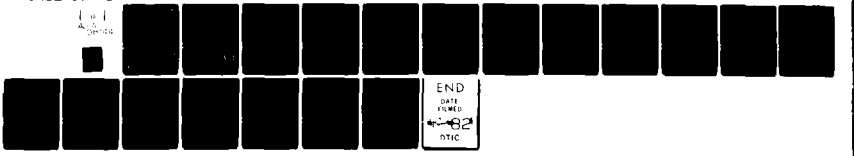
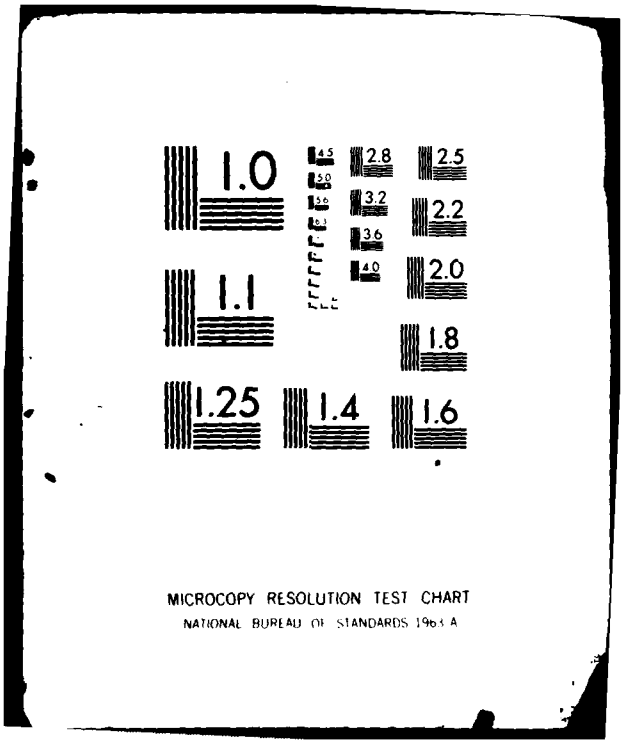
AD-A108 144 STANFORD UNIV CA DEPT OF COMPUTER SCIENCE F/G 12/2
THREE SHORT ESSAYS ON DECISIONS, REASONS, AND LOGICS,(U)
MAY 81 J DOYLE MDA903-80-C-0102
UNCLASSIFIED STAN-CS-81-864 NL



END
DATE FILMED
-82
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

May 1981

Report. No. STAN-CS-81-864

AD A108144

LEVEL II

Three Short Essays on Decisions, Reasons, and Logics

by

Jon Doyle

Research sponsored by

Advanced Research Projects Agency

Contract MDA 903-80-C-0102

Department of Computer Science

Stanford University
Stanford, CA 94305

DTIC
ELECTE
DEC 7 1981
S D
D

DTIC FILE COPY

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

81 11 16 026

Three Short Essays on Decisions, Reasons, and Logics

Jon Doyle

Accession For
NTIS GRA&I ☒
DTIC TAB ☐
Unannounced ☐
Justification
By Per Ltr. on file
Distribution/
Availability Codes
Dist | Avail and/or Special
A

Department of Computer Science
Stanford University
Stanford, California 94305
U.S.A.

May 1981

Abstract: This report collects together three short, survey-level essays introducing related approaches to decision-making, reasoning, and logic. The first essay, *Making Difficult Decisions*, was read to the Stanford Computer Forum on February 5, 1981. The following two essays, *Dependencies and Assumptions* (written with Johan de Kleer) and *Non-Deductive Reasoning and Non-Monotonic Logics*, are versions of articles to appear in the *Handbook of Artificial Intelligence* (A. Barr, P. Cohen and E. Feigenbaum, eds.).

DTIC
ELECTE
S DEC 7 1981 D
D

Acknowledgements: I thank Johan de Kleer for his permission to reprint our article here. Preparation of this report was supported by Stanford University under ARPA contract MDA903-80-C-0102.

Making Difficult Decisions

Jon Doyle

Abstract: Many techniques of formalized decision analysis and decision-making prove inadequate when applied to complex tasks. I discuss a new technique of formalized decision-making, called **reasoned deliberation,** *which captures much of the power and flexibility of informal human and organizational decision-making.*

The Nature of Decisions

All of us have to make decisions, and when the stakes are high, we must be particularly careful about how we make them. This is true for intelligent agents in general as well as for humans. In the next few minutes, I will try to sketch some of my recent studies into decision-making processes, studies taken both with an eye towards making more capable and responsible decision-making machines, and with an eye towards illuminating, recording, and possibly aiding the decision-making processes of human executives.

We face consequential decisions whenever we begin to attempt difficult, complex tasks—for example, providing for the future energy needs of civilization. To begin with, we must decide how to interpret our information about our current circumstances. For example, in energy policy, we must decide just how much recoverable oil we have, since different estimating techniques give different answers. We must also decide what our goals are. For example, do we want to build a larger civilization or restrict it to a certain size? We have to decide what are our means for accomplishing our goals. For example, can we rely on fusion, solar, or hydrothermal power a decade hence? And we also must decide just how efficacious these means are towards achieving our goals without creating or aggravating other problems.

