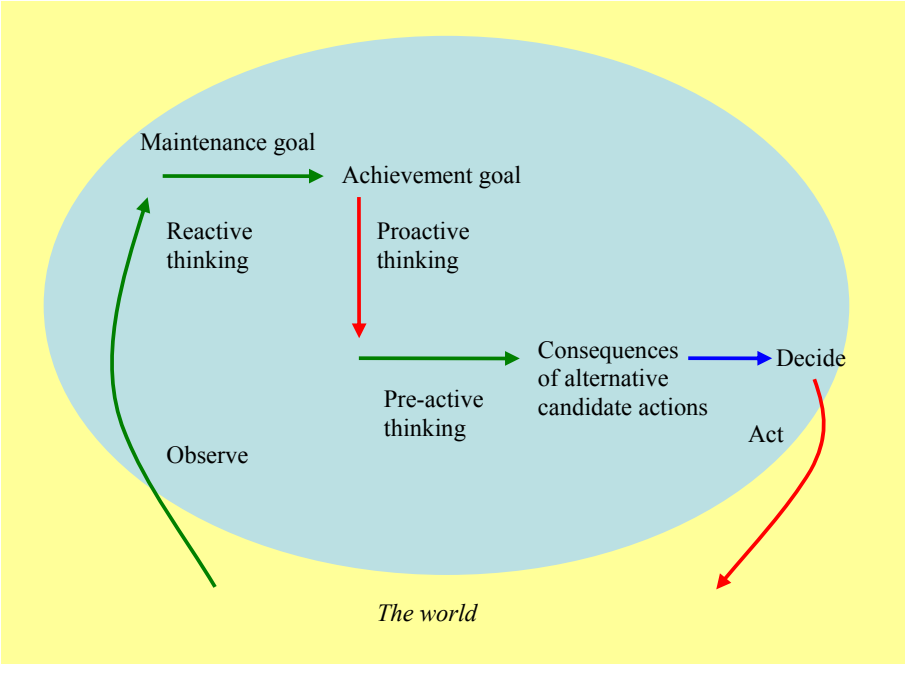

Fig. 5

A combination of abductive logic programming and Decision Theory has been developed by David Poole in his Independent Choice Logic [8]. He shows how the logic can be used to represent Bayesian networks, influence diagrams, Markov decision processes and the strategic and extensive form of games.

Poole focuses on the semantics of ICL, whereas I focus on the logical and computational procedures an individual agent might use in practice. One consequence of this difference is that he views condition-action rules as *policies*, and represents them by ordinary logic programs, whereas I view them as goals, and represent them as integrity constraints.

3.2 Decision Making Can Often Be Compiled into the Thinking Component of the Agent Cycle

The problem with Decision Theory is that it requires an unrealistic amount of information about utilities and probabilities and too much computation. Nonetheless, Decision Theory represents a normative ideal against which other, more practical decision-making methods can be evaluated.

In the case of taking or not taking an umbrella, a more practical alternative might be to use maintenance goals or condition-action rules instead[4]:

[4] In this representation the decision not to take an umbrella is implicit. It holds if the decision to take an umbrella does not hold.

If I leave home and it is raining then I take an umbrella.
If I leave home and there are dark clouds in the sky then I take an umbrella.
If I leave home and the weather forecast predicts rain then I take an umbrella.

The maintenance goals in this example compile decision-making into the thinking component of the agent cycle. In some cases, the compilation might be an exact implementation of the Decision Theoretic specification. In other cases, it might only be an approximation.

Other alternatives to Decision Theory include the use of priorities between different actions, goals or condition-action rules, and the use of default reasoning.

4 Combining Higher-Level and Lower-Level Thinking

4.1 Higher Levels of Thinking Can Be Compiled into Lower Levels

Abductive logic programs have a computational, as well as a logical, interpretation. Goals and beliefs expressed in logical form can be viewed as programs written in a high-level programming language. Programs written at this high, logical level are executed by backward and forward reasoning.

For the sake of efficiency, high-level programs are often compiled into lower-level programs and are executed using corresponding lower-level computational mechanisms. Usually the higher and lower-level programs are written in distinct programming languages. However, they can also be written in the same language.

Compiling a high level program into a more efficient, lower level program written in the same language is called *program transformation.* Program transformation typically gains efficiency by performing at compile time, once and for all, execution steps that would otherwise have to be performed repeatedly, at run time. In the case of abductive logic programs, higher-level programs can be transformed into lower-level, more efficient programs, by performing reasoning steps in advance, before they are needed.

