LIFE ON A DESERT ISLAND:
ONGOING WORK ON REAL-TIME REASONING

Jennifer Elgot – Drapkin[a,b] Michael Miller[a,c] Donald Perlis[a,d]

University of Maryland
College Park, Maryland 20742
(301) 454-2002

ABSTRACT

We discuss ongoing work in real-time reasoning with applications to default reasoning. We explore ways in which this may also be relevant to the frame problem and associated issues.

---

[a] Computer Science Department, University of Maryland.
[b] IBM Research Division, Yorktown Heights, NY.
[c] Systems Research Center, University of Maryland.
[d] Institute for Advanced Computer Studies, University of Maryland.

1. INTRODUCTION

Most of the A.I. 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. 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 [10], which in particular seems appropriate to an autonomous exploratory robot.

In the current paper[1], we imagine a robot on a desert island left to its own devices. It has been endowed with a data base of information that it can use in order to get along in this world. The robot must be able to use (incomplete) information in real-time. 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. We claim that virtually all A.I. ventures of this sort will sooner or later have to address the issues of *default reasoning*, *non-monotonicity*, and the *frame problem*.

The remainder of this paper is outlined as follows: In section 2, we present the background material which we take as a point of departure in our current work. Section 3 presents a description of the reasoning mechanism with which we are working. Section 4 explains the relationship of non-monotonicity to our mechanism. Finally, section 5 sketches some thoughts on applications to the frame problem.

2. BACKGROUND

2.1 NON-MONOTONICITY, DEFAULTS, AND THE FRAME PROBLEM

It is clear that an autonomous robot will have to deal with a world about which it will have only partial knowledge. Conclusions will frequently be drawn without full justification. As a consequence, some facts will have to be retracted in the face of further information. We therefore join the considerable ongoing research effort to characterize and utilize non-monotonic or default 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. Also see [6,8,12,13].)

It has been observed that non-monotonic reasoning may be an appropriate vehicle for treating the frame problem (see [5,7].) Roughly, it seems that default reasoning applies to situations in which one desires to know which assertions are true and yet insufficient data is available. The frame problem might be seen as a special case of this in which the assertions in question are of the form, ''If action s occurs in a situation in which proposition P holds, will P still hold after s is completed?'' An appropriate rule might be called a frame-default rule. An example is the rule that things that aren't known to change stay the same. However, the central role that time plays in the frame problem makes it a very special case of default reasoning.

It would not be a surprise, then, that a real-time reasoner that has features relevant to default reasoning would also be of potential application to frame-default reasoning, especially because the real-time aspect suggests a special architecture. In what follows, we discuss a system on which we are currently working that we hope will prove robost enough to allow for some fairly sophisticated real-time default reasoning.

---

[1] An outgrowth of work reported in [11].

2.2 APPROACHES TO DEFAULT REASONING

Our own efforts reported in this paper take a somewhat different approach than most. In particular, we are concerned with certain implementational issues concerning commonsense reasoning. An underlying premise 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, our research 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 commonsense 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 commonsense reasoning has been that of (global and derivational) consistency tests. Even when, as in the case of circumscription (see [6,7]), 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 commonsense conclusion is drawn. Thus a strong flavor of idealized reasoning has persisted.

Suppose 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, it seems, nearly all rules 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. One would then proceed as normal in the inference process until a direct contradiction is somehow brought to the reasoner's attention (a process which will be explained shortly). At this point something would have to be done to resolve the inconsistency. It is important to note that a contradiction should not be something that will incapacitate our system. Rather, it is something that is to be expected, recognized (but, only when relevant to the current focus of attention), and resolved by the system as part of its normal operation. 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 [3] 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 [2] we analyze in greater detail the underlying philosophy and logic of this approach.

3. DETAILS OF OUR MODEL

3.1 ARCHITECTURE

Our reasoning model contains five key elements: STM, LTM, ITM, QTM, and RTM. STM, LTM, and ITM are standard parts of cognitively-based models of memory. STM is meant to represent one's current focus of attention. It is a small set of beliefs that are currently ''active''. These beliefs are represented as logical formulae and are used to help establish STM's next state, a process which will be

described shortly. STM is structured as a FIFO queue. Since STM's size is limited[2] , as new facts are brought into STM, old facts must be discarded. That is, the older, discarded facts are no longer in focus.