It is characteristic of big problems that we have to make all these decisions at once. For example, politics is called the art of the possible, because political decision-making often requires giving up tentatively agreed-upon goals when no means for achieving them are immediately apparent. Because of this, political geniuses are those who save a prized goal by finding a hitherto overlooked compromise. Similarly, what we admit as means depends also on their efficacy and accuracy, and all too often, we color our judgement of our current circumstances by what we wish would be.

To make these several complex decisions concurrently, we must be able to treat the making of decisions as complex tasks themselves, requiring in turn further, although smaller, decisions and planning. We cannot simply decide on circumstances, decide on goals, decide on means, and then pick the winner—we simply do not have the information or computing competence to do so. Instead, we must keep breaking big problems down into smaller ones until we reach decisions which we feel confident

of handling correctly. We then combine these smaller decisions into bigger ones, but sometimes we must take back or revise one decision because of information discovered or conclusions drawn in making another decision. In addition, once the combined decisions grow large enough, we grow suspicious and reassess our decisions and the processes by which we arrived at them, to make sure we have not forgotten something, and to make sure we have not made an error of calculation somewhere.

Thus rather than viewing our decision-making process as an information-grinder (put the information in top, turn the crank, and out comes the decision), we see them more as a process of tentative, hesitant muddling through, of putting together lots of little decisions, and toying with them and their combinations until we can look back on the result and say "We'll stand by this one—it's the best we can do."

Reasoned Deliberation

How does this view of the decision-making process get translated into formal decision-making techniques?

Well, Bayesian decision analysis has been popular for some time, but has many difficulties of the information-grinder sort. One has to supply up front the entire formulation of the problem, assign all values and uncertainties, and then have to scrap both formulation and result if new information comes along. When applicable, Bayesian decision analysis has a lot going for it. It is just that all too often, one winds up butchering the problem to fit the technique. Of course, there are several sophistications of Bayesian decision analysis aimed at overcoming some of these problems, but by and large they still suffer from many of the original limitations.

I have grown interested in a class of decision-making procedures I call collectively *reasoned deliberation*. Reasoned deliberation is a decision-making technique based on keeping careful records of parts of the process of decision-making, so that they can be reviewed and perhaps revised later in a flexible fashion.

The key idea in reasoned deliberation is that one records the *reasons* for each conclusion drawn while deliberating. For example, if one concludes that coal burning is an option for energy production, one writes down the reason for this conclusion, perhaps "The National Academy of Sciences report says so." These recorded reasons allow later challenging and buttressing of the arguments developed during the course of the decision. For example, this reason for the coal option might be challenged by giving a reason showing the NAS report to be suspect, such as "The Nuclear Regulatory Commission report says the NAS used faulty data." This challenging reason may be challenged in turn, for example, with "The New York Times reported that the NRC report was influenced by industry pressure on the authors." Note that neither of these subidiary reasons directly addresses the question of whether coal is an option or not. They merely argue about what constitutes a sound reason for believing coal an option.

The major components of information in the reasoned deliberation process are these:

- the Purpose, the task at whose command the decision is being made;

- the Options, the possible outcomes of making the decision;
- the Reasons for and against the options, and supporting or challenging other reasons; and
- the Policies, the situational imperatives examined to determine reasons in the current decision.

The overall form of deliberation is to repeatedly retrieve and apply the policies relevant to the current state of the decision-making process. Policies are of the general form of condition-action rules. If a policy is applicable, that is, if current circumstances satisfy its condition, then its actions may be taken. Corresponding to the wide range of actions in human decision-making, policy actions admit a wide range, such as:

- Adding a new option (coal)
- Adding a new reason for or against an option (NAS for coal)
- Adding a new reason for or against a reason (NRC against NAS)
- Adding a new subdecision to be made or task to be carried out (Get a new opinion from the U.S. Geological Survey)

The decision-making process itself can control the retrieval and application of these policies in many ways, and can take actions such as:

- Applying another policy
- Retrieving policies relevant to a new reason or option (*I.e. reflecting on one's conclusions*)
- Carrying out a subtask or making a subdecision
- Postponing the decision to work on others
- Abandoning the decision (I.e. giving up its purpose)
- Deciding on a particular option as the outcome.

In summary, reasoned deliberation makes explicit the reasoning involved in decision-making so as to allow arguments to be constructed concurrently, incrementally, and to be reviewed and revised if need be—just those common features of complex decision-making that less flexible decision analysis schemes prohibit.

The Future

I have been exploring a larger theory of mind which integrates this sort of deliberative process with compatible theories of memory, actions, adaptiveness, and self-consciousness. Much theoretical work remains to be done. (See [Doyle 1980] for more details.)

