SRC TR 85-35

Real-Time Default Reasoning, Relevance, and Memory Models

By

J. Drapkin, M. Miller
and
D. Perlis

REAL-TIME DEFAULT REASONING, RELEVANCE, AND MEMORY MODELS

J. Drapkin M. Miller D. Perlis

Computer Science Department
University of Maryland
College Park, MD

ABSTRACT

We describe a working example of mechanical default reasoning that is rather robust, in that it can detect a conflict between a default conclusion and another assertion (or conclusion), can (under suitable conditions) decide between them, and can maintain this decision indefinitely until overridden by information to do so. Our model contains five key elements: STM, LTM, ITM, QTM, and RTM. STM, LTM, and ITM are standard parts of cognitively-based models of memory. QTM is a technical device that controls the flow of information into STM, and RTM is the repository of default resolution and relevance.

Support for the research reported herein from both the Systems Research Center at the University of Maryland [NSF contract #OIR-8500108] and the Martin Marietta Corporation is gratefully acknowledged.

INTRODUCTION

Most of the AI systems built today are designed to solve one problem only. That is, the system is turned on, labors away for some period of time, then spits out the correct answer. It is then turned off, or works on another problem with no knowledge of its past history. In contrast to this type of system, we are interested in building an all-purpose system -- a system with a life-time of its own.

We envision an autonomous system that can act in an ever-changing world. It must relate one problem to another, and survive in a real-time environment. The reasoning agent must be able to monitor itself, having a sense of what it is doing, what it has done, what it is going to do, i.e.; some notion of temporality. In order to achieve this, the system must be able to do commonsense reasoning, that is, real-time shallow reasoning. We postulate an attention mechanism for this purpose.

THE MODEL

The basic memory model as currently implemented is comprised of three parts: Short Term Memory (STM), Intermediate Term Memory (ITM), and Long Term Memory (LTM). STM is meant to represent one's current focus of attention. It is a small subset (currently fixed at eight "elements" -- though its size is easily changed) of all beliefs in existence; only those pieces of information currently being accessed. Information enters STM via one of four means: direct observation, modus ponens (MP), LTM, or ITM. Currently direct observation is modeled by allowing the user of the system to simply assert a fact. MP (applied in the following form: from Ac and (x)(Ax --> Bx) Bc can be inferred) is used to bring Bc into STM if Ac and (x)(Ax --> Bx) are already contained in STM. Facts from LTM are brought into STM by association; that is, when a fact in STM, A, triggers a fact in LTM, B, that new fact, B is brought into STM. Facts that enter via ITM will be described momentarily.

STM is structured as a FIFO queue. Since STM's size is limited, as new facts are brought into STM, old ones must be discarded. Those discarded facts flow directly into ITM. Hence, ITM can be regarded as a chronological view of all past thoughts, where those most recently entered will be those most easily accessed. ITM, then, is essentially a record of beliefs. It allows readily accessible feedback about what has been accomplished with the intent of providing a basis for the system's learning capabilities.

LTM is implemented as a series of tuples. It is the largest of the memory stores, and serves to hold the bulk of stored beliefs. The idea behind LTM is that one holds one's beliefs as a series of association pairs/tuples. Thinking about a subject/item P triggers (possibly many) past associations. These associations would then be brought into STM, where MP might be applied, or further associations made. A typical LTM entry looks like:

(T1,..., Tn, B),

where the Ti and B are facts/beliefs represented by logical formulae. The presence of T1, ..., Tn in STM "triggers" B to appear in STM. For any given item, there may be many associations that are triggered. Currently LTM is fixed. A hope of ours is to model learning by adjusting and/or adding to LTM based on knowledge accumulated in ITM.

As an example of the model in action, suppose LTM contained the following information (where variables are universally quantified) :

LTM: < bird(x), bird(x) --> flies(x) >

< penguin(x), penguin(x) --> bird(x) >

< penguin(x), penguin(x) --> -flies(x) >

The first of these tuples indicates that if we have in STM the fact "bird(Tweety)" (i.e. Tweety is a bird), the rule all birds fly (bird(x) --> flies(x) for all x) will be brought into STM. Let us suppose that STM can hold up to four elements and that it currently contains

the single element:

STM: penguin(Tweety).

