SRC TR 86-21

A Memory Model for Real-Time Commonsense Reasoning

by

J. Drapkin, M. Miller, and D. Perlis

A MEMORY MODEL FOR REAL-TIME COMMONSENSE REASONING

J. Drapkin M. Miller D. Perlis

Computer Science Department
University of Maryland
College Park, MD

Abstract

This paper reports on a significantly improved version of a system for real-time commonsense reasoning previously sketched in [7]. The research is based on the hypothesis that a simple conceptual architecture for memory suffices for a very broad range of behaviors in the commonsense world. In particular, we describe a working example of mechanical reasoner that is rather flexible and robust, in that it can tolerate some inconsistencies; can work on goals; can "ruminate" without goals; can forget; can remember; can make assumptions and subsequently 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.

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

0. 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 (hopefully 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 a system with a life-time of its own, that is, one whose behavior significantly depends on its continued dealings with a variety of issues. In this respect it can be considered as an experimental approach to Nilsson's notion of a computer individual [6].

Thus the issue is the following: a vast amount of data is to be handled in an efficient way, including sorting out conflicts when things seem to go wrong. In particular, typical settings must be distinguished from exceptional ones, even if only after the fact. In other words, we argue that virtually all artificial intelligence ventures will sooner or later have to address the issue of default reasoning.

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. We will explain these in detail shortly.

The rest of this paper is organized as follows: In section 1, the overall philosophy of our approach is sketched. Then, in section 2, we give a more detailed look at the various portions of the memory model. Section 3 then discusses default reasoning, and presents an extended example. Section 4 concludes with a summary, and describes future work.

B. *Discussion*

This arrangement has been borrowed from cognitive psychology, as well as from the production system model of A.I., with liberal alteration to suit our purposes. The chief purpose of STM is to allow access to a very large database (LTM), yet not suffer an exponential explosion of inferences.

Several points are worth making at the outset concerning general features of the model. Although we will not explore all of them in depth here, brief mention will serve to illustrate our research intentions.

First, in most of our work, we have limited the size of STM to eight "elements". By this we mean that eight formulae can be held in STM at any one time. (Experimentation has shown that a size of roughly eight is the smallest that has led to effective task-oriented behavior over several domains, and that larger sizes have offered no advantage. This is in surprising accord with psychological data on human short-term memory which has been measured to hold seven plus-or-minus two "chunks" of data at any one time [5].) In the examples which we present later in this paper, however, we use somewhat smaller STM sizes, for ease of illustration.

Second, LTM can hold inconsistent data without the usual disastrous consequences of customary inference systems. That is, as long as a direct contradiction does not occur in STM, no inconsistency is detected.

Third, the system is capable of meta-inference or "introspection" very simply by searching its list of STM elements. E.g., it can determine whether a given formula and its negation are both currently in STM. This activity occurs via inference steps no different in principle from any of its other inferences. In effect, the system may look at snapshots of itself as it runs, rather than extrapolating to some "final" state.

2. Details of the Model

A. *The Modules*

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. The discarded facts flow directly into ITM. ITM's structure is similar to a stack, in that the most recently entered facts are the most easily accessed, although not actually removed.

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 focus, i.e. STM, when appropriate. A typical LTM entry looks like:

$$<T1,..., Tn, B>,$$

where the Ti and B are represented by logical formulae.

B. *Inference*

An inference cycle can be thought of as the process of updating the system's current focus of attention (STM). Given some state of STM, four different mechanisms work simultaneously to produce a new state of STM. These four mechanisms are direct observation, modus ponens (MP), semantic retrieval (from LTM), and episodic retrieval (from ITM). To model this simultaneity, our implementation uses a temporary waiting queue (QTM) which holds the next cycle's STM facts until all four mechanisms have finished working on the old STM facts. Once they have finished, elements of QTM are placed into STM one at a time, disallowing repetition of facts in STM. Throughout this process, "older" items in STM are moved into ITM as needed to maintain STM's size. Note that if QTM is very large then other means will be required to reasonably select elements to go into STM. In part this is addressed by RTM (see section 3).

Currently direct observation is provided by allowing outsiders to simply assert a fact to the system. This allows us the pretense of an autonomous system noting events in a dynamic environment.

MP is applied in the following form: from Ac and (Ax → Bx), Bc is inferred. That is, Bc is brought into QTM if Ac and (Ax → Bx) are already contained in STM. Consequently, at the end of such a cycle, Bc will be in STM (unless QTM has too many elements to fit into STM, a problem that has not arisen in current domains).

Facts from LTM are brought into STM by association. That is, when facts in STM unify with the first n elements of an (n+1)-tuple, <T1,...,Tn,B>, in LTM, then B will be brought into QTM (and subsequently into STM), with its variables properly bound.