In addition to the theoretical studies, I forsee immediate applications of reasoned deliberation in improving the decisions of current consultant programs, such as those under development in medicine, molecular biology, geological exploration, and computer-aided design. For example, MYCIN makes

decisions about infections and treatments using simple sorts of condition-action rules, but MYCIN rules can only vote for or against conclusions, and cannot argue about the rules previously applied. Thus many sorts of rules an expert might employ about special-case exceptions to general MYCIN rules cannot even be stated in MYCIN. MYCIN simply cannot take back a conclusion upon later realizing it is in special circumstances invalidating the former conclusion.

In closing, let me mention a speculative interest of mine, that of applying organizing principles discovered through the study of these sorts of decision-making processes to making human organizations more efficient and efficacious. Human organizational decision-making processes provide many lessons for my studies, and I hope that some day computational experiments with alternative organizational structures may return the favor.

Thank you very much.

January 18, 1981
Palo Alto, California

[1] Doyle, J., 1980. A model for deliberation, action, and introspection, Cambridge: MIT Artificial Intelligence Laboratory, TR-581.

Dependencies and Assumptions

Johan de Kleer and Jon Doyle

Abstract: This essay outlines the technique of recording the inferential steps taken by a program as **dependency records** *linking conclusions with their antecedents. Dependencies are crucial in providing explanation capabilities, in transferring expertise, and in learning. In addition, these inference records can be used to maintain the currently active set of program database elements when new inferences, actions, or assumptions are made or changed. Further, dependencies can aid in controlling the program's actions by representing the inferential connections between assumptions, goals, and actions.*

Introduction

One important requirement on the design of intelligent or expert programs is that they be responsibly humble to some degree about their conclusions and actions; that they be able to explain their conclusions and actions in terms of the information supplied by their informants. This ability to defer responsibility for conclusions to their sources is crucial in transferring expertise from an expert to a program, for the expert must be able to assign credit or blame for unexpected conclusions and actions to forgotten, missing, or erroneous database elements or procedures. Similarly, this humility is crucial in learning by the program, for the program must be able to analyze the reasons for its own successes and failures.

These considerations alone are enough to suggest that intelligent programs should record information about their inferences, so as to be able to refer to these records during credit or blame assignment. But in fact, several other considerations also suggest keeping these dependency records, namely the control of program actions, the adoption and abandonment of assumptions, and the recovery of imprudently abandoned conclusions.

Recorded dependencies aid in controlling the actions of a program, because credit assignment is important in search as well as in transfer of expertise and learning. One classical search technique is chronological backtracking, in which the sequence of hypothetical situations being searched through is annotated with the actions or assumptions which lead to each successive situation. An error or inconsistency reached in one of these situations signals for backtracking, that is, retracting all actions and assumptions since the most recent choice point, and then proceeding with the next alternative for that choice. Because errors and inconsistencies are often discovered in situations in which the temporal order of previous choices, actions, and assumptions is irrelevant, chronological backtracking is needlessly inefficient. Instead of tracing errors and inconsistencies to their relevant sources, for example by tracing recorded dependencies, chronological backtracking often searches throught numerous irrelevant

combinations of assumptions and actions.

In addition to their credit assignment applications, recorded dependencies can be used to maintain the set of currently active database elements. Often in the past this task was accomplished by manual changes, contexts, or procedures triggering on the addition and erasure of database entries. Each of these aproaches had problems. Manual changes made difficult the updating of consequences of changed entries. Contexts shared the problems of chronological backtracking (in which they played an important role) of needlessly discarding information discovered in an abandoned search. And finally, change-triggered procedures had to be carefully tailored as a set to avoid unintended infinite loops of adding and erasing sets of database entries. By allowing a global static perspective on the actions of the program's inference procedures, dependencies allow the coherent treatment both of new inferences and of assumptions made on the basis of incomplete information. The techniques described below allow adopting an assumption when necessary on the basis of incomplete information, the subsequent abandoning of the conclusions drawn from the assumption if it is abandoned, and the recovery of the previously abandoned conclusions if the assumption is later reinstated.

Dependency records were first employed in robot problem solving programs as aids to the erasure of consequential database entries following actions or failures. Action effects were typically represented in add/delete lists. Fikes [1975] kept track of derivations so that all consequences of a database entry could be erased when that entry was deleted by order of a delete list. Hayes [1975] kept track of the dependence of each planning decision on other such decisions, so that all consequential decisions could be abandoned when some plan execution error or independent worldly change invalidated the preconditions of a decision about some plan step.

The role of dependencies was then widened by Stallman and Sussman [1977], who in the context of electronic circuit analysis employed dependencies in explanations of conclusions, in abandoning and retrieving consequences of abandoned or reinstated assumptions, and in dependency-directed or non-chronological backtracking. London [1978] developed similar techniques for use in robot planning, improved to conduct incremental revisions of conclusions.

