Artificial Intelligence and the Natural World

Robert Kowalski
Imperial College
London

March 2001

1 Introduction

Until recently, the symbolic knowledge representation school was the dominant school of thought in Artificial Intelligence. Its aim was to develop a computational theory of intelligence in which all thought was represented in symbolic, logical form, and intelligence itself was viewed primarily as a goal-oriented problem-solving activity. The world, in which thinking took place, was given little if any attention.

This approach has given rise to many useful computer applications, including expert systems in areas such as law and medicine, and even in mathematical theorem-proving. It also gave rise to the logic programming (LP) paradigm of computing and the computer language Prolog.

In recent years, however, the symbolic knowledge representation school has come under increasing attack. These attacks have come from developers of neural networks, who argue that thinking is better understood in terms of highly parallel, non-symbolic neural networks. They have also come from builders of "situated" robots, who argue that interactions between an agent and the world can be hardwired directly, without the need for symbolic representations. Intelligence, in both of these approaches, is viewed primarily as the ability of an agent to react appropriately and in a timely manner to changes in the environment.

The neural network and situated intelligence approaches have also led to many useful applications, including applications in robotics, speech recognition, vision, and expert systems.

As is typical of work in Artificial Intelligence, the different approaches have not only engineering applications, but also scientific and philosophical implications. Viewed in scientific terms, they provide testable theories about the nature of intelligence. Viewed in philosophical terms, they can be interpreted more broadly as providing models of the human condition in general.

This paper is an informal overview of a recent attempt to reconcile some of the features of the different and opposing schools of thought [Kowalski 1995; Kowalski & Sadri 1996; Davila 1997; Kowalski & Sadri 1999]. It takes from the symbolic knowledge representation school the use of logic to represent thinking in symbolic form. It takes from the neural network and situated robotics schools the need for thinking agents to interact with the world.

For this purpose, it employs an observation-thought-action cycle, in which thinking is only one component and in which observations and actions are dealt with in the other components. Although it focuses on the representation of thinking in logical terms, it

leaves open the possibility that some forms of thinking cannot be represented symbolically at all.

The agent cycle uses abductive logic programming (ALP) [Console, Dupre & Torasso 1991; Kakas, Kowalski & Toni 1992; Denecker & De Schreye 1993; Denecker & De Schreye 1998] for the thinking component. ALP combines normal logic programming for goal-oriented reasoning, with integrity constraints for reactive reasoning and with abduction for generating actions.

As in many other models of intelligent agents [Rao & Georgeff 1992; Shoham 1993], in our agent model beliefs and goals constitute the main components of an agent's internal state. Beliefs include both abductive logic programs and observations from the environment. Goals, on the other hand, include desires, integrity constraints, queries, condition-action rules, commitment rules, active database rules, obligations and prohibitions. The claim that so many different kinds of sentence can be treated in the same way is one of the most radical features of our proposal.

The paper assumes little or no previous knowledge and is based upon a series of examples. Formal details can be found in [Kowalski 1995; Kowalski & Sadri 1996; Davila 1997; Kowalski & Sadri 1999].

2 The observation-thought-action agent cycle

The observation-thought-action agent cycle specifies the interface between an agent and its world. It consists of repeated cycles during which the agent records observations expressed in symbolic form and derives actions also expressed in symbolic form. For simplicity, we assume that the sensory experience of the world, which is recorded by an observation, lies just outside the cycle itself. Similarly, we also assume that the actual change of the world, which is described by an action, also lies on the other side of the interface.

to cycle at time T,
observe any inputs at time T,
think,
select and commit to any actions to perform,
act,
cycle at time T+n.

The agent cycle acknowledges that there is more to intelligence than just thinking, and that thinking itself is not static. Thinking takes place in the dynamic environment of a changing world, which generates observations that need to be assimilated. Thinking, also, is not passive. It generates actions that change the world.

The time T in the cycle is subjective time as measured by the agent's own internal clock. It functions both to limit the amount of time spent thinking in a single cycle and to timestamp observations and actions, so that otherwise identical observations or actions occurring at different times can be distinguished.

For the cycle to be effective, observations must be made and actions must be performed in a timely manner. Therefore, the time, n, between one cycle and the next

must be sufficiently small; and thinking, if it needs more than one cycle to be completed, must be spread over several cycles, interrupted by any necessary observations and actions.

Thinking can take different forms. It can be *reactive,* generating actions in response to external inputs. But it can also be *proactive,* generating actions to accomplish the agent's own internal goals.

Proactive thinking can be represented naturally in logical terms – even in terms of logic programs, using implications of the form:

conclusion if conditions.

These implications are used backwards to reduce goals that match the *conclusion* of the implication to sub-goals that match the *conditions*. Sub-goals can include actions to perform.

Reactive thinking can also often be represented in logical terms – using condition-action rules of the form:

if conditions then action.

These if-then rules are used forwards, given the conditions, to derive candidate actions. We will see later how such logically represented reactive and proactive thinking are combined in abductive logic programming.

However, it may not be possible to express all reactions in logical form. For example, the reactions of a soccer player to the changing state of the game; or of a predator, such as a fox, to the changing location of its prey. Such reactions may need to be hardwired into the agent cycle, bypassing logical thought.

More generally, it may not be possible to express all thoughts in logical form. In particular, if a thought cannot be expressed in ordinary language, then neither can it be expressed in logical terms. For example, the thoughts I might have when remember my first day at school.

However, even though it may not be possible to represent all thoughts in logical form, both reactive and proactive thinking can leave an agent with a choice of candidate actions to perform. The general agent model still applies, and the agent needs a way of deciding which actions to perform.

At one extreme, this choice can be made in advance, by assigning different priorities to different kinds of action, depending perhaps on the way in which the actions have been derived. At another extreme, the choice can be made by a decision theoretic analysis of the utilities and probabilities of the various outcomes of the different actions. It can even be made at random. No matter how the decision is made, in theory, at least, the agent has complete freedom to choose any action.

Choosing an action creates a commitment, in the sense that, once the action is performed, there is no turning back the clock and choosing another action. If, after an

action has been successfully performed, an agent decides that it was a mistake, he may still be able to perform other actions in the future to try to undo the earlier action's undesirable consequences.