This fact matches with two LTM entries. The corresponding entries of the tuples would then enter STM, resulting in:

STM: penguin(x) --> -flies(x)

penguin(x) --> bird(x)

penguin(Tweety)

MP would then be applied twice, resulting in:

STM: -flies(Tweety)

bird(Tweety)

penguin(x) --> -flies(x)

penguin(x) --> bird(x)

ITM: penguin(Tweety)

Note that because STM can hold only four elements, the belief "-flies(Tweety)" has pushed the belief "penguin(Tweety)" out of STM, and into ITM.

Because we must deal in a world about which we have only partial knowledge, conclusions must frequently be drawn, which may later be retracted in the face of new information. For example, we may be told that Tweety is a bird. After some thought, we may then infer that Tweety can fly (believing, for example, that (x) (bird(x) --> flies(x))). We later discover, however, that in fact, Tweety cannot fly (he is a penguin, or an ostrich, or one of his wings is broken, etc.). We must then retract the former conclusion that Tweety can fly. This non-monotonicity is a common phenomenon in human reasoning, (see McCarthy [1980] for a discussion of this) and, therefore, should be represented in our model.

COMMON SENSE INCONSISTENCY

We would like to have the ability to use a rule such as A --> B to conclude B, given A, and later, in the face of new evidence, be able to retract our belief in B. Such is a type of default mechanism. A somewhat standard way of dealing with this is to use a rule such as, "if A and it is consistent to believe B, then conclude B ". This is called a default rule.

There is a question as to whether or not these default rules (found in LTM) should be encoded as such. Is there any real difference between a 'normal' rule and a default rule? All rules, it seems, are actually defaults, since we can never really be sure of anything in the real world (although some rules may perhaps be 'stronger' than others). It is then tempting to use a more "brute force" method of encoding these defaults: simply encode the rule as "If A, then conclude B", with no "unless" (it is not consistent to do so) condition. One would then proceed as normal in the inference process until a direct contradiction were found in STM. At this point, something would have to be done to solve the inconsistency.

As an example of this default mechanism in action, consider the following state of affairs, again with STM fixed at four:

STM: bird(Tweety)

ITM: < ... empty ... >

LTM: <bird(x), bird(x) --> flies(x) >

<ostrich(x), ostrich(x) --> -flies(x) >

The fact that Tweety is a bird would trigger the rule that all birds fly, resulting in:

STM: bird(x) --> flies(x)

bird(Tweety)

An application of MP would then leave:

STM: flies(Tweety)
bird(x) --> flies(x)
bird(Tweety)

Suppose we then discover (through direct observation, or some other means) that Tweety is an ostrich. We would then have:

STM: ostrich(Tweety)
flies(Tweety)
bird(x) --> flies(x)
bird(Tweety)

This new fact would then trigger the rule from LTM that ostriches do not fly:

STM: ostrich(x) --> -flies(x)
ostrich(Tweety)
flies(Tweety)
bird(x) --> flies(x)

ITM: bird(Tweety)

Again MP may be applied, resulting in:

STM: -flies(Tweety)
ostrich(x) --> -flies(x)
ostrich(Tweety)
flies(Tweety)

ITM: bird(x) --> flies(x)

bird(Tweety)

Note at this point that STM contains the belief that Tweety does not fly, as well as the belief that in fact Tweety does fly. Is this a problem? We think not. We would like to be able to say that the fact that Tweety flies was concluded BY DEFAULT, that is, in the absence of any information to the contrary. Now given the additional information that, in fact, Tweety is an ostrich, we would like to be able to retract our belief that Tweety flies, and instead conclude that Tweety does in fact not fly.

McDermott and Doyle [1980], and Reiter [1980] have done recent work in this area. A system of beliefs is inconsistent if there exists an x such that both x and -x are believed as true in that system. This is generally considered undesirable, because anything can then be shown true through the use of x and -x in the proof. Thus each has attacked the problem of maintaining consistency within the entire belief system.

In order to show that A is consistent with a given body of beliefs, it is necessary to show that -A is not known nor derivable from the database of beliefs. This proves to be computationally tractable only under severe restrictions, such as the closed world assumption (see, for example, Reiter [1978]). The closed world assumption states that for an arbitrary predicate P, if it is consistent to assume -P, then assume -P. That is, failure to find P true, allows one to conclude its negation.