Building on the work of Stallman and Sussman, Doyle [1979] introduced assumptions as dependency records of non-monotonic inferences, such as Planner's THNOT (see [McDermott and Doyle 1980] and the following essay), and identified the role of such assumptions in dependency-directed backtracking. De Kleer, *et al.*[1979] and Shrobe [1979] illustrated the use of dependencies in explicitly guiding program actions. Doyle [1980] extended this use to include a process of decision-making based on dialectical argumentation (see the preceding essay).

The reader is advised to consult the survey and bibliography by Doyle and London [1980] for descriptions of papers concerned with these techniques.

Recording and Maintaining Dependencies

The fundamental data-structures involved in dependency records are nodes and justifications. Nodes

are used to label program database entries, inference rules, procedures, etc., for use in justifications. Justifications, in turn, represent inference steps from combinations of nodes to another node, or properly, from the referents of combinations of nodes to the referent of another node. The simplest sort of justification is a list of antecedent nodes. For example, if a program labelled the statements "The patient has a cold" and "The patient has a cold implies the patient sneezes" *Node-1* and *Node-2*, labelled the *Modus Ponens* inference rule "If A, and $A \supset B$, then B" *Node-3*, the program might infer a new statement "The patient sneezes," labelled *Node-4*, and record the list *(Node-1 Node-2 Node-3)* as a justification of *Node-4*.

The currently active set of nodes can be defined in terms of the current set of justifications. A node is currently active if it has a valid justification, where a justification is valid if each of the nodes it mentions is currently active. Newly created nodes are thus inactive. Justifications mentioning no antecedent nodes are always valid, and the nodes they justify are called premises. Premises are always active, and form a base from which all other currently active nodes may be explained in terms of valid justifications.

The currently active set of nodes controls the program actions. For example, instead of using procedures triggering on the addition or erasure of a database entry, a program might use procedures triggering on the inactive-active or active-inactive transitions of a node. While nothing prevents the justification of one node in terms of currently inactive nodes, the normal use is to draw conclusions from currently active nodes.

The currently active set of nodes is updated whenever justifications are added or erased by a revision procedure, which we will call TMS. (TMS stands for the misnomer "truth maintenance system." TMS has nothing to do with truth.) This revision process is also referred to as "dependency-network maintenance" or "belief revision" in the literature. If a justification J for node N is added, TMS checks to see if each node in J is active. If so, N is made active, and recursively, all inactive nodes with justifications mentioning N are re-examined to see if they too can be made active. If a valid justification for N is erased, a list is made of N together with all the active nodes depending on N via other valid justifications. Each of these nodes is made inactive. Then, each node on the list is re-examined to see if it can be made active. This deactivation and re-examination is used to avoid making nodes active on the basis of circular arguments. TMS avoids circular arguments to ensure that all nodes are consequences of explicit premises and assumptions, rather than consequences of the implicit premises made by TMS. This restriction amounts to avoiding errors like concluding A and B from the axioms $A \supset B$ and $B \supset A$.

Although one may erase justifications, e.g. to change the current set of premises, this is bad practice whose desired end is properly achieved by using non-monotonic justifications to create defeasible assumptions. On the face of it, the easiest way to change the basis of the current set of active nodes is for the program to add and erase premise justifications as desired. However, erasing justifications throws away valuable information about past inferences. In addition, this simplistic approach puts the burden on the program of keeping track of the dependence of past and potential premises on each other, so that premises can be manipulated in a coherent manner. But this is just the burden that dependencies are

designed to shoulder. Rather than effectively replicating the mechanisms of TMS within the program, we use non-monotonic justifications to create assumptions instead of using premise justifications to create premises. A non-monotonic justification is a pair *(A, I)* of lists of nodes, and is valid only if each node in *A* is active and each node in *I* is inactive. For example, to assume the statement "The patient has normal digestion," labelled *Node-5,* a justification *((),(Node-6))* might be created for *Node-5,* where *Node-6* labels the negation of the first statement, that is, "The patient has abnormal digestion." As long as no valid justification is known for this latter statement, the former would be active. These justifications are called non-monotonic because the set of active nodes can shrink upon the addition of a new justification. Non-monotonic justifications allow the effect of erasing justifications without losing information. For example, instead of making the normal digestion statement above a premise, it might be an assumption as shown, and defeated by providing a valid justification for its negation.

Although the extension of TMS to handling non-monotonic justifications may appear to be a simple task, the surprising subtlety of the many problems involved make this extension a nest of traps for the unwary. Two key reasons for the delicacy of this matter are that inactive nodes have consequences and that ambiguities of revision must be settled. Although a node may be inactive, other nodes through non-monotonic justifications may depend on its inactivity. Hence, a change of that node from inactive to active invalidates those justifications, thus making inactive the nodes they support. Also, for example, suppose that *Node-7* has the justification *((),(Node-8))* and *Node-8* has the justification *((),(Node-7)).* TMS must choose one of *Node-7* and *Node-8* to be active. Great care is required in cases like this since other nodes, possibly including other assumptions and contradictory nodes, may depend on some of these choices. TMS must be able to sift through the global ramifications of such local ambiguities. The details of how this can be done are beyond the scope of this article. (See Doyle [1979].)