Let me illustrate the use of abductive logic programming for proactive thinking with an example.

3 The story of the fox and the crow

Most people are familiar with the Aesop fable of the fox and crow: The crow is sitting in a tree with some cheese in its mouth. The fox is on the ground under the tree and wants to have the cheese.

Being a rational creature, the fox has goals and beliefs, both of which can be expressed in logical form. Among these are the following:

Goal **The fox has 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.

The fox is near the cheese if the crow sings.

The crow sings if the fox praises the crow.

The fox is not only a logician, but also an amateur physicist. In particular, its belief about being near the cheese if the crow sings incorporates not only knowledge about its immediate surroundings (that it is situated on the ground beneath the crow), but also knowledge about the laws of gravity (that the cheese will fall to the ground if the crow sings).

The fox is also an amateur psychologist, a behavioural psychologist in fact. He understands the crow in terms of its input-output behaviour (that it will sing when it is praised).

Not only do the beliefs of the fox have a logical form, but they also have the form of a logic program. Viewed as a logic program, the beliefs function as a collection of procedures[1]:

To have an object
be near the object
and pick up the object.

To be near the cheese

[1] In general, logic programming uses rules of the form *conclusion if conditions* as procedures: to accomplish the *conclusion*, accomplish the *conditions*.

make the crow sing.

To make the crow sing
praise the crow.

Using these procedures, reasoning backwards from the goal, the fox can reduce the goal to a plan of actions that accomplishes the goal. The plan consists of two action sub-goals:

the fox praises the crow and the fox picks up the cheese.

In the past, for many followers of the symbolic knowledge representation school of AI, this would have been the end of the story. Thinking about actions would have been just as good as actually doing them.

However, within the agent framework of the observation-thought-action cycle, the story is more complicated. There are observations to be made and actions to be performed. Moreover the thinking needed to construct the plan might be longer than the time available in a single cycle.

Assume for simplicity, that the fox can perform only one step of backward reasoning in a single cycle. It then takes the fox three cycles to generate the plan. In each cycle, one sub-goal is selected and matched with the conclusion of an implication. The selected sub-goal is replaced by the conditions of the implication.

There are no observations to make during any of these cycles, and no actions that can successfully be performed until the end of the third cycle.

In the first cycle, the fox reduces the original goal of having the cheese to the two sub-goals:

the fox is near the cheese and the fox picks up the cheese.

The second of these two sub-goals is an action that, in theory, is a candidate for attempted execution. Assume, however, that the fox can tell that this is not the right time to do so.[2]

In the second cycle, the fox focuses on the first of its two sub-goals, the sub-goal of being near the cheese, and reduces it to the sub-goal of making the crow sing. The resulting state of its goals is now:

the crow sings and the fox picks up the cheese.

The time is still not right to pick up the cheese, and there are no other feasible candidate actions to select in this cycle.

---

[2] The simplified representation of actions in this example does not indicate the time at which actions take place. For the fox to realise it is too early to try to pick up the cheese, the fox needs to realise that it needs to be near the cheese before it picks it up. We will see later how to associate time with actions and properties that change.

In the third cycle, the fox reduces the sub-goal of making the crow sing to the sub-goal of praising the crow. The resulting state of its goals is now:

the fox praises the crow and the fox picks up the cheese.

The fox now selects and successfully executes the only feasible action:

the fox praises the crow.

As the fox predicted and as the fox can then observe on the fourth iteration of its cycle:

the crow sings.

During this cycle, the only thinking needed is to assimilate the observation. The fact that the observation is predictable from one of its beliefs can be used as evidence to reinforce the degree of conviction in that belief.[3]

At this point, the fox may or may not select and attempt to execute the remaining action in its plan:

the fox picks up the object.

If the cheese has not yet fallen to the ground and the fox tries to pick it up, the action will be premature and will fail. The action will need to be retried in the next or later cycles. However, if the cheese has already fallen to the ground, and the fox waits too long, then the location of the cheese might change.

Consequently, the fox might want to wait[4] and execute the action in a cycle when it observes:

the fox is near the cheese.

This observation, too, is predictable, and confirms that the plan is on course. The fox can then execute the remainder of the plan, in the reasonable expectation that its location relative to the cheese will not change between the observation and the action:

the fox picks up the object.

---

[3] Such reinforcement of beliefs is a higher "meta"-level activity that is not represented explicitly in our simplified agent cycle, which assumes that the only beliefs that change are observations. In the more general, more realistic case, other beliefs can change as well, by means of learning for example.

However, learning by generalisation, in particular, is an error prone activity; and a belief might have to be abandoned if it leads to erroneous conclusions. Keeping track of how often a belief has proved useful in the past and using it to associate a degree of conviction in that belief can help to decide which beliefs to revise when more than one belief contributes to the derivation of an erroneous conclusion.

[4] The fox's cycle and its beliefs are not sophisticated enough for the fox to be able to work this out for himself. A more sophisticated representation incorporating time and integrity constraints is necessary for this.

Finally, the fox can now observe that it has achieved its original goal:

the fox has the cheese.

further reinforcing the beliefs that were used in solving the goal.

The story of the fox and crow illustrates only certain features of the agent cycle. We will explore reactivity in the next section and other features, including the representation of time, in later sections.

4 Reactive agents and condition-action rule production systems

The fox, as we have described him, is proactive and goal-oriented. He observes and takes into account inputs from the environment, but pursues his own goals single-mindedly. Other agents, like the crow perhaps, are purely reactive. Their actions are determined solely by the inputs they receive from the environment. Any appearance they might give of goal-oriented behaviour is simply our interpretation of the results achieved by their actions.

To the extent that observations and actions can be linguistically represented, reactive agents can be modeled by means of production systems consisting of condition-action rules. Production systems were invented in the 1930's by the logician, Emil Post, but were proposed as a computational model of human intelligence by Alan Newell [Newell 1973; Anderson & Bower 1973].

Like our agent model, production systems also employ an observation-thought-action cycle:

to cycle
observe any updates,
think,
select and commit to any actions to perform,
act,
cycle again.

