Non-monotonicity and Real-time Reasoning

D. Perlis University of Maryland College Park

I. Introduction

All forms of non-monotonic reasoning so far proposed suffer from a common failing: they are not computationally highly effective, indeed for the most part cannot be carried out even in principle. They involve either self-consistency tests, or general theorem-proving techniques. Here we argue that, in fact, all non-monotonic logics proposed involve a kind of self-referential reasoning, and that this leads to an inherent resource-limitation. This, in turn, leads to a different view of non-monotonicity than has been given before. Finally, we suggest that this can be turned into a plus for reasoning systems in specific implementational details.

Indeed, the apparent motivating idea of default reasoning, and more generally commonsense reasoning, is to give up the hopelessly impractical approach of assuming complete world knowlege and complete deductive powers, in favor of a more real-time orientation. We are then suggesting that such real-time issues be inspected more closely.

II. Self reference

Non-monotonic reasoning (e.g., McDermott & Doyle (1980), McCarthy (1980), Reiter (1980)) involves an unusual kind of inference in which the inference mechanisms themselves are fodder for still another inference mechanism. Specifically, if a system cannot derive not-P by certain rules, then it can derive P from that very fact! This is often regarded as a default process, in that we are assuming P by default if nothing contradicts it, i.e., if not-P is not known. (See Stalnaker (1980) and Moore (1983) for a novel view of this situation.) The more general form of such an inference can be represented as

[B & -Known(-P)] --> P,

i.e., under condition B, if P is not known to be false, assume it is true.

In effect this recognizes the truism that a system must encode parts of itself to state what is not itself. A consequence of this is the objection raised by Kowalski (1979), Israel (1980), and others, that in fact the non-monotonicity hides an underlying monotonic process. If system S believes F(T) as a consequence of believing both B(T) and the default rule

$$[B(x) \& -Known(-F(x))] \dashrightarrow F(x),$$

then in fact it has, by hook or by crook, deduced and utilized -Known(-F(T)) in the process, even if this deduction is hidden away as an inference rule. Now, once this deduction is made, -Known(-F(T)) becomes a belief and will not disappear except by explicitly being removed. This is often overlooked in stating inference-rule versions of default reasoning. In fact, the process of inferring that S does not know something is highly non-trivial, as is that of removing such a belief afterwards. [Indeed, if S is to infer that it does not know something, presumably either S has axioms allowing it to deduce this (a situation hard to imagine except by explicitly including all not-knowings of S as axioms of S as well--not a pleasant computational prospect) or S somehow examines itself to determine that an assertion is or is not among its beliefs, much as searching through a book, or even consulting ones wristwatch. More on this later.]

The reason for removing a belief about what S does not know is evident when new information becomes available, e.g., that after all -F(T). For then we no longer expect S to believe F(T)! But as long as -Known(-F(T)) remains in S´s database, F(T) will remain as a consequence of S´s axioms and rules. So S should lose the information -Known(-F(T)) when -F(T) becomes known. Indeed, Known(-F(T)) should become derivable, and then this should take precedence over the old -Known(-F(T)). This is exactly the part that usually is missing from non-monotonic accounts of default reasoning. An explicit self-reference must be performed, in which S´s database is reviewed by S so that Known or -Known can be inferred for some predicate. It is this that makes such systems non-effective, since in general determining whether a formula X is derivable from a set of axioms is not decidable.

In any case, the non-monotonic character of inference is dispelled, for now the crux of the matter has been transferred to two things: determining what is or is not known, and then doing some error-correcting to resolve a conflict. Since both of these are of clear importance to an intelligent system in any case, then default reasoning can be seen as a combination of these rather than a new form of reasoning in its own right. For this reason we speak of self-referential reasoning here as a more basic form than non-monotonic or default reasoning.