Assumptions enter dependency-directed backtracking by filling the role of choice points in chronological backtracking. For each active node, TMS maintains a distinguished non-circular explanation by picking one valid justification as the supporting justification. Thus an assumption is properly a active node whose supporting justification is non-monotonic. To reject a node, that is, to make it inactive, the program just traces through its supporting justification, chooses some underlying assumption as the culprit. It then adds a new justification for an inactive node mentioned in the supporting justification of the culprit assumption, thereby defeating the assumption. For example, suppose *Node-9* has the supporting justification *((),(Node-10 Node-11)).* This can be thought of saying make *Node-9* active first, and if that fails try Node-10 or Node-11. Backtracking might add the justification ((),(Node-11)) for *Node-10,* thereby defeating the assumption *Node-9* and setting up *Node-11* as the next alternative. As this example shows, assumptions may be defeated without discarding justifications, thereby avoiding loss of information about past assumptions. In other words, the set of justifications always grows monotonically, while giving rise to a non-monotonically changing set of active data-base entries.

Discussion

Many unanswered questions make the technique of dependencies and assumptions an actively developing

area of investigation. One unanswered question concerns what mechanisms should TMS use to resolve the ambiguities when several possible revisions present themselves. It seems clear that this choice of revision should be controllable in some way, but the details remain to be worked out. A second question is how the existing TMS algorithms might be improved, since each has certain deficiencies. Finally, a third question is how to organize systems so that dependency information can be conveniently recorded and used. Some proposals have been made (e.g. de Kleer, *et al.* [1979] and Doyle [1980]), but this question is still largely unexplored.

In spite of these unanswered questions, the technique of dependencies and assumptions is important because the solution of complex problems often demands making simplifying or heuristic assumptions, either about the problem itself or about the way to proceed in solving it. Later these assumptions can be reasoned about in several ways. One way is to correct the assumptions leading to inconsistencies or to fruitless paths of investigation. Other ways are to extend the solution of the simplified problem to that of the complex original problem, to contrast the import of alternate collections of assumptions, and to index abstract solutions to problems.

December 4, 1980
Palo Alto, California

References

[1] de Kleer, J., Doyle, J., Steele, G., and Sussman, G., 1979. Explicit control of reasoning, in *Artificial Intelligence: an MIT Perspective*, (P. H. Winston and R. H. Brown, eds.), Cambridge: MIT Press, Vol. 1, pp. 94-116.

[2] Doyle, J., 1979. A truth maintenance system, *Artificial Intelligence* **12**, 231-272.

[3] Doyle, J., 1980. A model for deliberation, action, and introspection, Cambridge: MIT Artificial Intelligence Laboratory, TR-581.

[4] Doyle, J., and London, P., 1980. A selected descriptor-indexed bibliography to the literature on belief revision, *SIGART Newsletter* **71**, 7-23.

[5] Fikes, R. E., 1975. Deductive retrieval mechanisms for state description models, *Proc. Fifth International Joint Conference on Artificial Intelligence*, 99-106.

[6] Hayes, P. J., 1975. A representation for robot plans, Proc. Fifth International Joint Conference on Artificial Intelligence, 181-188.

[7] London, P. E., 1978. Dependency networks as a representation for modelling in general problem solvers, Department of Computer Science, University of Maryland, TR-698.

[8] McDermott, D., 1978. Planning and acting, *Cognitive Science* **2**, 71-109.

[9] McDermott, D., and Doyle, J., 1980. Non-monotonic logic—I, *Artificial Intelligence* **13**, 41-72.

[10] Shrobe, H. E., 1979. Explicit Control of Reasoning in the programmer's apprentice, *Proc. 4th Workshop on Automated Deduction,* Austin, Texas, 97-102.

[11] Stallman, R. M., and Sussman, G. J., 1977. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis, *Artificial Intelligence* 9, 135-196.

Non-Deductive Reasoning and Non-Monotonic Logics

Jon Doyle

Abstract: Several forms of non-deductive reasoning have attracted careful scrutiny. Purely deductive reasoning techniques have long been recognized as inadequate for capturing all intelligent thought. Statistical and inductive reasoning, which concern inexact and generalizing reasoning, have received much study as possible extentions or alternatives to deductive reasoning. Non-monotonic reasoning, recently formalized in non-monotonic logics, is the latest developed extension to deductive reasoning. This section sketches the nature of, motivations of, and approaches to, non-monotonic logics.

The task of logic

The task of logic is the judgment of arguments. Once upon a time logic was the science of argumentation, the study of which arguments are good, and which not good. Different purposes engendered different conceptions of good. Arguments to convince capricious, distracted, and sometimes irrational humans were judged by the standards of effective rhetoric, which concern, among other things, the size, structure, motivation, and emotional impact of arguments and their steps. Inductive logics judged arguments making generalizations; statistical logics judged arguments dealing with frequencies and probabilities, and deductive logics judged arguments making restatements; that is, truth-preserving inferences.