A typical production system, however, does not use time to control the amount of thinking that takes place within a cycle, because it completes all thinking that is needed within a single cycle.

The internal state of a production system consists of two components: a short term memory, which represents the current state of an agent's thoughts, including any observations, and a long-term memory, which consists of condition-action rules. The short term memory typically consists of positive and possibly negative "facts".

Thinking consists in matching facts in the short term memory with the conditions of the condition-action rules and in deriving candidate actions. If more than one action is candidate for execution, then "conflict resolution" is performed, a committed choice is made, and the resulting one or more selected actions are executed, as they are in our own agent model.

Actions are either applied externally to the environment or performed internally on the short term memory. Similarly, observations can be generated externally as inputs by the environment or can be generated internally by actions on the short term memory.

Condition-action rules provide a behavioural model of intelligence. For example, for an external observer, unable to examine the fox's internal mental state, the fox's behaviour can be modelled by a collection of two condition-action rules:

If the fox sees that the crow has cheese, then the fox praises the crow.
If the fox is near the cheese, then the fox picks up the cheese.

In this model, the fox doesn't achieve the goal:

The fox has the cheese.

explicitly by the deliberate application of backward reasoning to reduce goals to sub-goals, but the goal emerges implicitly as the result of the condition-action rules.

However, the fact that a condition-action rule production system might serve as a correct external model of an agent's behaviour, does not mean that the agent itself actually reasons by means of condition-action rules, as we saw in the first version of our story of the fox and crow.

Similarly, from the point of view of an external observer, the condition-action rule:

If the fox praises the crow then the crow sings.

might correctly describe the behaviour of the crow, but it does not follow that the crow itself actually reasons by means of condition-action rules. Unbeknown to the fox, the crow might actually have an explicit goal such as:

The crow is pleased with himself.

and a belief such as:

The crow is pleased with himself if
the crow sings whenever the fox praises the crow. [5]

The crow might use these proactively to exploit the fox's praise for its own purpose, namely to achieve its own goal.

[5] This can also be written in the form we will use later:
The crow is pleased with himself if
if the fox praises the crow then the crow sings.
This form may seem a little confusing, but technically it has the same meaning as the form given above.

We will come back to the relationship between goals and condition-action rules later, when we argue that condition-action rules can be understood as a kind of goal in their own right. But first we will look at beliefs in greater detail.

5 The logic of beliefs

Although much of human thinking is undoubtedly *reactive*, in many other cases, it is genuinely *proactive*. In our general agent model, reactive and proactive thinking are combined in a single general framework.

In the general framework, an agent's mental state consists of two separate components: beliefs, representing the way things are, and goals, representing the way things ought to be. It is the different ways that beliefs and goals can interact that distinguishes between reactive and proactive thinking.

Beliefs are of two kinds: *implications*, which have the form

conclusion if conditions

of logic programs, and *observations*, which have the form of atomic sentences[6]. The entire collection of an agent's beliefs is often referred to as its *knowledge base*.

Implications must have exactly one atomic conclusion, but can have any number of conditions. In particular, if the number of conditions is 0, then an implication represents an unconditional "fact" that is not an observation. This is equivalent to the case where there is a single condition *true*. In such a case, we also write the implication as a simple unconditional conclusion. For example:

0 is a number

in the logic program:

0 is a number
N+1 is a number if N is a number.

Here **N** is a variable. Implicitly, the variable is "universally quantified". In other words, the second belief above should be read as:

For all N, N+1 is a number if N is a number.

In general, variables in implications are implicitly universally quantified.

In the implication:

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

[6] An atomic sentence is one that makes a single statement without logical connectives, such as "and", "or, "if" and "not", and without the quantifiers "all" and "some".

the English phrases:

an object
an animal

are used informally, but precisely, to indicate the first occurrence of a universally quantified variable in an implication. The phrases:

the object
the animal

are used to indicate later occurrences of the same variable in the same implication.

Using symbolic variables, the same implication could be written more formally as:

A has O if A is near O
and A picks up O.

Implicational beliefs are relatively static. However, they can change as the result of theory formation and belief revision, including learning. Except for the comments in the earlier footnote about belief reinforcement, we do not consider such changes of belief in this paper.

Observations, on the other hand, are dynamic by their very nature. They change as new information is obtained about the changing world. They include both externally generated observations, which are outside the agent's control, as well as observations of the results of the agent's own internally generated actions. Observations[7] record concrete situations and therefore have the form of atomic sentences not containing variables.

Both implications and observations can be used to solve atomic sub-goals. Given a conjunction of sub-goals:

atomic-sub-goal & other-sub-goals

where the sub-goal *atomic-sub-goal* is selected for attention, and given an observation or condition-less implication of the form:

conclusion

which matches the atomic sub-goal, backward reasoning solves the sub-goal, without introducing new sub-goals, generating the new conjunction:

other-sub-goals.

Given the same conjunction of sub-goals and an

[7] In ordinary language the term "observation" generally refers to a pre-linguistic experience. For simplicity in this paper, we use the term "observation" to refer to what might normally be called "the record of an observation".

implication of the form:

conclusion if conditions
whose conclusion matches the atomic sub-goal, backward reasoning matches *atomic-sub-goal* and the atomic *conclusion* to generate a new conjunction of sub-goals[8]:

conditions & other-sub-goals.

If the conclusions of several implications:

conclusion if conditions$_1$
conclusion if conditions$_2$
.
.
.
conclusion if conditions$_n$

match the selected atomic sub-goal, then backward reasoning generates a disjunction of alternative sub-goals:

conditions$_1$ *& other-sub-goals or*
conditions$_2$ *& other-sub-goals or*
.
.
. *or*
conditions$_n$ *& other-sub-goals.*

6 Abduction

The combination of relatively static implicational beliefs and inherently dynamic observations can be modelled by means of abductive logic programs. Concepts (predicates) that are relatively static are defined by means of normal logic programs. Concepts that are dynamic have no such definitions and are regarded as *abducible*.

Abduction was identified by the philosopher Charles Pierce as a form of reasoning that explains observations by means of hypotheses. For example, in the context of the belief:

The chicken is dead if the fox kills the chicken.

the observation:

The chicken is dead.

can be explained by the hypothesis:

[8] In the general case, in addition, any instantiation of variables needed to match the *conclusion* and the selected *atomic-sub-goal* is applied to the new conjunction of sub-goals.