III. Real-time concerns

Now we take another point of view. If smart systems do in fact perform default reasoning, and if they do do so by effective means (as they must), then what is decided is not derivability in the logical sense, but something different. (Konolige (1984) has explored certain modes for representing deductions agents may perform, without investing them with full logical consequence as their means of inference.) Here we wish to suggest a specifi c and yet quite general kind of self-referential inference that can be applied to the above ideas about defaults. In particular we consider that a system may have to reason about its own process of reasoning, such as that it has begun to reason about something but has not yet fi nished.

Consider our system S as before, in which however the deductions S performs are seen as occurring over time. Here we do not want to assume that at some time S stops inferring new things. We picture S as a ´ computer individual´ --to borrow a phrase from Nilsson (1983)--that goes on thinking as it interacts with the world. Thus there is no time at which it has ´ got´ all its conclusions. However, at any time, there are conclusions it already has, and ones it hasn´ t. We endow S with a mechanism Introspect(X), that allows it to infer whether it already knows X. This can amount to a mere lookup in the database of what S already knows. The result of performing Introspect(X) is that either Know(X) or -Know(X) will appear in S´ s database, depending on whether in fact X was or was not already in S´ s database. It matters not that S may know A and A-->X; as long as X itself is not present as a separate item in the database, -Know(x) is returned.

This seemingly rather obtuse procedure in fact is anything but. For recall that S is continuously thinking, deducing. If at any moment before the Introspection is done, X is inferred, then Know(X) will appear instead of -Know(X). And if it is of suffi cient interest to S to apply Introspect at all, then it is most likely that S already will have been trying to decide X, and would have proven X from such simple axioms as A and A-->X before a complex default procedure could have fi nished.

Specifi cally, imagine S wanting to know whether Tweety can fly (i.e., whether Flies(Tw)). S considers both proving Flies(Tw) and denying Flies(Tw) (proving -Flies(Tw)). If S knows Ostrich(Tw) and Bird(Tw) and that

Ostrich(x) -->. -Flies(x)

and also knows a default rule such as

Bird(x) & -Known(-Flies(x)) .--> Flies(x)

then S can more quickly (in terms of steps in a proof) prove -Flies(Tw) than invoke the default. Even if Introspect is invoked before -Flies(Tw) is proven, so that the antecedent -Known(-Flies(Tw)) is determined fi rst, the default Flies(Tw) still will not yet have been accomplished, for another step (modus ponens) is needed.

If by some chance the default is invoked fi rst to prove Flies(Tw) only later to be contradicted by the direct proof of -Flies(Tw) -- for instance if Ostrich(Tw) is learned or deduced (possibly by a background process) only later -- we still have a conflict resolution to fall back on, namely, that not only are Flies(Tw) and -Flies(Tw) in conflict, but also are Known(-Flies(Tw)) and -Known(-Flies(Tw)). Now the latter is easy to resolve. If Known(-Flies(Tw)) has been deduced from -Flies(Tw) (by Introspect), then it takes precedence over -Known(-Flies(Tw)), since the latter was derived not from Flies(Tw) but from the absence of its negation, and this no longer obtains. Once this is done and -Known(-Flies(Tw)) is removed, the basis for -Flies(Tw) is gone and it too can be removed a la Doyle (1979). So in any case, we end up with Flies(Tw) and Known(Flies(Tw)) as desired. (It is not claimed that ALL types of conflicts are so easily resolved. However, when the conflict is between Known(X) and -Known(X) and if X is in fact known, then Introspect settles the matter nicely.)

We see then that two scenarios are possible in this example. -Flies(Tw) can be derived outright at the start, or S can temporarily deduce Flies(Tw) only later to revise itself in favor of -Flies(Tw).