Information retrieved from ITM into STM can take several forms. For example, since ITM is a chronological listing of all past STM facts, its structure allows for the retrieval of goal statements that are not yet satisfied, but that have already been pushed out of STM. This allows the system to work through a goal-subgoal process. We will not amplify on this aspect, since greater emphasis in this paper is on default reasoning. However, it is worth noting that the architecture allows representation of goal-driven behavior.

3. **Defaults and Relevance**

A. *Preliminary discussion*

Because we often deal with 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 typically birds can fly). 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, and, it seems easy to grant, must be a part of any kind of high intelligence. We therefore join the considerable ongoing research effort to characterize and utilize such reasoning in mechanical systems. (For a look at the general field of work in this area, see the Proceedings of the Workshop on Non-monotonic Reasoning, held in New Paltz, New York, October, 1984, sponsored by the AAAI, as well as [3,4,8,9].)

Our own efforts reported in this paper take a somewhat different approach than most. In particular, we are concerned with implementational issues concerning default reasoning. An underlying premise of default reasoning is that a real-world reasoner is limited, at least in terms of the scope and accuracy of the information to which it has access. As such, research in default reasoning can be viewed as backing off a bit from the more traditional topic of idealized reasoning agents that are infallible and omniscient. What we propose is that going one step further, namely, studying agents that also have limited computing resources (much as people do), makes some of the difficulties of formal representation of default reasoning more tractable. That is, greater limitations serve to constrain solutions to the point that answers may be more easily seen.

In particular, a centerpiece (and bugbear) of formal research in default reasoning has been that of (global and derivational) consistency tests. Even when, as in the case of circumscription (see McCarthy [3]), direct testing of consistency is avoided by clever syntactic manipulations, there is still implicit reference to global properties of the reasoning system (i.e., its set of axioms). Time is then taken to assess logical consequences of these properties before a default conclusion is drawn. Thus there is still a strong flavor of idealized reasoning here.

Suppose, for instance, we would like to have a reasoning system use a rule such as A $\rightarrow$ B to conclude B, given A, and later, in the face of new evidence, be able to retract its belief in B. A somewhat standard (idealized) 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, and is the source of the aforementioned consistency tests. The point of the "it is consistent to conclude B" is to see whether there *already* is evidence to retract B, i.e., to prevent the conclusion B in the first place. Our idea is that instead of holding up the system's conclusion until such a test can be made, we can let it "jump" directly to the conclusion B, and *then* decide whether it was rash.

Thus the question is whether or not these default rules should be encoded as such. In commonsense reasoning, nearly all rules, it seems, are actually defaults, since we can rarely 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. Regarding our own system, it would be natural to represent such a rule as an item to be retrieved from LTM when appropriate. One would then proceed as normal in the inference process until a direct contradiction is found in STM. At this point something would have to be done to resolve the inconsistency. In particular this allows the *use* of a rule such as A $\rightarrow$ B, even though it may be recognized as not strictly true. (Though this is not a trivial point, we will leave further discussion for a future paper. See [2] for a related idea.)

We then conceive of a real-world resource-limited entity as acting (somewhat slowly) over time while using defaults to allow itself a short-cut to quick (although fallible) answers. Our contention is that precisely such an entity is one to have a particular need for default reasoning, and to have means for performing such effectively. Elsewhere [1] we analyze in greater detail the underlying philosophy and logic of this approach.

B. *A default example*

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

ITM: < ... empty ... >

STM: bird(Tweety)

LTM: <bird(x), bird(x) → flies(x) >
<ostrich(x), ostrich(x) → ¬flies(x) >

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

STM: bird(Tweety)
* bird(x) → flies(x) (Here the star (*) indicates an item newly placed in STM.)

An application of MP would then leave:

STM: bird(Tweety)
bird(x) → flies(x)
* flies(Tweety)

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

STM: bird(Tweety)
bird(x) → flies(x)
flies(Tweety)
* ostrich(Tweety)

This new fact would then trigger the rule from LTM that ostriches do not fly (and have the side-effect of pushing "bird(Tweety)" into ITM).

ITM: bird(Tweety)

STM: bird(x) → flies(x)
flies(Tweety)
ostrich(Tweety)
* ostrich(x) → ¬flies(x)

Again MP may be applied, resulting in:

ITM: bird(Tweety)
bird(x) → flies(x)

STM: flies(Tweety)

ostrich(Tweety)
ostrich(x) → ¬flies(x)
* ¬flies(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, through the use of a rule of typicality. 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 not in fact fly.

As indicated earlier, our approach is, first, to let an inconsistency arise. Then once both x and ¬x are together in STM, we want to be able to decide which (if either) of the two should be kept as a belief. Since STM is small, we will always be able to determine quickly and easily whether such a direct contradiction exists.