Our approach is to let the inconsistency go; that is, concluding Tweety flies, then later concluding that Tweety does not fly is acceptable, AS LONG AS BOTH BELIEFS ARE NOT SIMULTANEOUSLY FOUND IN STM. This may be acceptable, as one will not be able to use both conflicting beliefs in a given line of reasoning without both beliefs being present in STM. This perhaps corresponds to what actually happens in the human mind. If, however, both ARE in STM, we want to be able to decide which of the two should be our true belief. Only in this situation need we worry about resolving the inconsistency. Since STM is small, we will always be able to determine quickly and

easily whether such an inconsistency exists; hence a tractable solution in this setting.

A possible solution to this problem of two conflicting beliefs is to store information in LTM to decide which belief should be held. For instance, we could encode a Wins predicate that determines, given two conflicting beliefs, which should be held as true, and which should be discarded. So, for example, we might have something like: Wins(C, A, B), meaning, under conditions C, A should be concluded over B. As a specific example, consider Wins(x directly observed & -x concluded by default, x, -x). This says that if we have both x and -x, where x was directly observed, and -x was concluded by default, then we should maintain our belief in x and discard our belief in -x.

Once an inconsistency is found and subsequently resolved, we would like to maintain the information that such a problem arose, so that one need not repeatedly go through this process of resolution. It is conceivable however, that the system could "flip-flop" between two sets of conflicting beliefs, with a clash never actually occurring in STM. For this reason, the apparent indecision might never be detected.

There are several ways of handling this. Links could be stored in LTM that relate the two inconsistent beliefs. In this way, if an association caused -x to be brought into STM, we could instead bring in x, realizing that x was our preferred belief over -x. Another possible way of handling this is to maintain this information in a new store called Relevant Term Memory (RTM).

RELEVANCE

As explained earlier, STM is very small. As such, at any given time it contains very little information, specifically only that information that is being immediately used. It seems reasonable to postulate a store of relevant information, which is, on the one hand, larger than STM, and on the other hand, smaller than ITM. Though ITM contains a

history of past inferences, it contains too much information for this sort of use. Not only does it quickly become very large, and hence difficult to search for information, but it will include both relevant and no longer relevant information.

There are several ways to model RTM. Since RTM is meant to contain information that is somehow 'relevant', it is reasonable to employ some sort of decay mechanism, whereby the use and/or reference of a belief in STM gives that belief x more units of time to exist in RTM. In this way, beliefs which are continually in focus will remain in the relevant pool. Since the idea is to encapsulate a wider selection of beliefs than that contained in STM, it seems reasonable to expect beliefs to remain in RTM longer than they actually remain in STM.

Another possible way to model RTM is through the use of the Wins predicate previously mentioned. Any time the Wins predicate is used to decide on belief A over another belief B, A should be stored in RTM. Although at first glance this might seem to do the trick, one should note that sooner or later one will move on to a new line of thinking/concentration. At this point it may very well be the case that A is no longer relevant, and hence ought to no longer remain in RTM. However, one would still like to keep the information that A was a belief that 'won' over another belief. This fact could be stored in LTM, as previously suggested.

Because of the aforementioned arguments, it seems reasonable to postulate a model of RTM incorporating both the decay mechanism and the Wins predicate. Experiments clearly are called for in order to test these ideas about the function of such a Relevant Term Memory.

INNOVATIONS

In what follows we describe more precisely what we have done in the way of initial experimentation in this direction. We will also present a detailed example of the use of our

system in dealing with an extended session of mechanical default reasoning in one domain, namely, that of the flying ability (or lack thereof) of various type of birds when the information about the birds in question is of non-uniform degree of specificity. For instance, a bird may be stated to be just that, with no further information about its species, in which case it is desired that a default conclusion be reached to the effect that the bird can fly. Or a bird (the same or another bird) may be stated (say at a later time) to be an ostrich, which should trigger a chain of inferences with the ultimate effect of blocking or reversing the conclusion that it can fly.