While important insights were gained into the philosophical and practical questions underlying rhetorical, statistical, and inductive reasoning, perhaps the philosophically most striking advances were made in connection with deductive reasoning. Philosophers, logicians and mathematicians explored the powerful ideas of formal languages, truth-theoretic semantics, set theory, and the mathematics of formal systems, model and proof theory. These ideas proved so fruitful that logic for the most part came to be identified with deductive logic, the study of truth-preserving inferences. This identification grew so strong that many proposed non-deductive logics have been attacked as false logics. But logic is a science of thought and argument, not merely a science of truth-preserving inferences.

The task of non-monotonic logic

The task of non-monotonic logics is to judge cases of non-monotonic reasoning. Non-monotonic reasoning involves adopting assumptions which may have to be abandoned in light of new information. For example, a scheduling secretary may employ the inference rule that he should schedule each new meeting on the closest future Wednesday unless and until he finds reasons for scheduling the meeting otherwise.

While working out the week's schedule, the secretary may tentatively schedule the first meeting on the next Wednesday, only to later reschedule it, thereby abandoning his initial assumption, when he learns that a later meeting is requested for that Wednesday specifically to accommodate a visitor.

This sort of reasoning is called non-monotonic in contrast to the monotonicity of the set of theorems of a set of axioms in deductive logic. In deductive logic, the addition of new axioms to a set of axioms can never decrease the set of theorems. At most, the new axioms can give rise to new theorems, so that the set of theorems grows monotonically with the set of axioms. In non-monotonic logics, the set of theorems may both lose members as well as gain members when new axioms are added.

Reasoning by default

Two cases of non-monotonic reasoning have been studied; reasoning by *default*, and *reasoning* by circumscription.

The defaults of reasoning by default are statements or rules which, as in the above scheduling example, state that unless and until otherwise demonstrated, some statement is to be believed. Defaults can be found in many places in standard A.I. techniques. They are used in stating generalities to which exceptions may be acknowledged without catastrophe. For example, a default might be that all birds can fly, penguins and ostriches being exceptions. In structured knowledge representation systems, such defaults often take the form of default fillers of frame slots. For example, an airline reservation system might describe each customer with a "passenger" frame in which the "class" slot has the default value "coach." Defaults also enter into many knowledge representation systems implicitly through what is known as the *closed world assumption*. The closed world assumption is that all relationships not explicitly stated to hold do not hold. For example, typical procedures for inheriting statements in one frame from more general frames via "generalization" links assume that a frame is generalized only by those frames explicitly listed as generalizations, or, in turn, by their generalizations. Thus, if the "elephant" frame has a sole generalization link to the "mammal" frame, the inheritance procedures will search only "mammal" and not any other frames, in spite of the possibility that new generalization links may be attached to "elephant" later and would then be searched as well. Yet another use of defaults is in the typical "STRIPS assumption" that performed actions change none of the program's beliefs about the world except those explicitly listed in the description of the action. For example, a description of a robot's action of moving from one location to another would list only changes in beliefs about the robot's position. When the robot moves, the STRIPS assumption default would leave its belief about world geography intact.

Reasoning by circumscription

Another case of non-monotonic reasoning, which may well overlap defaults in some (or even all) cases, is that of parsimonious reasoning, or reasoning by circumscription. When reasoning about some problem,

one often assumes that the problem involves only those objects and relationships which it mentions, and no others. The inheritance procedures mentioned above made such an assumption (the closed world assumption) about the non-existence of unlisted generalization links and generalizing frames. As another example, in the well-known missionary and cannibal problem of traversing a river uneaten, one typically does not think of solutions involving bridges, rocket ships, handcuffs, murder of the cannibals, or holes in the boat. Another way of viewing the circumscription principle is as the assumption that all qualifications to the problem have been stated explicitly.

Formal characterizations of defaults

Two sorts of detailed formalizations of non-monotonic defaults have been proposed; Reiter's logic of defaults and McDermott and Doyle's non-monotonic logics.

Each of the logic of defaults and non-monotonic logics interprets "Default S" roughly as "S is provable unless and until S can be disproven." The difficulty with this interpretation is its circularity, that what can be inferred depends on what inference rules are applicable, while at the same time, what inference rules are applicable depends on what can be inferred. For example, suppose we decide to use only the ordinary logical rules of inference in attempting to disprove statements, and that the information to be captured consists of three statements: "Default A," "Default B," and "$\neg(A \wedge B)$." Here neither A nor B can be disproven using the ordinary rules of inference of logic, so we declare both A and B to be provable by means of the default statements. These two new conclusions are inconsistent with $\neg(A \wedge B)$. Instead of declaring the initial three statements to be inconsistent, the non-monotonic logics try to refine the notions of provability to say that there are two coherent interpretations of these axioms, namely one in which A and $\neg B$ are provable, and one in which B and $\neg A$ are provable. This is a big departure from ordinary logic, in which a single set of axioms has exactly one set of conclusions that can be drawn from it. The key problem addressed by the non-monotonic logics is that of providing some well-defined semantics for defaults which allows a single set of axioms and defaults to have several coherent interpretations.