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 condition C, A should be preferred 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. One way of handling this is to maintain this information in a new store called Relevant Term Memory (RTM). While in general the resolution of conflicting beliefs surely depends on a vast amount of domain specific knowledge, so that no single formal trick will settle all such conflicts in an intuitively acceptable way, nonetheless RTM does

seem to offer some useful features in this regard. We turn to this now.

C. *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 irrelevant (old) information.

Since RTM is meant to contain information that is somehow "relevant", it is reasonable to employ a decay mechanism. This provides any formula coming into STM with a maximum decay time (measured in inference cycles) during which a copy of it remains in RTM. The continued presence or reappearance of the formula in STM at any time in this period will maintain or reset its decay time at the maximum value. In this way, beliefs which are frequently in focus will tend to remain relevant. 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 remain in STM.

However, this will not be enough, for we also want to be able to maintain use of the "wins" predicate previously mentioned, as long as such use is relevant. Any time the "wins" predicate is used to decide on belief A over $\neg$A, the fact that A "wins" should be stored in RTM. It seems reasonable then to postulate a model of RTM incorporating both the decay mechanism and special treatment of the "wins" predicate. Experiments clearly are called for in order to test these ideas about the function of such a Relevant Term Memory. We anticipate that much work will have to be done to make RTM function flexibly. In what follows we illustrate more precisely what we have done in the way of initial experimentation

in this direction.

D. *Experimental Illustration*

We will present an example of the use of RTM in our system in one domain, namely, that of the flying ability (or lack thereof) of various types 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 several inference cycles, STM may now contain the conclusion that Tweety can fly. Later we observe that Tweety is a penguin, i.e., STM will contain "(penguin(Tweety), OBSERVATION)". After some reasoning it is discovered that Tweety cannot fly (because of "justification2"). Using some reasonable rules for conflict resolution (weighting "justification1" against "justification2", we now have a mechanism in place that puts a "wins" predicate in front of "(¬flies(Tweety), justification2)", and places this in RTM. As RTM contains relevant information, we use RTM to keep "false" information from finding its way back into STM. For example, in the

above case, "(flies(Tweety), justification3)" will not reenter STM unless justification3 is sufficient to remove "wins(¬flies(Tweety), justification2)" from RTM (e.g., Tweety is found to be a rare type of penguin that does indeed fly).

We present 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 deductions 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, unless the fact is obtained through observation (indicated by "[OBS]"); it should be clear in the example how all facts have found their way into STM. RTM elements are shown with their decay time (i.e. time to be left in RTM, measured in inference cycles). For illustration we have chosen a maximum decay time of 20. The initial configuration is as follows:

LTM	**STM**
< bird(x), bird(x) → flies(x) >	bird(Tweety) [OBS]
< penguin(x), penguin(x) → bird(x) >	
< penguin(x), penguin(x) → ¬flies(x) >	
< ostrich(x), ostrich(x) → ¬flies(x) >	

The following extended list of consecutive states of STM illustrates the system's performance. As before, in each stage we mark the new items with a star (*); they will appear at the bottom within each stage. For purposes of illustration, we have set the STM limit to six elements. Recall that old facts are removed from STM (and added to ITM) to allow new ones to enter into STM. Note that RTM contains relevant information about what has occurred.

STM State		STM Contents	RTM Contents
(1)	*	bird(Tweety) [OBS]	20 bird(Tweety)
(2)		bird(Tweety)	20 bird(Tweety)
	*	bird(x) → flies(x)	20 bird(x) → flies(x)
(3)		bird(Tweety)	20 bird(Tweety)
		bird(x) → flies(x)	20 bird(x) → flies(x)
	*	flies(Tweety)	20 flies(Tweety)
(4)		bird(Tweety)	20 bird(Tweety)
		bird(x) → flies(x)	20 bird(x) → flies(x)
		flies(Tweety)	20 flies(Tweety)
	*	penguin(Tweety) [OBS]	20 penguin(Tweety)
(5)		bird(Tweety)	20 bird(Tweety)
		bird(x) → flies(x)	20 bird(x) → flies(x)
		flies(Tweety)	20 flies(Tweety)
		penguin(Tweety)	20 penguin(Tweety)
	*	penguin(x) → ¬flies(x)	20 penguin(x) → ¬flies(x)
	*	penguin(x) → bird(x)	20 penguin(x) → bird(x)
(6)		flies(Tweety)	20 flies(Tweety)
		penguin(Tweety)	20 penguin(Tweety)
		penguin(x) → ¬flies(x)	20 penguin(x) → ¬flies(x)
		penguin(x) → bird(x)	20 penguin(x) → bird(x)
	*	¬flies(Tweety)	20 ¬flies(Tweety)
	*	bird(Tweety)	20 bird(Tweety)
			19 bird(x) → flies(x)
(7)		penguin(x) → bird(x)	20 penguin(x) → bird(x)
		¬flies(Tweety)	20 ¬flies(Tweety)
		bird(Tweety)	20 bird(Tweety)
	*	wins(¬flies(Tweety))	20 wins(¬flies(Tweety))
	*	loses(flies(Tweety))	20 loses(flies(Tweety))
	*	bird(x) → flies(x)	20 bird(x) → flies(x)
			19 flies(Tweety)
			19 penguin(Tweety)
			19 penguin(x) → ¬flies(x)
			18 bird(x) → flies(x)
(8)		¬flies(Tweety)	20 ¬flies(Tweety)
		bird(Tweety)	20 bird(Tweety)
		wins(¬flies(Tweety))	20 wins(¬flies(Tweety))
		loses(flies(Tweety))	20 loses(flies(Tweety))
		bird(x) → flies(x)	20 bird(x) → flies(x)
	*	bird(Oscar) [OBS]	20 bird(Oscar)
			19 penguin(x) → bird(x)