We have elected to implement our default mechanism as follows. The first thing that we have done is to add an extra term to each statement in STM. That is, instead of simply noting, say, "bird(Tweety)", we place "(bird(Tweety) justification)" in STM. This second term is intended to indicate just how some fact has found its way into STM. For example, it might be by direct observation that the system believes that Tweety is a bird.

Now that we have some justification for (tentatively) believing a fact, we also have an idea of how to resolve inconsistencies in some cases. Suppose for example that we know that stm currently contains "(bird (Tweety) justification1)". After some reasoning, STM is able to conclude that Tweety can fly. Later we observe that Tweety is a penguin; i.e., STM will contain "(penguin(Tweety) OBSERVATION)". Again, after some reasoning, it is discovered that Tweety cannot fly. Given some reasonable rules for conflict resolution, we now have a mechanism in place that puts a wins predicate in front of "(flies(Tweety) justification2)" and at the same time places "-(flies(Tweety)) in RTM. As RTM contains relevant information, we can rely on RTM to suppress false information from finding it's way back into STM. For example, in the above case, "(flies(Tweety) justification3)" will not reenter STM unless justification3 is sufficient to remove "-(flies(Tweety)) from RTM (e.g., Tweety is found to be a rare type of penguin that does indeed fly).

EXAMPLE OF DEFAULT REASONING

We present below an example of how the current experimental system behaves when presented with partial information at various times. In general, certain categories, such as birds, are by default assumed to fly unless that conclusion is suppressed by a fact in RTM. If such suppression does occur, future deduction about Tweety will be subject to this condition. In other words, it is considered "relevant" to bring the old conflict resolution (in favor of Tweety NOT flying) to bear in future situations.

In this example we have elected not to include the justifications that would normally be found in STM. It should be clear in the example how all facts have found their way into STM.

The initial configuration that the example below is based on consists of certain facts that can be retrieved from LTM, including

(1) bird(x) -> flies(x)

(2) penguin(x) -> bird(x)

(3) penguin(x) -> -flies(x)

(4) ostrich(x) -> -flies(x)

Also, initially, STM consists of the facts

bird(Tweety)

bird(Chirpy)

As a consequence, after a round of inferences is performed, the next state of STM will consist of

bird(Tweety)

bird(Chirpy)

bird(x) -> flies(x)

The following extended list of consecutive states of STM illustrates the system's performance. We begin by repeating the above states. In each stage we mark the new items with a star (*); they will appear at the bottom within each stage. Keep in mind that the maximum size of STM at any time is eight facts, and that oldest facts are removed to allow new ones to enter.

STM state		STM contents
(1)		bird(Tweety)
		bird(Chirpy)
(2)		bird(Tweety)
		bird(Chirpy)
	*	bird(x) -> flies(x)
(3)		bird(Tweety)
		bird(Chirpy)
		bird(x) -> flies(x)
	*	flies(Tweety)
	*	flies(Chirpy)
(4)		bird(Tweety)
		bird(Chirpy)
		bird(x) -> flies(x)
		flies(Tweety)
		flies(Chirpy)

* penguin(Tweety) [outside information supplied]

(5) bird(Tweety)

bird(Chirpy)

bird(x) -> flies(x)

flies(Tweety)

flies(Chirpy)

penguin(Tweety)

* penguin(x) -> -flies(x)

* penguin(x) -> bird(x)

(6) bird(Chirpy) [STM truncates to size 8]

bird(x) -> flies(x)

flies(Tweety)

flies(Chirpy)

penguin(Tweety)

penguin(x) -> -flies(x)

penguin(x) -> bird(x)

* -flies(Tweety)

(7) flies(Chirpy)

penguin(Tweety)

penguin(x) -> -flies(x)

penguin(x) -> bird(x)

-flies(Tweety)

* wins(-flies(Tweety))

* loses(flies(Tweety)) [RTM records flies(Tweety) as suppressed]

* bird(Tweety)

(8) penguin(Tweety)

penguin(x) -> -flies(x)

penguin(x) -> bird(x)

-flies(Tweety)

wins(-flies(Tweety))

loses(flies(Tweety))

bird(Tweety)

* bird(x) -> flies(x)

(9) penguin(x) -> -flies(x)

penguin(x) -> bird(x)