This is easy to see in the London underground example. The original high-level ALP representation can be compiled/transformed into the equivalent, more efficient condition-action rule representation:

If there are flames then I press the alarm signal button.
If there is smoke then I press the alarm signal button.
If one person attacks another then I press the alarm signal button.
If someone becomes seriously ill then I press the alarm signal button.
If there is an accident then I press the alarm signal button.

This lower-level program is written in the same higher-level ALP language as the original representation, but it now consists of five maintenance goals, rather than one maintenance goal and eight beliefs. It is obtained by reasoning in advance, replacing the concept of “emergency” by all of the alternative types of emergency, replacing the concept of “fire” by the two different ways of recognizing a fire, and reducing “getting help” to the action of pressing the alarm signal button.

The two representations are computationally equivalent, in the sense that they give rise to the same externally observable behaviour. However, the lower-level program is more efficient. Not only does it require fewer steps to execute at run time, but it uses simpler reasoning techniques, consisting of forward reasoning alone, instead of the combination of forward and backward reasoning needed by the higher-level program.

The two representations are not logically equivalent. The high-level representation logically implies the lower-level representation, but not vice versa. In particular, the higher-level representation has an explicit representation of the concepts of there being an emergency and of getting help, which are only implicit in the lower-level representation. Moreover, the higher-level representation also has an explicit representation of the purpose of the agent's behaviour, namely to get help whenever there is an emergency, which is only implicit as an *emergent goal* in the lower-level representation.

In computing, higher-level representations (including program specifications) are generally developed, before they are compiled/transformed into lower-level representations for the sake of efficiency. However, if anything then goes wrong with the lower-level representation, it is generally easier to debug and correct the higher-level representation and to recompile it into the lower-level form, than it is to change the lower-level representation itself.

For example, if something goes wrong with the condition-action rule formulation of the London underground rules - if the button doesn't work, or if the driver doesn't get help - then the rules will fail, but the passenger might not even recognise there is a problem. Or, if the environment changes – if new kinds of emergencies arise or if better ways of getting help are developed – then it is easier to extend the higher-level representation than it is to modify the lower-level rules.

In computing, it is common to iterate the compilation of programs into a number of increasingly lower-levels, and ultimately into hardware. Historically, however, lower-level languages were used before higher-level, more human-oriented languages were developed. Because legacy systems originally developed and implemented in such lower-level languages are difficult to maintain, it is common to re-implement them in modern higher-level languages. This can sometimes be done by an inverse process of *decompiling* lower-level programs into higher-level programs. However, because of the undisciplined nature of low-level programming languages, the attempt to decompile such programs may only be partially successful. In many cases it may only be possible to approximate the lower-level programs by higher-level ones, sometimes only guessing at their original purpose.

4.2 Combining Deliberative and Intuitive Thinking

The relationship between deliberative and intuitive thinking is analogous to the relationship between higher-level and lower-level program execution.

The simplest relationship is when, as the result of frequent repetition, deliberative thinking migrates to the intuitive level – when, for example, a person learns to use a keyboard, play a musical instrument, or drive a car. This is like compiling or transforming a high-level program into a lower-level program. After a particular combination of high-level, general-purpose procedures has been used many times over, the combination is compressed into a computationally equivalent, lower-level

shortcut. The shortcut is a special-purpose procedure, which achieves the same result as the combination of more general procedures, but it does so more efficiently and with less awareness of its purpose.

Conversely, intuitive thinking and tacit knowledge can sometimes be made explicit – for example, when a linguist constructs a formal grammar for a natural language, a coach explains how to swing a golf club, or a knowledge engineer develops an expert system. This is like decompiling a low-level representation into a higher-level representation. In many cases it can be hard to distinguish whether the low-level representation is implemented in hardware or in software, and the resulting higher-level representation may only be approximate.

In computing, higher-level and lower-level programs can operate in tandem, as when the lower-level program is used on a routine basis, but the higher-level program is used to modify and recompile the lower-level program when it goes wrong or needs to be updated. In human thinking, however, intuitive and deliberative thinking are often coupled together more closely. Intuitive thinking generates candidate judgments and actions rapidly and unconsciously, while deliberative thinking consciously monitors the results. This close coupling of deliberative and intuitive thinking is like the use of pre-active thinking in ALP agents to monitor candidate actions generated reactively by condition-action rules.