The fox kills the chicken.

The hypothesis is not a deductive consequence of the observation and the beliefs, and it might be refuted by new information, such as the fact that the chicken committed suicide.

Abduction has many applications. Here we are concerned mainly with its application for generating candidate plans of actions for achieving goals. For example, given the beliefs of the fox in the story of the fox and the crow, the goal

The fox has the cheese.

can be "explained" by the hypotheses:

praise the crow and pick up the object.

The two concepts of *praising* and *picking up* are abducible in the sense that they have no definitions. The agent's beliefs about such abducible concepts are obtained by its observing the results of its own actions, and the agent records them in its knowledge base in the form of atomic sentences.

Formally, the difference between the classical use of abduction to explain observations and our use of abduction to generate plans is that classical explanations are events or states of affairs that might have held in the past, whereas our explanations are actions that might hold in the future.

Although abducible predicates are not defined, they can be constrained by means of integrity constraints. Integrity constraints have the same form and the same meaning as goals, describing the way the world should be rather than the way it actually is.

For example *picking up* might be constrained by an integrity constraint such as:

If pick up an object then hands are free.

The constraint can be used to eliminate a candidate pick up action if the agent's hands are not free. It can also be used, more actively, to generate the auxiliary sub-goal of making the hands free if they are not free already.

I will argue that goals, integrity constraints and condition-action rules can all be treated uniformly, in the same syntax, with the same semantics, and reasoning in the same forward direction. But first we will look at an example of proactive reasoning without integrity constraints.

7 A proactive robot

The following example illustrates the beliefs a proactive robot might use as a logic program to explore its environment:

explore for period (T1, T2) if clear at T1 & forward at T1+1 &

explore for period (T1+1, T2)
explore for period (T1, T2) if obstacle at T1 & right at T1+1 &
explore for period (T1+1, T2)
explore for period (T1, T+1) if can-not-see at T1 & park at T1+1.

The program consists of three procedures, each dealing with a different case. The first deals with the case where the robot's way ahead is clear, in which case it then moves one step forward and continues to explore. The second deals with the case where there is an obstacle ahead, in which case the robot then turns to the right and continues to explore. The third deals with the case where the robot cannot see, in which case it then simply parks and terminates its exploration.

Here time is made explicit, in order to distinguish between the same property or action at different states of the environment. For example, the property **clear** might hold at time 1, but **obstacle** might hold at time 2 and **clear** might hold again at time 3.

The variable T1 in all three implications stands for the time of the observation, and it is assumed, for simplicity, that the corresponding action is executed in the next time step T1+1. The process of exploration itself takes place over a period of time, from T1 to T2, identified by the pair (T1, T2). A similar representation of time can also be used for the story of the fox and the crow[9].

To illustrate the interaction of the logic program with the environment, assume that the following observations are given dynamically as inputs:

clear at time 1
obstacle at time 2
clear at time 3
can-not-see at time 4.

Here the times, 1, 2, 3, 4, are time stamps, as measured by the agent's own subjective clock. Thus an observation such as **clear at time 3**, for example, means that the observation that the way ahead is clear was made at time 3.

Assume that the initial goal is:

explore for period (1, T2).

Implicitly, the variable **T2** is "existentially quantified". In other words, the goal should be read as:

For some T2, explore for period (1, T2).

Assume, for simplicity, also that the time taken for a single cycle is one time unit and that this is sufficient time to perform all the reasoning that is needed in the cycle.

[9] And, indeed, as noted before, a representation of time is necessary for the fox to realise that it cannot pick up the cheese until it has first praised the crow and the cheese has fallen to the ground.

In the first cycle, at time 1, the agent first records the observation that the way ahead is currently clear. It then uses the three implications to reduce the initial goal to three alternative sub-goals:

clear at time1 & forward at time 2 & explore for period (2, T2) or
obstacle at time1 & right at time 2 & explore for period (2, T2) or
can-not-see at time1 & park at time 2 & T2=2.

It then uses the observation **clear at time 1** to solve and therefore remove the first sub-goal of the first alternative:

forward at time 2 & explore for period (2, T2) or
obstacle at time1 & right at time 2 & explore for period (2, T2) or
can-not-see at time1 & park at time 2 & T2=2.

Assume that the agent has the time-specific knowledge that the conditions **obstacle at time 1** and **can-not-see at time 1** can never be true in the future (because time 1 is past and because the fact that it was clear at time 1 precludes these other alternatives). Assume too that it can use this knowledge to determine that the second and third alternatives are consequently false and therefore unachievable, leaving only the remainder of the first alternative:

forward at time 2 & explore for period (2, T2).

The first sub-goal is an abducible action sub-goal. The second sub-goal can be reduced to sub-goals similarly to the reduction of the initial goal. However, reducing the second sub-goal may take make more time than is available in the cycle. Moreover, its reduction is unnecessary at this time. Better to select and execute the only action that is a candidate, to move **forward at time 2.**

In theory this action can fail for several reasons, for example because of a failure of the robot's motors, because of a faulty observation at time 1 or even because of the unforeseen introduction of a new obstacle between times 1 and 2. Assume, however, that the action succeeds. This results in positive feedback in the form of an observation that the robot has indeed moved forward at time 2. Assume, for simplicity, that the observation is recorded at the same time, 2, that the action is executed.

At the beginning of the second cycle, the current state of the sub-goals is:

explore for period (2, T2)

which is analogous to the situation at the beginning of the first cycle. The remainder of the second cycle, therefore, proceeds analogously to the first cycle, ending with the sub-goals:

right at time 3 & explore for period (3, T2).

As before, we assume that the selected action succeeds, resulting in the sub-goal:

explore for period (3, T2)

at the beginning of the third cycle.

Similarly, the third cycle ends with the sub-goal:

forward at time 4 & explore for period (4, T2).

Assuming that the moving forward action succeeds, the resulting sub-goal at the beginning of the fourth cycle is:

explore for period (4, T2).