			18	flies(Tweety)
			18	penguin(Tweety)
			18	penguin(x) → ¬flies(x)
			17	bird(x) → flies(x)
(9)		wins(¬flies(Tweety))	20	wins(¬flies(Tweety))
		loses(flies(Tweety))	20	loses(flies(Tweety))
		bird(Tweety)	20	bird(Tweety)
		bird(x) → flies(x)	20	bird(x) → flies(x)
		bird(Oscar)	20	bird(Oscar)
	*	flies(Oscar)	20	flies(Oscar)
			19	¬flies(Tweety)
			19	bird(Tweety)
			18	penguin(x) → bird(x)
			17	flies(Tweety)
			17	penguin(Tweety)
			17	penguin(x) → ¬flies(x)
			16	bird(x) → flies(x)
(10)				

Note, in step 6, that several facts have been pushed out of STM (and into ITM). Further note that in this step "bird(Tweety)" is immediately brought back into STM through an application of MP (so it now appears at the bottom of STM, instead of the top).

The "loses(flies(Tweety))" in step 7 will cause further inferences leading to "flies(Tweety)" to be suppressed. Note that both "bird(Tweety)" and "bird(x) → flies(x)" are in STM in step 7, but that "flies(Tweety)" has not appeared in STM in step 8, since RTM is now suppressing this fact.

Nothing new is brought into STM after step 9. There are no new facts in LTM that can be associated with current STM information, and none of the conceivable inferences lead to anything new, except for "flies(Tweety)", which is suppressed.

It is important to note how RTM is functioning. All facts in STM enter RTM with a decay time of 20. (If a fact is already in RTM, its decay time is reset to 20.) With each iteration, the decay times in RTM are decremented. Note that facts that have been pushed out of STM may still remain in RTM, but will have lower decay times. Thus after perhaps 20 more inference cycles, in which additional observations may have resulted in new elements coming into STM, "wins(¬Flies(Tweety))" will likely have decayed to 0 in RTM. At such a time,

"flies(Tweety)" will no longer be suppressed from STM if MP set it up in QTM should bird(Tweety) happen to reappear in STM.

4. **Conclusions and Future Work**

We have presented a detailed model of real-time reasoning intended for a commonsense domain. Particular emphasis has been placed on default reasoning. Features of our approach include the ability to incorporate inconsistent information into the database, and the ability to draw conclusions and revise them later if necessary, without first checking their soundness with respect to the entire database.

We are currently at a stage in the development and implementation of this work where choice of a suitable domain for long-term research is advisable. Such a domain should have a number of characteristics if it is to challenge our techniques and yet be within grasp. Specifically, it should be both complex enough to test our mechanisms fairly severely, and yet simple enough to allow reasonable experiments to be done. In effect, we need to go one step up from the blocks world.

The domain we have chosen for our next stage of work we call the Desert Island, or just "The Island". We envision a robot "marooned" on a desert island with a small quantity of fauna, flora, etc.: birds (of course! -- who could do defaults without them?), fish, mangos, trees, sand, water, clouds, rain, sun. The robot is to have both short-range and long-range goals. The former type include eating when hungry, escaping danger, etc. The latter include learning about the island, such as learning that there are interesting classes of things (e.g. birds).

BIBLIOGRAPHY

(1) Drapkin, J., Miller, M., and Perlis, D. [1985] Consistency before and after, working paper.

(2) Glymour, C. and Thomason, R. [1984] Default reasoning and the logic of theory perturbation. Workshop on Non-monotonic Reasoning, New Paltz, NY, Oct. 17-19.

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

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

(5) Miller, G. [1956] The magical number seven plus or minus two, Psych. Rev. 63.

(6) Nilsson, N. [1983] Artificial intelligence prepares for 2001. AI Magazine, 4, 4.

(7) Perlis, D. [1984] Non-monotonicity and real-time reasoning, Workshop on Non-monotonic Reasoning, New Paltz, NY, Oct. 17-19.

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

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