These relationships between different levels of thinking are pictured, somewhat imperfectly, in figure 6.

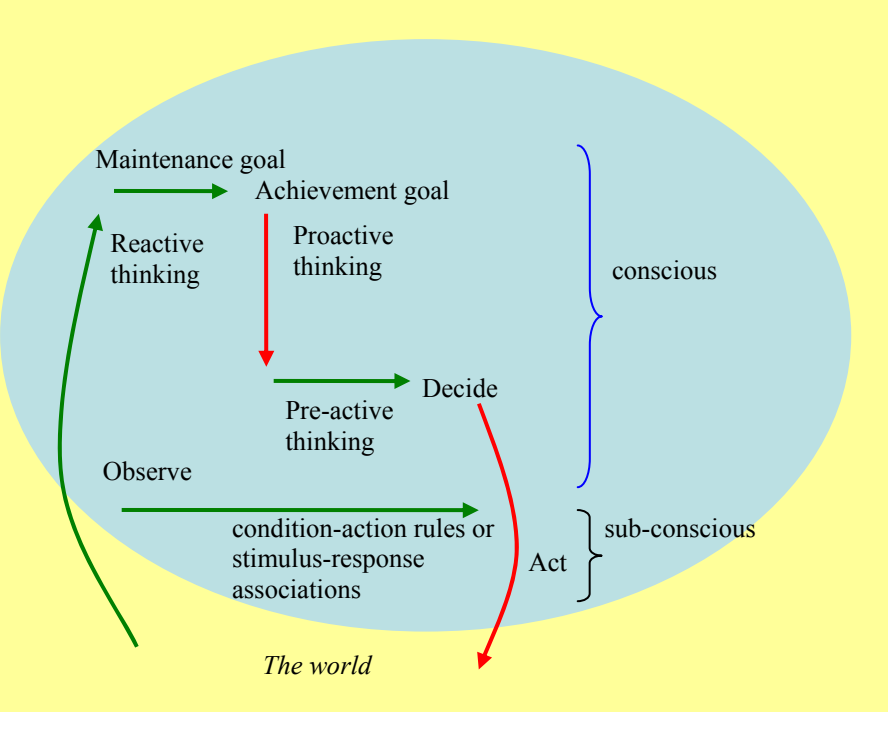

Fig. 6

5 Neural Networks

It is a common view in Cognitive Science that intuitive thinking is best modelled by sub-symbolic neural networks [13], which employ distributed representations with hidden nodes that do not have a symbolic interpretation. However, in their text book, *Computational Intelligence: A Logical Approach*, Poole *et al* [9] show how to represent any feed-forward neural network as a logic program. Forward reasoning with the logic program simulates forward execution of the neural network.

Poole *et al* illustrate their representation with the example (figure 7) of a person's decision whether to read an article. The decision is based upon such factors as whether the author is known or unknown, the article starts a new thread or is a follow-up article, the article is short or long, and the person is at home or at work.

The weights on the arcs are obtained by training an initial version of the network with a training set of examples. In the logic program, "f" is a sigmoid function that coerces the real numbers into the range [0,1]. Similarly, the "strengths" of the inputs lie in the range [0,1], where 0 is associated with the Boolean value *false* and 1 with *true*.

It is generally held that neural networks are unlike logic, in that they can have hidden units that can not be interpreted as concepts or propositions. Indeed, Poole *et al* characterize their example as illustrating just that point. However, in my formulation of the logic program, to make it more readable, I have given the predicate symbols "meaningful" predicate names, interpreting the hidden units in the middle layer of the network as summarizing the arguments for and against reading the paper.

Example	Action	Author	Thread	Length	Where read
E1	skip	known	new	long	home
E2	reads	unknown	new	short	work
E3	skips	unknown	follow-up	long	work

Neural network

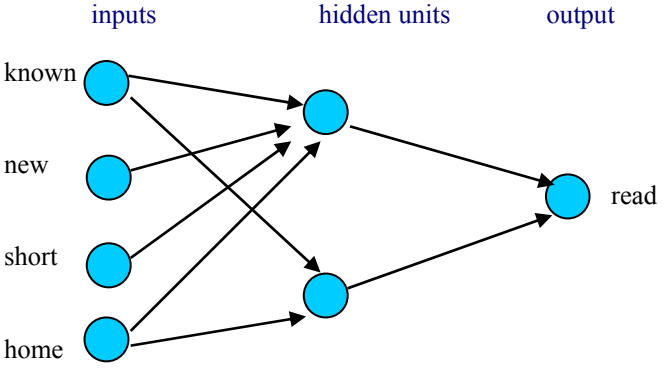

Fig. 7

Logic program

I read with strength S_3
if there is an argument for reading with strength S_1
and there is an argument against reading with strength S_2
and $S_3 = f(-2.98 + 6.88\ S_1 - 2.1\ S_2)$