After observing **can-not-see at time 4**

the cycle ends with the single subgoal:

park at time 5.

Assuming that the action is selected and succeeds, there are no other sub-goals to solve and the entire cycling process terminates.

Thus, the robot generates the sequence of actions:

forward at time 2
right at time 3
forward at time 4
park at time 5

interleaved with the corresponding sequence of observations.

As a side effect of solving the initial goal and generating these actions, the initially unknown value of the variable T2 is instantiated to 5. Thus, if the initial goal is viewed as a query:

Is there a time T2 such that explore for (1, T2)?

then the instantiation:

T2 = 5

can be viewed as an answer.

8 The logic of goals

Traditional logic, as applied, for example, to the foundations of mathematics, focuses on the representation of beliefs. The axioms of Euclidian and non-Euclidean geometry, for example, formalise different beliefs about the structure of space.

Given a set of axioms, theorems are deductive consequences of the axioms, inescapable consequences, whether we like them or not. However, in many respects, theorems also behave as goals, because if you want to find out whether a given sentence is a theorem, then attempting to show that it is a theorem becomes a goal. Moreover, in mathematical practice, as Lakatos[10] [Lakatos 1977] has argued, the goal of showing that a certain sentence is a theorem can sometimes determine the choice of the axioms and definitions themselves.

Computer database systems, on the other hand, devote equal attention to beliefs and goals. Data represents beliefs; and queries and integrity constraints represent goals.

Queries represent temporary goals that expect an answer. (Such as the instantiation T2=5 in the robot example above.) Integrity constraints represent goals that must always be satisfied and that restrict the updates a database system will accept. For example, the integrity constraint

if a person is an employee
then there is a person who is the person's manager

would be used to reject any input of a new employee into a database if there is no record of the person's manager in the database.

In contrast with beliefs, which can be expressed naturally as implications or as atomic sentences, queries and integrity constraints generally require the full expressive power of first-order logic. However, they can also be expressed in the more restricted form of if-then rules[11]:

if conditions then conclusions.

There can be any number of conditions and conclusions. In particular, if the number of conditions is 0, then the rule represents an unconditional desire, command or obligation. This is equivalent to the case where the conditions are just the single condition *true*, in which case the rule can also be written as a simple unconditional conclusion. For example:

if true then co-operate ***[i.e.* co-operate*]***
if true then for-some time T explore for period (1, T)
[i.e.* for-some time T explore for period (1, T)*]

[10] Lakatos studied the history of Euler's theorem: for any convex polyhedron, the number of vertices and faces together is exactly two more than the number of edges. He showed that the history of the attempted proof of the theorem is largely the history of successive definitions of the concept of polyhedron.

[11] An if-then rule, *if conditions then conclusion* is equivalent to the implication *conclusion if conditions*. However, we write if-then rules and implications differently because of the different ways in which they behave as goals and beliefs respectively. Thus when we write *if conditions then conclusion*, we mean, not only that the *conclusion* holds when the *conditions* hold, but that the *conclusion* **must** hold when the *conditions* hold.

If the number of conclusions is 0, then the rule represents a prohibition. This is equivalent to the case where the conclusions are the single conclusion *false*. For example:

for-all time T if steal at T then false ***[i.e.* do not steal*]***
if take-drugs then false ***[i.e.* do not take-drugs*]***

Notice that if-then rules can include variables, such as the time variable T. These can be "quantified" by such expressions as "**for-some**" and "**for-all**". Such quantification is restricted in various ways that need not concern us here. However, the most interesting feature of these restrictions is that they make it possible to drop explicit expression of the quantifiers, in such a way that, in the context of an entire goal, quantification of the variables can be restored unambiguously.

If-then rules used for the representation of goals are similar to the implications used for the representation of beliefs, but they are written in the forward rather than in the backward direction. Moreover, whereas logic programs have only one atomic conclusion per implication, if-then rules can have several alternative conclusions. It is this ability to represent disjunctive conclusions that gives if-then rules much of the expressive power of full first-order logic.

Writing logic programming implications in the form:

conclusion if conditions

and if-then rules in the opposite direction:

if conditions then conclusions

has the attraction that in both cases the sentences are used for reasoning in the direction in which they are written: If-then rules are used for reasoning forwards from conditions to conclusions. The reasoning is triggered, in most cases, by the assimilation of a new observation that matches one of the conditions. Logic programming implications are used for reasoning backwards from a conclusion to conditions. The reasoning is triggered by the selection of an atomic sub-goal that matches the conclusion.

The search space of goals

In general, at the top-most level, the search space of an agent's goals has the form of the conclusions of an if-then rule, as a disjunction of conjunctions of sub-goals. In the simplest case, these sub-goals are all atomic, as illustrated by the simplified example:

[drop-out & take-drugs] or
[study-hard & get-good-job & get-rich] or
[study-hard & get-good-job & help-the-needy].

Notice that the example is simplified, partly because the temporal sequence of the atomic sub-goals is unspecified.

The example also illustrates a situation in which, during the agent cycle, an agent might need to **select and commit** to an action to perform. Assume for simplicity that both of the atomic subgoals

drop-out, study-hard

are abducible and executable in the current cycle. Then selecting the action **study-hard** has the advantage that it contributes to two of the three alternatives, whereas **drop-out** contributes to only one.

Clearly, deciding what actions to choose is a complex matter that can require as much thought as generating the alternative plans of action in the first place. In this paper, we ignore the issues and techniques involved in such higher level thinking. This higher level, at which thought is given to the lower level objects of thought is called the *meta-level.* The lower level is called the *object level*.

The situation in which the search space of an agent's goals is a disjunction of conjunctions of atomic sub-goals is a simplified case. In many cases some sub-goals have the form of if-then rules, representing integrity constraints or condition-action rules. Such non-atomic sub-goals are often global to the search space of goals, in the sense that they belong to every alternative.

Thus, the non-atomic sub-goal:

if take-drugs then false

added to the search space of goals above, gives the new search space:

[drop-out & take-drugs & if take-drugs then false] or
[study-hard & get-good-job & get-rich & if take-drugs then false] or
[study-hard & get-good-job & help-the-needy & if take-drugs then false].