It is convenient to conceive of STM as a theorem prover to which LTM supplies axioms. LTM may then be thought of as a data base of information to which the robot has access. LTM is implemented as a series of tuples of the form:

$$<T_1,...,T_n,\mathrm{B}>$$

where the T_i and B are represented by logical formulae. The idea behind LTM is that beliefs are held as a series of associations. Thinking about a subject triggers (possibly many) past associations. These associations are then brought into focus (i.e. STM) when appropriate.

ITM is of unbounded size and holds all information which ''spills over'' from STM in chronological order, i.e., in order of entry. Thus, ITM is implemented similarly to a stack, in that the most recently entered facts are the most easily accessed, although they are never removed.

QTM is a technical device that controls the flow of information into STM.

RTM is the repository of default resolution and relevance. RTM is implemented as a list of facts that have most recently been in STM. Facts are coded with a time decay variable so that facts can decay out of RTM as they are no longer relevant; i.e, not found in STM for a specified number of inference cycles.

This architecture is illustrated in Figure 1. The 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.

FIGURE 1. Architecture of the memory model.

[2]In our implementation, STM's size is fixed, yet easily changed for experimental purposes. An interesting sidelight is that an STM size of 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 which measure human short-term memory to hold seven plus-or-minus two ''chunks'' of data at one time [9].

3.2 GENERAL FEATURES

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. 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.

Fourth, the utility of RTM is to allow for such things as prohibiting faulty default conclusions. That is, since STM is so small, it is likely that information that would typically block a default conclusion from being drawn has recently left STM, but it still remains in RTM. Being in RTM is sufficient to prohibit a faulty default, as we consider RTM's entries as relevant enough to have a bearing on reasoning, yet not central enough to be the catalyst of further inference.

Finally, information stored in ITM and in RTM is at times accessible to STM. Thus, information from the past can be brought back into focus when appropriate. This allows the system to use such information in working through goal-subgoal behavior as well as using past information as a default when the frame problem arises.

3.3 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, other means will be required to reasonably select elements to go into STM. In part this is addressed by RTM.

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 $\rightarrow$ Bx), Bc is inferred. That is, Bc is brought into QTM if Ac and (Ax $\rightarrow$ 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, $<T_1,...,T_n,B>$, 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.

3.4 AN EXAMPLE

As an example of this mechanism in action, consider the following state of affairs, with the size of STM fi xed at a maximum size of four:

ITM: MX< ... empty ... >

STM: bird(Tweety)

LTM: <bird(x), bird(x) → flies(x) >
<flies(x), flies(x) → winged(x) >

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

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

The star (*) indicates an item newly placed in STM. An application of MP would then leave:

STM: bird(Tweety)
bird(x) → flies(x)
* .ls 1

Again, a new association will be triggered from ltm, resulting in:

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

This new fact would then trigger MP again (and have the side-effect of pushing ‘‘bird(Tweety)’’ into ITM).

ITM: bird(Tweety)

STM: bird(x) → flies(x)
flies(Tweety)
flies(x) → winged(x)
* .sp 2

4. REAL-TIME NON-MONOTONICITY

It does not take an especially large effort to produce a rudimentary non-monotonic-type reasoning system from the above architecture. We present an example which is similar to the above, but which incorporates non-monotonicity into the reasoning system.

Let us this time start the system in the following state, where the second entry in LTM is different from before:

ITM: MX< ... empty ... >

Life on a Desert Island...

STM: bird(Tweety)

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

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

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

An application of MP would then result in:

STM: bird(Tweety)
bird(x) → flies(x)
* .ls 1

Now suppose the system discovers (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)
* .ls 1

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)
* → ¬flies(x)

Again MP is applied, resulting in:

ITM:.br bird(x) → flies(x)

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

Note at this point that STM contains both the belief that Tweety does not fly, as well as the belief that 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 the system to be able to retract the belief that Tweety flies, and instead conclude that Tweety, in fact, does not 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.

Several methods of conflict resolution are available to us, each requiring nothing more than providing an extra term to facts in STM which indicates the justification for bringing that fact into focus. For example, something that is brought into STM as a result of direct observation is tagged with the term ''OBS'', while a fact deduced through modus ponens is tagged with ''MP'', etc. These tags then allow the system to favor an observed fact over a deduced fact, and a more recent observation over an earlier one.