In all the non-monotonic logics, the meanings of "provable" and "consistent" for a statement and a set of axioms are defined non-constructively by a mathematical definition of what "coherent" sets of conclusions are, relative to a given set of axioms and defaults. These definitions are non-constructive primarily because the coherent interpretations supplied by the logics are in general not even recursively enumerable. Roughly put, the logics declare that interpretations are found by adding in as many statements (assumptions) as possible, in accord with the defaults, but at the same time avoiding adding in so many assumptions as to produce an ordinary logical inconsistency. In the above example, for instance, the two coherent interpretations of the three statements are produced by adding in just one assumption, either A or B. By the time one assumption is added in, the negation of the other can be deduced by ordinary logical rules of inference, so the other assumption is ruled out, as it would lead to an inconsistency. This rough description of the semantics provided by the logics does not do them justice. For the precise definitions involved, the reader is referred to the original papers.

While the two approaches to formalizing defaults have much in common in the way they interpret defaults and in their major theoretical properties, they differ in logical form, as one approach formalizes defaults as inference rules, and the other as modal formulas. Unless one is vitally interested in logic for its own sake, or in pursuing the future development of better non-monotonic logics, these differences in logical form can be passed over as small differences in notation for capturing the same ideas.

For example, Reiter [1980] formalizes defaults by adjoining a new sort of inference rule called a default to an ordinary logic of statements and inference rules. Default inference rules are of the form "If P, and it is consistent to assume Q, then infer R," written $P{:}Q/R$, where P, Q, and R are ordinary formulas. Given conditions P, a default allows the inference of R providing that Q is not disprovable. With this notation, the simplest sort of default, that of "Assume A if it cannot be disproven" is written simply as $:A/A$, that is, P is empty and $Q{=}R{=}A$.

Instead of stating defaults as inference rules, McDermott and Doyle [1980, McDermott 1980] state defaults as modal formulas. They use an ordinary logical language extended by the unary modal operator "not disprovable." The analogue in non-monotonic logic of a default inference rule $P{:}Q/R$ of the logic of defaults is $P \wedge \textit{not-disprovable}\ Q \supset R$. Thus the simplest sort of default is stated in these non-monotonic logics as $\textit{not-disprovable}\ A \supset A$. Although we said earlier that non-monotonic logics and the logic of defaults are for many purposes syntactic variants, that is not really true. The modal non-monotonic logic formulations are, for better or worse, actually more expressive than the non-modal logic of defaults. This is because one can make statements about defaults, e.g.

$$\textit{not-disprovable}(\textit{not-disprovable}\ A \supset A) \supset (\textit{not-disprovable}\ A \supset A),$$

in non-monotonic logics, where in the logic of defaults no means exists for referring to the default inference rules.

The genesis of practical non-monotonic inference rules

Neither of these approaches says anything about which non-monotonic statements or rules should be used in representing information about a particular domain. The logics all leave that decision to the A.I. system designer. However, McCarthy and Dacey have each developed theories that seem to bear on the problem of formulating defaults. McCarthy formalizes reasoning by circumscription as an explicitly non-monotonic rule of inference. Dacey, on the other hand, formalizes his *theory of conclusions* in terms of classical decision theory, rather than in terms of non-monotonic reasoning.

The idea of circumscription, in McCarthy's [1980] treatment, becomes an inference rule for formulating sets of assumptions on the basis of the available information. The circumscription inference rule computes axiom schema from sets of axioms, axiom schema which can be applied to make a variety of assumptions. To circumscribe a set of axioms A with respect to some predicate P mentioned in A, one constructs a sentence schemata stating that the only objects satisfying P are those whose doing so follows

from the axioms A. All statements following via ordinary deductive rules of inference from that sentence schemata are said to be the conclusions reached by *circumscriptive inference* with respect to P from the original axioms A. For example, suppose we know only one red-haired person, our friend Jane. If we see someone looking like Jane in the crude sense of merely being red-haired, we might, via circumscription assume that that person is Jane, she being the only person we know fitting that description. This inference is non-monotonic, of course, since if we now learn that Jane has an identical twin sister Joan, we can no longer conclude that anyone who looks like Jane is Jane. Expressed formally in terms of McCarthy's circumscription, this example might be translated as follows. We start with the set of axioms $A = (red\text{-}haired(Jane))$ and circumscribe on the predicate "*red-haired*." The circumscription of this predicate in A is the axiom schema

$$\Phi(Jane) \wedge \forall x(\Phi(x) \supset red\text{-}haired(x)) \supset \forall x(red\text{-}haired(x) \supset \Phi(x)).$$

If we now substitute our only know instance of a red-haired person into this schema, that is, if we substitute the formula $x=Jane$ for $\Phi(x)$, we get

$$Jane=Jane \wedge \forall x(x=Jane \supset red\text{-}haired(x)) \supset \forall x(red\text{-}haired(x) \supset x=Jane).$$

The first two parts of this formula are true, and simplifying it leaves the resulting assumption or "default" $\forall x(red\text{-}haired(x) \supset x=Jane)$ which we can apply to any new person that looks like Jane (is red-haired). Yet this inference is non-monotonic, in that if we add the new axiom *red-haired(Joan)* to A, we can no longer draw any such identifying conclusion. At best, we can infer via another application of circumscription the less specific conclusion $\forall x(red\text{-}haired(x) \supset x=Jane \vee x=Joan)$.