The variety of goals

Perhaps the most radical feature of our agent model is our uniform treatment of theorems, queries, desires, commands, obligations, prohibitions, integrity constraints and condition-action rules as different kinds of goals. The argument that so many different kinds of sentence can be treated uniformly is based on syntactic, semantic and behavioural grounds.

The syntactic ground is that all of these different kinds of sentences can be represented by means of if-then rules, by including the special cases in which conditions are **true** or conclusions are **false**.

The semantic justification is that they are all concerned with the way things ought to be, rather than with the way things actually are. To be more precise, an if-then rule of the form:

if conditions then conclusions

can be understood as meaning

if the conditions hold then the conclusions must hold as well.

If the conclusion is **false,** then the rule means, in effect, that the conditions can not hold.

The behavioural justification is that it is natural to reason with all of the different kinds of sentence in the same way:

> If an if-then rule has no conditions (or, equivalently, if the condition is simply **true**), then any atomic sub-goal in the conclusions can be selected for attempted solution. The attempt to solve the sub-goal can be made by matching it with an observation or by matching it with the conclusion of an implication.
>
> If the selected atomic sub-goal is an abducible action, then the sub-goal becomes a candidate for execution in any cycle whose time is compatible with the time of the action. If it is selected and successfully executed in such a cycle, it is removed on the next cycle by matching it with the observation that the action has succeeded.
>
> If the action sub-goal is not selected or else it fails, and its time is past, then it is evaluated to false. The entire alternative containing the false sub-goal is also evaluated to false, and the alternative is eliminated from the search space of goals.
>
> Otherwise, whether or not the action sub-goal has previously been selected and failed, if it can still be performed in the future, then it still remains a candidate for selection in the future.
>
> However if an if-then rule has one or more conditions (different from the trivial condition **true**), i.e. if it has the form:
>
> *if condition & other conditions then conclusions*
>
> and some observation matches one of the conditions (say *condition*), then the new if-then rule:
>
> *if other conditions then conclusions*
>
> is derived[12]. In such a case, we say that the observation *triggers* the rule. Any other conditions in the new rule can then be removed either by backward[13] or by forward reasoning, until the conclusion of the rule is derived.

Queries and integrity constraints as goals

[12] As in the case of backward reasoning, any instantiation of variables needed to match the *condition* and the *observation* is applied to the derived if-then rule.

[13] To justify the use of backward reasoning to remove *other conditions* of the if-then rule, it is necessary to assume the only-if halfs of the implications, turning them into if-and-only-if definitions.

Although there have been several different proposals to give semantics to integrity constraints, most of them agree that integrity constraints and queries have the same semantics. The difference between them is mainly pragmatic: Queries are transient goals that are input from the environment and disappear once they have been answered. Integrity constraints, on the other hand, are long-term goals that are generally independent from the environment and persist over the lifetime of the database.

For example, the goal:

if a person is an employee
then there is a person who is the person's manager

could serve either as query or as an integrity constraint. As a query, it asks the question: whether, in the current state of the database, every employee has a manager. The answer is either "yes" or "no". As an integrity constraint, however, it imposes an obligation on the database that in all states of the database every employee must have a manager. In every state, the answer must be "yes".

In traditional database systems, if an update violates an integrity constraint, then the usual remedy is to reject the update. However, in the context of an agent that can perform actions, the agent might be able to employ other measures to ensure that integrity is maintained. The resulting agent behaviour is similar to the behaviour of an active database

Suppose, for example, the database contains the facts (condition-less implications or observations):

Mary is the regular manager of John
John is the regular manager of Mary.

Suppose that it contains the implications:

person1 is the manager of person2 if
person1 is the regular manager of the person2.

person1 is the manager of person2 if
it is not the case that some person is the regular manager of the person2 &
ask Bob "who is the manager of the person2" &
Bob says "person1 is the manager of person2".

Given the observation:

James is an employee

as an update, the integrity constraint is triggered by the observation and the conclusion of the integrity constraint is derived as a goal:

there is a person who is James' manager. i.e.

Find a person such that
the person is James' manager.

There is no regular manager of James in the knowledge base. So backward reasoning using the first implication fails. However, the second implication succeeds in reducing the goal to the two sub-goals:

ask Bob "who is the manager of James" &
Bob says "the person is the manager of James".

The first sub-goal is an action that the agent can perform. Assume that the agent selects the action and successfully executes it. Assume also that a little later the agent observes:

Bob says "Mary is the manager of James".

This solves the second sub-goal and ensures that the integrity constraint has not been violated.

Prohibitions as goals

So far, the only case we have considered is the case where the trigger of an if-then rule is an observation. However, the trigger can also be an atomic sub-goal. In such a case, the rule implements a form of hypothetical reasoning, investigating the consequences of successfully solving the sub-goal.

Such hypothetical reasoning with sub-goals can be used to implement prohibitions. Assume, for example, that the following search space of goals is given:

[drop-out & take-drugs & if take-drugs then false] or
[study-hard & get-good-job & get-rich & if take-drugs then false] or
[study-hard & get-good-job & help-the-needy & if take-drugs then false]

Hypothetical reasoning can be applied in the first alternative using the rule

if take-drugs then false

triggered by the atomic sub-goal:

take-drugs.

The result is to add the conclusion, **false,** to the other sub-goals of the first alternative:

[drop-out & take-drugs & if take-drugs then false & false].

But then the entire alternative is equivalent to **false,** and the entire search space is logically equivalent to the disjunction of the two other alternatives:

[study-hard & get-good-job & get-rich & if take-drugs then false] or

[study-hard & get-good-job & help-the-needy & if take-drugs then false].

In this example, therefore, the non-atomic sub-goal **if take-drugs then false** serves as a prohibition that prevents the execution both of the action **take-drugs** and of any plan that includes the action.

This example is a very simple case. In a more realistic case, an elaborate chain of reasoning might be need to recognize that an action needs to be rejected because it leads to unacceptable consequences.

Condition-action rules as goals

In the proactive robot example, the behaviour of the robot is generated by backward reasoning. However, the same behaviour can also be generated reactively by forward reasoning, using if-then rules, which behave like condition-action rules:

for-all time T
if clear at T then forward at T+1 &
if obstacle at T then right at T+1 &
if can-not-see at T then park at T+1

Consider, as before, the sequence of observations:

clear at time 1
obstacle at time 2
clear at time 3
can-not-see at time 4.