The lesson we derive from this is as follows. The database changes in many ways. Yet S needs to keep reasoning AS it changes, and WITH its axioms that are changing: about it, with it, as it changes. On replacing consistency (i.e., derivability) with mere (temporal) membership in the know
consistency is no longer necessary to preserve at all times. Local consistency, i.e., in the working set of facts, can be restored temporari

The type of system we are describing can have many different designs; it could even run a "background" complete theorem prover that would, in time, produce any given logical consequence of its knowledge. However, the thrust of our examples is that such is not practical, and commonsense reasoning is perhaps better seen as a hit or miss phenomenon, in which mistakes are made, things overlooked, but with an overall good batting average as time proceeds. Such a system is not a problem-solver in the usual sense; it inhabits and deals with a world that may provide it with many "problems" but it itself must manage to keep going despite its possible failures. (We again call attention to the "computer individual" of Nilsson (1983).)

IV. A system

We have implemented a system (Perlis 1981) that operates along the lines specifi ed above, with an introspection device for self-knowledge, and are currently upgrading it for a variety of default situations. In what follows we describe some very preliminary results from the older system.

In order to keep the
introspections and other inferences rapid, only a small ´ working subset´ of the entire database is searched when an introspection is performed: the set of ´ currently used´ beliefs. These are kept in a queue that is updated on a least-recently-used basis, in analogy with human short-term memory. The entire architecture is highly reminiscent of the production systems of Newell and Simon (1972), and indeed early experiments with our system show that a short-term memory size that works best for trial problems to date is approximately that of human short-term memory as well.

(There are well-known measures of human short term memory (STM) as having 7 plus or minus 2 ´ chunks´ of information (Miller 1956).)

The model hypothesizes a three-part memory: long-term memory (LTM) for the bulk of stored beliefs, intermediate-term memory (ITM) for a record of recently accessed beliefs, and short-term memory (STM) for beliefs currently being accessed. (It is a variation on the production system format.) Inference itself takes place in STM, which is quite small. ITM serves both to keep an implicit stack of goals and to allow some readily accessible feedback about what has been accomplished and in what order; ITM then is larger than STM. Finally, LTM is very large and includes associations (productions) that largely determine what beliefs will be brought into STM. The main inferential mechanism we consider is modus ponens (MP).

The rules for use of LTM and ITM are as follows. First, we fi x the size of STM. The rules then are simple: each set of STM elements trigger all possible MP inferences and LTM associations (usually there aren´ t many) thereby pushing these into STM and ´ popping´ older elements further into ITM.

Experiment has shown Miller's 7 "chunks" (formulas, although this is an ill-defi ned measure) to be a roughly optimum STM size, at least for the present application. E.g., while 8 worked fi ne (in 45 stages of STM inference) the same program except with STM of 4 elements failed to get very far at all before starting to loop. On the other hand, increasing STM to 16 actually slowed the system down, although it eventually achieved the same outcomes as with 8.

Default reasoning can proceed as follows in such a system. We refer back to the problem of Tweety. Suppose that someone mentions Tweety, thereby causing the recollection in our hypothetical system S that Tweety is a bird, which in turn causes the recollection that birds can fly unless known otherwise. That is, Bird(Tw) and

$$\text{Bird}(x) \ \& \ \text{-Known}(\text{-Flies}(x)) \ . \text{-->} \ \text{Flies}(x)$$

come into STM. Then the conclusion Flies(Tw) is inferred in STM. [Here -Known(-Flies(Tw)) results from an INtrospection of STM. This is the afore-mentioned analogue of looking at a wristwatch to see the time. There is no strangeness to the logic; S simply knows it can get certain information by performing a particular operation.]

In the meantime, it is mentioned that Tweety is in Africa, and this triggers S's memory that Tweety is an ostrich, which in turn brings Ostrich(x) --> -Flies(x) into STM. Thus our system S can deduce in STM that -Flies(Tw), even though it has just deduced Flies(Tw) a moment earlier! The punchline is that this is not now catastrophic, for S can use this event (a direct contradiction in STM) as a cue to temporarily mark both items as uncertain and bring in any available information to help resolve the conflict. Thus a kind of local (to STM) consistency is restored. For instance, since -Flies(Tw) was not derived with use of a default axiom, whereas Flies(Tw) was, the former may carry more weight. This itself can take the form of a rule (default or otherwise) in LTM that is triggered whenever a conflict occurs; of course this necessitates a mechanism for recalling how recent inferences were made, and this function is served by ITM.