There is an argument for reading with strength S_1
if known with strength S_4
and new with strength S_5
and short with strength S_6
and home with strength S_7
and $S_1 = f(-5.25 + 1.98\ S_4 + 1.86\ S_5 + 4.71\ S_6 - .389\ S_7)$

There is an argument against reading with strength S_2
if known with strength S_4
and new with strength S_5
and short with strength S_6
and home with strength S_7
and $S_2 = f(.493 - 1.03\ S_4 - 1.06\ S_5 - .749\ S_6 + .126\ S_7)$

The logic program is an exact, logical representation of the neural network. However, it employs numerical values and functions, which can only be approximated by a natural language representation, such as this:

I read an article
if the argument for reading the article is strong
and the argument against reading the article is weak.

There is an argument for reading an article
if the article is short.

There is an argument against reading an article
if the thread is a follow-up and the author is unknown.

The terms “strong” and “weak” are explicitly vague, whereas the notions of “an article being short”, “a thread being new” and “an author being known” are implicitly vague. Taking this into account, the representation can be transformed into a simpler form, where all vagueness is implicit and where the arguments for and against reading an article are also implicit:

I read an article
if the article is short and the thread is new.

I read an article
if the article is short and the thread is a follow-up and the author is known.

Expressed in this way and treating the sentences as goals rather than as beliefs, the problem of deciding whether to read an article is similar to the problem of deciding whether to take an umbrella when leaving home. In both cases, the decision depends

upon the interpretation of implicitly vague concepts, such as an article being short or there being dark clouds in the sky.

In both cases, moreover, the decision can also be made at a higher-level, by analysing the goals and other outcomes that the decision might be expected to achieve. In the case of reading an article, is the higher-level goal to gain information? Or is it simply to be entertained? If it is to gain information, how likely is it that the article will contain the information I am looking for? Is it worth the effort involved? Or would it be better to consult an expert instead?

In the case of taking an umbrella when I leave home, is the higher-level goal to keep my hair and clothing neat and tidy? Or is it to avoid catching a chill and coming down with a cold? In the first case, maybe it should just wear a hat and some suitable outdoor clothing. In the second case, if I am so fragile, then maybe I should stay home or travel by taxi.

6 Logic as Wide-Spectrum Language of Thought

The neural network example suggests that logic can represent a wide spectrum of levels of thought, ranging from subconscious thought at the neural network level to conscious thought in natural language. At the neural network level, logic programs can represent connections among vague concepts that hold with varying degrees of strength. Although these degrees might have precise values at the neurological level, they are not accessible to higher-levels of consciousness and can only be approximated in natural language.

A number of authors have investigated the relationship between neural networks and logic programs. One of the earliest of these investigations, by Holldobler and Kalinke [15], studied the problem of translating normal logic programs into neural networks. More recently, Stenning and van Lambalgen [16] have argued that the implementation of logic programs by neural networks shows that logic can model intuitive thinking in the dual process model. D'Avila Garcez, Broda and Gabbay [14], on the other hand, studied the problem of extracting higher-level, "meaningful" logic programs from neural networks. Taken together with the direct translation of neural networks into correspondingly low-level logic programs of Poole et al [9], these studies suggest that logic can model different levels of thought, from neural networks to natural language.

The relationship between logic and natural language is normally viewed from the linguistic perspective, by studying the problem of extracting logical meaning from natural language. But it can also be viewed from the knowledge representation perspective, by comparing the logical form of an agent's thoughts with the communication of those thoughts to another agent in natural language.

Although logical representations are normally presented in symbolic, mathematical form, they can also be expressed in a stylized form of natural language, as in this paper. Both of these forms are unambiguous and context-independent. Thus, to the extent that some form of logic is adequate for knowledge representation, this provides evidence that human agents might think in a mental language that is a logical form of natural language.