Assume, also as before, that the time taken for a single cycle is one time unit and that this is sufficient to perform all the reasoning that is needed in the cycle. In the first cycle, one step of forward reasoning is sufficient to derive the new sub-goal:

forward at time 2

which is selected and executed in the same cycle. Similarly, the sub-goals:

right at time 3
forward at time 4
park at time 5

are generated, selected and executed in the following cycles.

The resulting behaviour is identical to that of the proactive robot, but is generated with less effort and without an explicit representation of the higher-level exploration goal.

The gap between the logic program and the if-then rules can be reduced by rewriting the logic program in the equivalent form:

explore for period (T1, T2) if

if [clear at T1 then forward at T1+1 & explore for period (T1+1, T2)] &
if [obstacle at T1 then right at T1+1 & explore for period (T1+1, T2)] &
if [can-not-see at T1 then park at T2 &T2=T1+1].

It is possible to show that the rewritten form is equivalent to the original under the assumption that the three cases (**clear**, **obstacle** and **can-not-see**) are mutually exclusive.

Arguably, the if-then rules are not only easier to execute than the logic program, but they are also easier to understand. This is because the logic program needs recursion, whereas the if-then rules do not. The effect of the special-purpose recursion of the logic program is achieved implicitly by the if-then rules by means of the general-purpose recursion of the agent cycle itself.

Both our logic program for exploration and our if-then rules use an explicit representation of time. Both representations store observations, including any actions that are performed in a knowledge base.

However, it is also possible to employ a simplified representation without time stamps.

if clear then forward &
if obstacle then right &
if can-not-see then park

In this representation, observations and actions are ambiguous. None-the-less, in some cases, where a historical record of the past is not required, disambiguation is not necessary. It is sufficient for the agent simply to observe the current state of the environment, match the observation with the appropriate condition and generate and execute the derived action. The observation and the successfully executed action can be forgotten as soon as their time is passed. In such a case, it can be said that the agent has no model of the world, or, equivalently, that the world serves as its own model.

Alternatively, it is also possible to argue that the simplified representation is simply an optimization of the representation using an explicit representation of time. It might be possible to derive one representation from the other under the assumption that it is impossible or unnecessary for the agent to make any use of any record of the past. A similar argument has been made for a similar case in [Kowalski 1992].

Commitment rules as goals

It is necessary to represent time explicitly[14] when it is necessary to reason about past or future actions and states of affairs. The following if-then rule illustrates a situation where it is necessary to reason about the future:

[14] Instead of using explicit time points and time periods in the syntax of the representation language, it is also possible to use “modal operators”, such as “in the past”, “in the future”, “in the next situation” etc. In the modal approach, different times correspond to different possible worlds, which belong to the semantics rather than to the syntax of the language.

if **an agent asks at T1 "do an action at T3"**
then **confirm at T2 "can-do the action at T3" &**
do the action at T3 & T1 < T2 < T3

The rule is triggered by an observation that some (other) agent asks for an action to be performed in the future. The observation occurs at time T1 and the action is to be performed at the future time T3. Forward reasoning derives two actions that need to be performed. The first is to confirm at some time T2, after T1 and before T3, that the action can indeed be performed, and the second is to actually do it at the requested time T3. The rule can be made more realistic by adding extra constraints on the time T2, such as:

T2 < T1+5

and by adding an extra case to deal with the situation where the agent decides that it is unable to perform the action.

If-then rules whose conclusions contain action to be performed in the future are called *commitment rules* in the Agent0 agent model [Shoham 1993]. The example above illustrates that commitment rules are a special case of goals in our model.

Notice that the terms **agent, action, T1, T2, T3** in this example are all implicitly quantified variables. It is possible to restore the quantifiers automatically by means of an algorithm. The result of the algorithm is a commitment rule with explicit quantifiers:

for-all agent, action, T1, T3
if **an agent asks at T1 "do an action at T3"**
then **for-some T2**
confirm at T2 "can-do the action at T3" &
do the action at T3 & T1 < T2 < T3

According to the algorithm, the variable T2 is treated differently from the other variables because it occurs in the conclusions but not in the conditions of the rule.

How to have your cheese and eat it

In the following example, we consider the case of a crow that is able to reconcile its instinctive desire to sing whenever it is praised with the goal of eating the cheese.

The crow is reactive, because it has a goal:

In modal temporal logic, the structure of time, for example whether it is linear or branching, is reflected in the structure of the possible worlds. In our approach, using an explicit representation of time in the syntax, time is both linear and branching. It is linear in the sense that there is only one time line. But it is also branching because incomplete knowledge of the time line means that there are alternative branches from the known part of the time line, both in the future and in the past, corresponding to different ways in which incomplete knowledge can be completed.

if praise at T **then sing** at T' **&** T <_ T' <_ T+3

which is like a condition-action rule, except that the action of singing, which is triggered by an observation of being praised, is constrained to take place within three time units after the observation.

It is also proactive, because it has an explicit goal:

eat at T'' **&** 1 <_ T'' <_ 3

which has to be satisfied within the period of time (1, 3).

It has beliefs, which it can use backwards to try to accomplish its goals:

Beliefs **have cheese** at 1

have cheese at T2 **if have cheese** at T1 **&**
not (swallow at T **&** T1 <_ T <_ T2) **&**
not (sing at T **&** T1 <_ T <_ T2)

eat at T **if have cheese** at T **& swallow** at T

The second belief is an axiom of inertia: If the crow has cheese then it continues to have the cheese unless and until it swallows or sings [Kowalski & Sergot 1986]. The third belief serves as a procedure for eating the cheese.