This however is more awkward than need be. For it is not necessary to keep trailing along behind birds and flying all the qualifications about what is not known. Instead we can allow the bald assertion that birds fly, unqualified, and use it as such, backing off when counterexamples occur. Thus S hears about the bird Tweety, infers it can fly, then learns it is an ostrich, infers that Tweety cannot fly, sees the conflict, and temporarily reserves judgement (this is done by putting quotes around the two offending assertions). Now instead of the rule mentioned above, S can use an axiom that "some birds cannot fly and the axiom to the contrary (birds can fly) is only approximately true". That is, the known-to-be-false axiom that birds can fly, is kept in LTM and used regularly, to be replaced by the truer version (some birds cannot fly) only when needed (i.e., when a conflict results). [Just when one axiom is retrieved from LTM into STM rather than another, is however not a simple matter in general, and we do not claim to have this worked out yet in a large number of cases.]

The same suggestion seems to apply to interacting defaults (Reiter & Criscuolo 1981). In their example, the defaults "Typically high school dropouts are adults" and "Typically adults are employed" could, under one treatment of default reasoning, yield "Typically high school dropouts are employed", whereas they argue for another treatment avoiding this undesirable conclusion. However, our viewpoint here is that there must be some underlying world knowledge that allows US to decide that the first conclusion is undesirable, and that THIS is what should be present in the reasoning system. Then it is sufficient that the system recognize when two or more defaults are interacting, for it can at that point refrain from trusting any such conclusions until examined more closely. For instance it may know that high school dropouts are more likely to have trouble getting jobs than high school graduates, but that jobs in general are plentiful at the moment, etc., so that the situation is too complex to judge with much confidence in the absence of explicit information about the matter.

Now such world knowledge is certainly not a simple matter to describe. However, it would appear that our approach offers some hope of extensibility, for unlike traditional belief logics, there is no intrinsic concern over encountering contradictory information. New default axioms can be added directly without checking them against all other for possible interactions. A contradiction is simply a cue for further analysis. Of course, this analysis may be very complicated, and we offer no general method for such at this time.

V. Conclusions

In working with a temporal and computational model of belief and inference (Perlis 1981), it was found that a practical realization of non-monotonic reasoning suggests itself on such a basis. Also a surprising feature was noted: the ´ size´ of the short-term memory that worked best was, counter to expectation, in excellent accord with data on human short-term memory in recall tests.

References

Doyle, J. A truth maintenance system, Artifi cial Intelligence, 12, 1979.

Israel, D. What´ s wrong with non-monotonic logic? AAAI-1, 1980.

Konolige, K. Belief and incompleteness. SRI TR319, 1984.

Kowalski, R. Logic for Problem Solving. North Holland, 1979.

McCarthy, J. Circumscription: a form of non-monotonic reasoning, Artifi cial Intelligence, 13, 1980.

McDermott, D. A temporal logic for reasoning about processes and plans. Cog. Sci. 6, 1982.

McDermott, J. and Doyle, J. Non-monotonic logic I, Artifi cial Intelligence, 13, 1980.

Miller, G. The magical number seven plus or minus two. Psych. Rev. 63, 1956.

Moore, R. Semantical considerations on non-monotonic logic. SRI TR284, 1983.

Newell, A. and Simon, H. Human Problem Solving. Prentice Hall, 1972.

Nilsson, N. Artifi cial intelligence prepares for 2001. AI Magazine, 4, 4, 1983.

Perlis, D. Language, computation, and reality. Ph.D. thesis. Univ. of Rochester, 1981.

Reiter, R. A logic for default reasoning, Artifi cial Intelligence, 13, 1980.

Reiter, R. and Criscuolo, G. On interacting defaults. Proc. 7th IJCAI, 1981.

Stalnaker, R. A note on non-monotonic modal logic. Cornell Univ. TR, 1980.