Another approach to forming and rejecting tentative hypotheses, the *theory of conclusions* developed by Dacey [1978] after a suggestion of Tukey [1960], can be viewed as suggesting a general rule about when to adopt and abandon defaults. Dacey formulates conclusion theory in terms of classical decision theory rather than in the proof-theoretic terms of the preceding approaches. Classical decision theory analyzes how the strength of each of one's hypotheses about the world should be revised with each new evidential fact. The intent of conclusion theory is to avoid the continual re-evaluation of all hypotheses, to instead accept certain strong hypotheses as *conclusions*, and to hold these conclusions unless and until the introduction of very strong contrary evidence. Although Dacey apparently intends that the set of conclusions be the set of beliefs of the reasoner, his reasoner is isolated and unreflective, in that the rules of adoption and abandonment are used in developing scientific laws *de novo*. Once communication or summaries of conclusions are desired, as when writing an initially substantive A.I. program, the form of each conclusion seems to approximate that of a default. Thus conclusion theory might be adapted to the role of judging the propriety of adopting or abandoning defaults.

The mathematics of theory evolution

Each of the approaches above treats in detail primarily the atoms of reasoning, either individual inference steps or the sets of beliefs preceding and following the inference step. So far, much less attention has been

devoted to classifying the larger, more complex ways in which non-deductive inferences can change the current set of beliefs of a reasoner. The beginnings of a larger analysis of theory evolution are touched on by McDermott and Doyle [1980], Doyle [1979, 1980], Gumb [1978, 1979], and Weyhrauch [1980], and, typically less formally, in the philosophy of science literature in general (e.g., Quine and Ullian [1978]).

Conclusion

The area of non-monotonic reasoning has to date supplied more questions than definitive answers, but the questions it raises are vital. For further information, peruse the papers collected in the special issue of *Artificial Intelligence* on non-monotonic logic (such as those cited above), and the papers indexed in the bibliography by Doyle and London [1980]. Further fruitful formalizations of non-deductive reasoning techniques may well be awaiting discovery. For example, Collins [1978] and, less directly, Wason and Johnson-Laird [1972] investigate patterns of human non-deductive reasoning. Which of these may now succumb to formal analysis?

September 24, 1980
revised May 28, 1981
Palo Alto, California

References

[1] Collins, A., 1978. Fragments of a theory of human plausible reasoning, *Proc. Second Conf. on Theoretical Issues in Natural Language Processing,* 194-201.

[2] Dacey, R., 1978. A theory of conclusions, *Philosophy of Science* **45**, 563-574.

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

[4] Doyle, J., 1980. A model for deliberation, action, and introspection, Cambridge: MIT Artificial Intelligence Laboratory, TR-581.

[5] Doyle, J., and London, P., 1980. A selected descriptor-indexed bibliography to the literature on belief revision, *SIGART Newsletter* **71**, 7-23.

[6] Gumb, R. D., 1979. *Evolving Theories,* New York: Haven.

[7] Gumb, R. D., 1978. Summary of research on computational aspects of evolving theories, *SIGART Newsletter* **67**, 13.

[8] McCarthy, J., 1980. Circumscription—a form of non-monotonic reasoning, *Artificial Intelligence* **13**, 27-39.

[9] McDermott, D., 1980. Non-monotonic logic—II: Non-monotonic modal theories, Yale University, Department of Computer Science, Report 174.

[10] McDermott, D., and Doyle, J., 1980. Non-monotonic logic—I, *Artificial Intelligence* **13**, 41-72.

[11] Quine, W. V., and Ullian, J. S., 1978. *The Web of Belief,* second edition, New York: Random House.

[12] Reiter, R., 1978. On reasoning by default, *Proc. Second Conf. on Theoretical Issues in Natural Language Processing,* 210-218.

[13] Reiter, R., 1980. A logic for default reasoning, *Artifical Intelligence* **13**, 81-132.

[14] Tukey, J. W., 1960. Conclusions vs. decisions, *Technometrics* **2**, 423-433.

[15] Wason, P. C., and Johnson-Laird, P. N., 1972. *Psychology of Reasoning: Structure and Content,* Cambridge: Harvard.

[16] Weyhrauch, R. W., 1980. Prolegomena to a theory of mechanized formal reasoning, *Artificial Intelligence* **13**, 133-170.


FILMED
8