Assume that the crow has the following goals at time 1:

(if praise at T **then sing** at T' **&** T <_ T' <_ T+3**)**
& (**eat** at T'' **&** 1 <_ T'' <_ 3**)**

Assume also that the crow receives the following observation at time 1:

praise at 1

The observation triggers the if-then rule and adds a candidate singing action to the current state of its goals:

(if praise at T **then sing** at T' **&** T <_ T' <_ T+3**)**
& (**eat** at T'' **&** 1 <_ T'' <_ 3**)**
& (**sing** at T'**&** 1 <_ T' <_ 4**)**

The singing action is now a candidate for execution. However, if the crow can think fast enough, then it can continue thinking in the same cycle, before it needs to make a decision about what if any actions to perform and when to perform them.

The only possibility available for thinking is to reason backward from the goal of eating. First, it simply reduces the goal of eating to the sub-goals of having the cheese and swallowing it:

(**if praise** at T **then sing** at T' **&** T <_ T' <_ T+3)
& (**have cheese** at T'' **&** **swallow** at T'' **&** 1 <_ T'' <_ 3)
& (**sing** at T' **&** 1 <_ T' <_ 4)

It then solves the sub-goal of having the cheese by using the axiom of inertia:

(**if praise** at T **then sing** at T' **&** T <_ T' <_ T+3)
& (**have cheese** at T1 **&** **not** (**swallow** at T **&** T1 <_ T <_ T'')
& **not** (**sing** at T **&** T1 <_ T <_ T'')
& **swallow** at T'' **&** 1 <_ T'' <_ 3) **&** (**sing** at T' **&** 1 <_ T' <_ 4)

It solves the sub-goal of having the cheese at time T1 by using the fact that it has the cheese at time 1. This not only solves the sub-goal, but it instantiates the previously unknown time T1 to the value 1 in the other goals:

(**if praise** at T **then sing** at T' **&** T <_ T' <_ T+3)
& (**not**(**swallow** at T **&** 1<_T<_ T'')
& **not**(**sing** at T **&** 1<_T<_ T'')
& **swallow** at T''**&** 1<_T''<_3) **&** (**sing** at T' **&** 1 <_ T' <_ 4)

The crow now has two candidate actions from which to choose, swallowing and singing. However, assuming it still has enough time left in the current cycle, it can also reason hypothetically. Reasoning forward by means of the two sub-goals[15]

sing at T' **&** (**if sing** at T **&** 1<_T<_ T'' **then false**)

derives:

if 1<_T'<_ T'' **then false**

which, in turn, implies[16]:

T''< T'.

Thus, the crow can reason if the action of swallowing takes place before the action of singing, then both goals can be solved compatibly.

Conclusions

The agent model described in this paper aims to reconcile the traditional logic-based view of intelligence with the contrary view that intelligence is best understood in

[15] A sub-goal of the form *not P* is shorthand for *if P then false*.
[16] This conclusion can also be derived by forward reasoning, using the "if-then rule":
T''<_ T' **or** T'< T'', which has no conditions but a disjunctive conclusion, as an additional global sub-goal. Note that this integrity constraint formalises the requirement that time is linear. However, given only limited information about the linear ordering, there may be many possible models, in each of which time is linear, but which together represent the different possibilities along different branches of time.

terms of appropriate reactions to changes in the environment. The model uses abductive logic programming for the thinking component of an agent, and an observe-think-act agent cycle to interface the thinking agent with the changing environment.

I have argued that, in addition to abductive logic programming, the agent model includes and unifies many other intelligent methods, including the use of condition-action rules, Agent0 commitment rules, integrity checking, active database rules, obligations and prohibitions.

In our agent model, logic contributes only to the thinking part of an intelligent agent. An intelligent agent also needs to be able to interact with the environment.

I believe that, although such a view of logic is much more restricted than the traditional view, it still leaves logic as an important contributor, not only to Computing, but also to Human and Artificial Intelligence.

Acknowledgements

I am grateful to Alberto Pettorossi for his comments on an earlier draft of this paper.

References

Anderson, J. & Bower, G. (1973). *Human Associative Memory*. Washington, D.C.: Winston.

Console L.D., Dupre, T. & Torasso, P. (1991). On the relationship between abduction and deduction. *Journal of Logic and Computation, 2(5),* 661-90.

Davila, J. (1997). *Agents in logic programming*. (PhD thesis). London, UK: Imperial College of Science, Technology and Medicine.

Denecker, M. & De Schreye, D. (1993). Representing Incomplete Knowledge in Abductive Logic Programming. In *Proc. International Symposium on Logic Programming*, (pp. 147-163). Cambridge, Mass: MIT Press.

Denecker, M. & De Schreye, D. (1998). SLDNFA: an Abductive Procedure for Abductive Logic Programs. *The Journal of Logic programming, 34 (2),* 111-67.

Kakas, A., Kowalski, R. & Toni, F. (1992). Abductive Logic Programming. *Journal of Logic and Computation, 2 (6),* 719-70.

Kowalski, R. (1992). Database Updates in the Event Calculus. *Journal of Logic Programming, 12,* 121-46.

Kowalski, R. (1995). Using meta-logic to reconcile reactive with rational agents. In K. Apt and F. Turini (eds), *Meta-Logic and Logic Programming* (pp. 227-42). Cambridge, Mass: MIT Press.

Kowalski, R. & Sadri, F. (1996). Towards a Unified Agent Architecture that Combines Rationality with Reactivity. *Proceedings of International Workshop on Logic in Databases,* Springer-Verlag.

Kowalski, R. & Sadri, F. (1999). From Logic Programming towards Multi-agent Systems. *Annals of Mathematics and Artificial Intelligence, 25,* 391-419.

Kowalski, R. & Sergot, M. (1986). A Logic-based Calculus of Events. *New Generation Computing, 4 (1),* 67-95.

Lakatos, I. (1977). J Worrall and E G Zahar (eds), *Proofs and Refutations : The Logic of Mathematical Discovery*, Cambridge University Press.

Newell, A. (1973). Production Systems: Models of Control Structure. In W. Chase (ed), *Visual Information Processing* (pp. 463-526). New York: Academic Press.

Rao, A. S. & Georgeff, M.P. (1992). An abstract architecture for rational agents. In C. Rich, W. Swartout, and B.Nebel (eds), *Proceedings of Knowledge Representation and Reasoning (KR&R-92),* (pp. 439-49.

Shoham, Y. (1993). Agent-oriented programming. *AI Journal, 60, (1),* 51-92.