In contrast with the thoughts we have in our mind, our natural language communication of those thoughts is generally more ambiguous and context-sensitive than we intend. This suggests that our thoughts may be more logical than their natural language expression might suggest. Even natural language communications that seem to be in explicit logical form can be more ambiguous than they seem on the surface.

As Stenning and van Lambalgen [16] argue, natural language communications need to be interpreted to determine their intended logical form, even when those communications are already expressed in logical form. They argue that the gap between the surface logical structure of sentences and the deeper logical structure of their intended meanings helps to explain and refute certain psychological experiments that suggest that people are not logical. They show, moreover, that human performance in these experiments is compatible with the thesis that people apply logical reasoning to the intended meanings of sentences rather than to their surface form. In fact, in their main example, they show, not only that the intended meanings of sentences can be expressed in logical form, but that they have logic programming form, and that the minimal model semantics of logic programs gives a better analysis of human performance in these experiments than the classical semantics of traditional logic.

This difference between the surface structure of natural language and its underlying logical form is illustrated also by the second sentence of the London underground Emergency Notice:

> If there is an emergency then you press the alarm signal button.
> The driver will stop if any part of the train is in a station.

The second sentence has an explicitly logical form, due to its use of the logical connective "if" and the quantifier "any". However, taken literally, the sentence doesn't express what its authors probably had in mind:

> The driver will stop *the train* if *someone presses the alarm signal button*
> and any part of the train is in a station.

It is likely that most people interpret the second sentence of the Notice as it is intended, rather than as it is expressed. This suggests that people are more logical than many psychologists are inclined to believe.

7 Thinking = Logic + Control

The view that logic can serve as a wide-spectrum language of thought is in marked contrast with conventional views of logic in cognitive science. Paul Thagard [11], for example, in his introductory textbook, "Mind: Introduction to Cognitive Science" (page 45) writes:

> "In logic-based systems the fundamental operation of thinking is ***logical deduction***, but from the perspective of rule-based systems the fundamental operation of thinking is ***search***."

Here he uses the term "rule-based system" to refer to condition-action rule production systems. He then goes on to say that among the various models of thinking

investigated in cognitive science, rule-based systems have "the most psychological applications" (page 51).

Jonathan Baron [1] in his textbook, "Thinking and Deciding" writes, page 4:

> "Thinking about actions, beliefs and personal goals can all be described in terms of a common framework, which asserts that thinking consists of ***search*** and ***inference***. We ***search*** for certain objects and then ***make inferences*** from and about the objects we have found."

Baron associates logic with making inferences, but not with performing search. He also distinguishes thinking from deciding, but restricts the application of logic to the pre-active, inference-making component of thinking.

Both Thagard and Baron fail to recognize that, to be used in practice, logic needs to be controlled. This could be put in the form of a pseudo-equation[5]:

Thinking = Logic + Control.

Here the term "Logic" refers to goals and beliefs expressed in logical form and "Control" refers to the manner in which the inference rules of logic are applied. Control includes the use of forward and backward reasoning. In the case of backwards reasoning, it includes strategies for selecting sub-goals, as well as strategies for searching for alternative ways of solving goals and sub-goals. It also includes the application of inference rules in sequence or in parallel.

Frawley [13] argues that the analysis of algorithms into logic plus control also applies to mental algorithms and helps to explain different kinds of language disorders. He argues that Specific Language Impairment, for example, can be understood as a defect of the logic component of mental algorithms for natural language, whereas Williams syndrome and Turner syndrome can be understood as defects of the control component.

In fairness to Thagard and Baron, it has to be acknowledged that they are simply reporting generally held views of logic, which do not take into account some of the more recent developments of logic in Artificial Intelligence. Moreover, both, in their different ways, draw attention to characteristics of thinking that are missing both from traditional logic and from the simple pro-active model of thinking associated with logic programming. Thagard draws attention to the importance of reactive thinking with condition-action rules, and Baron to the importance of pre-active thinking by inference after search.

8 Conclusions

There isn't space in this paper to discuss all of the arguments that have been made against logic. Instead, I have considered only some of the most important alternatives that have been advocated – production systems, decision theory, and neural networks, in particular.

In the case of production systems, I have argued that condition-action rules are subsumed by maintenance goals in logical form. They are the special case of

[5] In the same sense that *Algorithm = Logic + Control* [5].

maintenance goals in which no forward reasoning is necessary to process observations, and no backward reasoning is necessary to reduce goals to sub-goals.