-flies(Tweety)

wins(-flies(Tweety))

loses(flies(Tweety))

bird(Tweety)

bird(x) -> flies(x)

* bird(Oscar) [outside information supplied]

(10) penguin(x) -> bird(x)

-flies(Tweety)

wins(-flies(Tweety))

loses(flies(Tweety))

bird(Tweety)

bird(x) -> flies(x)

bird(Oscar)

* flies(Oscar)

(11) -flies(Tweety)

wins(-flies(Tweety))

loses(flies(Tweety))

bird(Tweety)

bird(x) -> flies(x)

bird(Oscar)

flies(Oscar)

* ostrich(Oscar) [outside information supplied]

(12) wins(-flies(Tweety))

loses(flies(Tweety))

bird(Tweety)

bird(x) -> flies(x)

bird(Oscar)

flies(Oscar)

ostrich(Oscar)

* ostrich(x) -> -flies(x)

(13) loses(flies(Tweety))

bird(Tweety)

bird(x) -> flies(x)

bird(Oscar)

flies(Oscar)

ostrich(Oscar)

ostrich(x) -> -flies(x)

* -flies(Oscar)

Now the same kind of cycle will begin with Oscar, in which flies(Oscar) and -flies(Oscar) will resolve into a suppression of flies(Oscar) for future reference. Note that flies(Tweety) has not reappeared in STM, nor will it do so again. We continue the exam-

ple.

(14) bird(x) -> flies(x)

bird(Oscar)

flies(Oscar)

ostrich(Oscar)

ostrich(x) -> -flies(x)

-flies(Oscar)

* wins(-flies(Oscar))

* loses(flies(Oscar)) [flies(Oscar) now suppressed]

(15) bird(Oscar)

flies(Oscar)

ostrich(Oscar)

ostrich(x) -> -flies(x)

-flies(Oscar)

wins(-flies(Oscar))

loses(flies(Oscar))

* penguin(Tweety) [outside information supplied again]

(16) ostrich(x) -> -flies(x)

-flies(Oscar)

wins(-flies(Oscar))

loses(flies(Oscar))

penguin(Tweety)

* bird(x) -> flies(x)

* penguin(x) -> -flies(x)

* penguin(x) -> bird(x)

(17)

wins(-flies(Oscar))

loses(flies(Oscar))

penguin(Tweety)

bird(x) -> flies(x)

penguin(x) -> -flies(x)

penguin(x) -> bird(x)

* -flies(Tweety)

* bird(Tweety)

(18)

wins(-flies(Oscar))

loses(flies(Oscar))

penguin(Tweety)

bird(x) -> flies(x)

penguin(x) -> -flies(x)

penguin(x) -> bird(x)

-flies(Tweety)

bird(Tweety)

Nothing new is brought into STM now. There are no facts in LTM that can be associated with current STM information that are not already present in STM, and none of the conceivable inferences lead to anything new except for flies(Tweety), which is suppressed from previous experience.

We can now take a look at RTM after the above session has been run. It is important to note that RTM contains correct and relevant information about all that has occurred.

RTM contents

bird(Tweety)

bird(Chirpy)

penguin(Tweety)

-(flies(Tweety))

bird(Oscar)

ostrich(Oscar)

-(flies(Oscar))

SUMMARY

Believing that attention is what keeps computation tractable in a reasoning being, and that thinking is basically rule-based, involving only simple steps readily expressed in logic, we have proposed a model of memory for our reasoning being. It is our contention that apparently purposeful and coherent behavior can result from this model. While our work will continue in this general area for some time to come, as the problems addressed are open ended in nature, we nevertheless feel that an interesting first plateau has been reached from which to survey the task ahead.

BIBLIOGRAPHY

McCarthy, J. [1980] Circumscription--a form of non-monotonic reasoning. Artificial Intelligence, 13 (1,2), pp. 27-39.

McDermott, D. and Doyle, J. [1980] Non-monotonic logic I. Artificial Intelligence, 13 (1,2), pp. 41-72.

Reiter, R. [1978] On closed world databases. In: Logic and Databases, Gallaire, H. and Minker, J. (eds.), Plenum, pp. 55-76.

Reiter, R. [1980] A logic for default reasoning, Artificial Intelligence 13 (1,2), pp. 81-132.