This sort of self-adjusting can be thought of in terms of the following intuitive thinking: ''X ought to be true. Let me look and see.'' This is not to say that resolution of such contradictions is trivial; on the contrary, it is in general very hard. We are experimenting with a ''wins'' predicate that we hope will be useful in a fairly broad setting. The idea of ''wins'' is that given two conflicting conclusions A and B, an additional fact of the form Wins(A,B) will state that, say, A wins out over B, so that A is retained as the conclusion and B withdrawn. Of course this must be based on the context in which A and B were initially concluded, so that we are talking about a real-time kind of truth maintenance as in [1].

Notice also that all information about Tweety may soon leave STM, but will remain in RTM for some number of inference cycles (and thus still remain relevant). If at a later time (not too late, as decay out of RTM may eventually occur), Tweety is in focus again, RTM's record of Tweety's inability to fly will block the statement ''flies(Tweety)'' from reappearing in STM. Thus, the default rule is no longer applicable. That is, once a contradiction arises in STM, and we have resolved the contradiction in favor of one of the contradictory facts, we can simply remove the other fact from STM[3]. Furthermore, we can just as easily remove this fact from RTM so that it no longer bears any relevance to the reasoning from that point on.

5. THOUGHTS ON FRAME DEFAULTS

In developing and building our model of reasoning we have considered several types of reasoning that we would like to be able to capture. The most basic of these is the following: Imagine that our robot has recognized that a piece of edible fruit is on Tree A, and at some later time decides that it needs a piece of edible fruit (perhaps it is hungry!). We would like to have the robot remember that there was a piece of fruit on Tree A and that it is probably still there for the robot to eat. It seems simple enough to encode some sort of rule that allows the robot to deduce this fact. One might try a general rule such as that stated earlier: things that aren't known to change stay the same. This was largely captured by Hayes [4]; Lifschitz [5] recently has given dramatic force to it in the context of circumscription. These authors related what is known to change to what is known to *be caused* to change.

It would seem that we could employ much the same tack as before: simply turn default rules about causality into bald rules that can later be contradicted by other conclusions, and then use a wins predicate. Our real-time architecture was designed in large part to facilitate a wins predicate and other kinds self-adjusting over time.

In addition to seemingly being able to translate more standard default conclusions into real-time terms, we also have mechanisms that lend themselves to associated issues, such as granularity and context. Again, for illustration purposes, imagine a eagle flying over the island. Furthermore, suppose that the eagle is flying in a circular pattern (as they are known to do). The robot has already seen the eagle and wishes to answer the question, ''Where is the eagle now?'' One answer to this question is simply, ''The eagle is over the island.'' Another answer is the actual spatial coordinates of the eagle. Both are correct, though they differ in their degree of granularity.

---

[3]In our implementation, we actually retain the fact that has been determined to be incorrect, but we tag this fact in such a way so that its incorrect nature is evident. This is done so that ITM can maintain a complete chronological listing of STM facts. We feel that this will be important in the future as we may attempt to implement a learning device that scans ITM attempting to identify ''patterns of reasoning''.

A related issue is context, in the following sense: the goals of the reasoning system should partially determine the things reasoned about. Thus if it is known that an eagle is circling overhead and if the immediate goal is to find fruit, it may be wasteful to even consider the whereabouts of the eagle after 5 seconds, one minute, etc.

This matter of granularity is one that has cropped up because of the type of system that we are working on: a real-time reasoner. We have provided mechanisms for histories of events (ITM), for current events (STM), for a pool of relevant information (RTM), and general retrieval information (LTM) to provide appropriate focus for STM.

References

(1) Doyle, J. [1979] A truth maintenance system, *Artificial Intelligence* **12**, (3), 231-272.

(2) Drapkin, J., Miller, M., and Perlis, D. [1986] Consistency before and after. Working paper.

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

(4) Hayes, P. [1971] A logic of actions. *Machine Intelligence*, Meltzer, B., and Michie, D. (eds.), Edinburgh University Press. intelligence

(5) Lifschitz, V. [1986] Formal theories of actions. Manuscript.

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

(7) McCarthy, J. [1986] Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, **28** (1), 89-116.

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

(9) Miller, G. [1956] The magical number seven plus or minus two. *Psychology Review* **63**.

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

(11) Perlis, D. [1984] Non-monotonicity and real-time reasoning. *Proceedings of the Workshop on Non-monotonic Reasoning*, New Paltz, NY.

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

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