In the case of decision theory, I have argued that forward reasoning can be used pre-actively to derive possible consequences of candidate actions, and can be combined with any way of deciding between the alternatives. One such possibility is to use decision theory directly to choose a candidate action having maximum expected utility. Another is to compile such decisions into heuristic maintenance goals that approximate the decision-theoretic normative ideal.

In the case of neural networks, I have considered how the low-level logic-programming representation of feed-forward networks, given by *Poole et al*, might be approximated by higher-level logical representations. I have also suggested that such lower-level and higher-level logical representations might interact in a manner similar to the way in which intuitive and deliberative thinking interact in the dual process model. The lower-level representation proposes intuitive answers to problems as they arise, and the higher-level representation monitors and modifies the proposals as time allows.

I have restricted my attention in this paper to the way in which logic can be used to help control the routine, real-time behaviour of an intelligent agent. Except for program transformation, in which a higher-level representation is compiled into a more efficient, lower-level form, I have not considered the wider issues of learning and of revising goals and beliefs. Fortunately, there has been much work in this area, including the work on inductive logic programming, which is relevant to this issue.

Again for lack of space, I have not been able to discuss a number of extensions of logic that have been developed in Artificial Intelligence and that are important for human thinking. Among the most important of these is the treatment of default reasoning and its interpretation in terms of argumentation. Also, I do not want to give the impression that all of the problems have been solved. In particular, the treatment of vague concepts and their approximations is an important issue that needs further attention.

Despite the limitations of this paper, I hope that it will suggest, not only that logic deserves greater attention in Cognitive Science, but that it can be applied more effectively by ordinary people in everyday life.

Acknowledgements

Many thanks to the anonymous reviewers and to Ken Satoh for their helpful comments on an earlier draft of this paper.

References

1. Baron, J.: Thinking and Deciding (second edition). Cambridge University Press(1994)
2. Checkland, P.: Soft Systems Methodology: a thirty year retrospective. John Wiley Chichester (1999)
3. Kahneman, D., Shane F.: Representativeness revisited: Attributive substitution in intuitive judgement. In: Heuristics of Intuitive Judgement: Extensions and Applications, Cambridge University Press (2002)

4. Kakas, T., Kowalski, R., Toni, F.: The Role of Logic Programming in Abduction. In: Gabbay, D., Hogger, C.J., Robinson, J.A. (eds.): Handbook of Logic in Artificial Intelligence and Programming 5. Oxford University Press (1998) 235-324
5. Kowalski, R.: Logic for Problem Solving. North Holland Elsevier (1979)
6. Kowalski, R.: How to be artificially intelligent. http://www.doc.ic.ac.uk/~rak/ (2002-2006)
7. Kowalski, R., Sadri, F.: From Logic Programming towards Multi-agent Systems. Annals of Mathematics and Artificial Intelligence. Vol. 25 (1999) 391-419
8. Poole, D.L.: The independent choice logic for modeling multiple agents under uncertainty. Artificial Intelligence. Vol. 94 (1997) 7-56
9. Poole, D.L., Mackworth, A.K., Goebel, R.: Computational intelligence: a logical approach. Oxford University Press (1998)
10. Smith, E.R., DeCoster, J.: Dual-Process Models in Social and Cognitive Psychology: Conceptual Integration and Links to Underlying Memory Systems. Personality and Social Psychology Review. Vol. 4 (2000) 108-131
11. Thagard, P.: Mind: Introduction to Cognitive Science. MIT Press (1996)
12. Vickers, G.: The Art of Judgement. Chapman and Hall, London (1965)
13. Frawley, W.: Control and Cross-Domain Mental Computation: Evidence from Language Breakdown. *Computational Intelligence,* 18(1), (2002) 1-28
14. d'Avila Garcez, A.S., Broda, K., Gabbay, D.M.: Symbolic knowledge extraction from trained neural networks: A sound approach. Artificial Intelligence 125 (2001) 155–207
15. Holldobler, S. Kalinke, Y. : Toward a new massively parallel computational model for logic programming. In *Proceedings of the Workshop on Combining Symbolic and Connectionist Processing, ECAI 94*, (1994) 68-77
16. Stenning, K., van Lambalgen, M.: Semantic interpretation as computation in non-monotonic logic. Cognitive Science (2006)
17. Nicolas, J.M., Gallaire, H.: Database: Theory vs. interpretation. In Gallaire, H.,Minker, J. (eds.): Logic and Databases. Plenum, New York (1978)