GENERALITY IN ARTIFICIAL INTELLIGENCE

John McCarthy
Computer Science Department
Stanford University
Stanford, CA 94305
jmc@cs.stanford.edu
http://www-formal.stanford.edu/jmc/

1971-1987

Abstract

My 1971 Turing Award Lecture was entitled "Generality in Artificial Intelligence". The topic turned out to have been overambitious in that I discovered that I was unable to put my thoughts on the subject in a satisfactory written form at that time. It would have been better to have reviewed previous work rather than attempt something new, but such wasn't my custom at that time.

I am grateful to the ACM for the opportunity to try again. Unfortunately for our science, although perhaps fortunately for this project, the problem of generality in AI is almost as unsolved as ever, although we now have many ideas not available in 1971. This paper relies heavily on such ideas, but it is far from a full 1986 survey of approaches for achieving generality. Ideas are discussed at a length proportional to my familiarity with them rather than according to some objective criterion.

It was obvious in 1971 and even in 1958 that AI programs suffered from a lack of generality. It is still obvious, and now there are many more details. The first gross symptom is that a small addition to the idea of a program

often involves a complete rewrite beginning with the data structures. Some progress has been made in modularizing data structures, but small modifications of the search strategies are even less likely to be accomplished without rewriting.

Another symptom is that no-one knows how to make a general database of common sense knowledge that could be used by any program that needed the knowledge. Along with other information, such a database would contain what a robot would need to know about the effects of moving objects around, what a person can be expected to know about his family, and the facts about buying and selling. This doesn't depend on whether the knowledge is to be expressed in a logical language or in some other formalism. When we take the logic approach to AI, lack of generality shows up in that the axioms we devise to express common sense knowledge are too restricted in their applicability for a general common sense database. In my opinion, getting a language for expressing general common sense knowledge for inclusion in a general database is the key problem of generality in AI.

Here are some ideas for achieving generality proposed both before and after 1971. I repeat my disclaimer of comprehensiveness.

1 REPRESENTING BEHAVIOR BY PROGRAM

Friedberg (1958 and 1959) discussed a completely general way of representing behavior and provided a way of learning to improve it. Namely, the behavior is represented by a computer program and learning is accomplished by making random modifications to the program and testing the modified program. The Friedberg approach was successful in learning only how to move a single bit from one memory cell to another, and its scheme of rewarding instructions involved in successful runs by reducing their probability of modification was shown by Herbert Simon (a now substantiated rumor froma 1987 personal communication) to be inferior to testing each program thoroughly and completely scrapping any program that wasn't perfect. No-one seems to have attempted to follow up the idea of learning by modifying whole programs.

The defect of the Friedberg approach is that while representing behaviors by programs is entirely general, modifying behaviors by small modifications to the programs is very special. A small conceptual modification to a behavior is usually not represented by a small modification to the program, especially if machine language programs are used and any one small modification to the text of a program is considered as likely as any other.

It might be worth trying something more analogous to genetic evolution; duplicates of subroutines would be made, some copies would be modified and others left unchanged. The learning system would then experiment whether it was advantageous to change certain calls of the original subroutine to calls of the modified subroutine. Most likely even this wouldn't work unless the relevant small modifications of behavior were obtainable by calls to slightly modified subroutines. It would probably be necessary to provide for modifications to the number of arguments of subroutines.

While Friedberg's problem was learning from experience, all schemes for representing knowledge by program suffer from similar difficulties when the object is to combine disparate knowledge or to make programs that modify knowledge.

2 THE GPS AND ITS SUCCESSORS

One kind of generality in AI comprises methods for finding solutions that are independent of the problem domain. Allen Newell, Herbert Simon and their colleagues and students pioneered this approach and continue to pursue it.

Newell and Simon first proposed the General problem Solver GPS in their (1957) (also see (Ernst and Newell 1969). The initial idea was to represent problems of some general class as problems of transforming one expression into another by means of a set of allowed rules. It was even suggested in their (1960) that improving GPS could be thought of as a problem of this kind. In my opinion, GPS was unsuccessful as a general problem solver, because problems don't take this form in general and because most of the knowledge about the common sense needed for problem solving and achieving goals is not simply representable in the form of rules for transforming expressions. However, GPS was the first system to separate the problem solving structure of goals and subgoals from the particular domain.

If GPS had worked out to be really general, perhaps the Newell and Simon predictions about rapid success for AI would have been realized. Newell's current candidate for general problem representation is SOAR (Laird, Newell and Rosenbloom 1987), which, as I understand it, is concerned with transforming one state to another, where the states need not be represented by expressions.

3 PRODUCTION SYSTEMS

The first production systems were done by Newell and Simon in the 1950s, and the idea was written up in their (1972). A kind of generality is achieved by using the same goal seeking mechanism for all kinds of problems, changing only the particular productions. The early production systems have grown into the current proliferation of expert system shells.

Production systems represent knowledge in the form of facts and rules, and there is almost always a sharp syntactic distinction between the two. The facts usually correspond to ground instances of logical formulas, i.e. the correspond to predicate symbols applied to constant expressions. Unlike logic-based systems, these facts contain no variables or quantifiers. New facts are produced by inference, observation and user input. Variables are reserved for rules, which usually take a pattern-action form. Rules are put in the system by the programmer or "knowledge engineer" and in most systems cannot arise via the action of the system. In exchange for accepting these limitations, the production system programmer gets a relatively fast program.

Production system programs rarely use fundamental knowledge of the domain. For example, MYCIN (Buchanan and Shortliffe 1974) has many rules about how to infer which bacterium is causing an illness based on symptoms and the result of laboratory tests. However, its formalism has no way of expressing the fact that bacteria are organisms that grow within the body. In fact MYCIN has no way of representing processes occuring in time, although other production systems can represent processes at about the level of the situation calculus to be described in the next section.

The result of a production system pattern match is a substitution of constants for variables in the pattern part of the rule. Consequently production systems do not infer general propositions. For example, consider the definition that a container is sterile if it is sealed against entry by bacteria, and all the bacteria in it are dead. A production system (or a logic program) can only use this fact by substituting particular bacteria for the variables. Thus it cannot reason that heating a sealed container will sterilize it given that a heated bacterium dies, because it cannot reason about the unenumerated set of bacteria in the container. These matters are discussed further in

(McCarthy 1984).

4 REPRESENTING KNOWLEDGE IN LOGIC

It seemed to me in 1958 that small modifications in behavior are most often representable as small modifications in beliefs about the world, and this requires a system that represents beliefs explicitly.

"If one wants a machine to be able to discover an abstraction, it seems most likely that the machine must be able to represent this abstraction in some relatively simple way" (McCarthy 1959).

The 1958 idea for increasing generality was to use logic to express facts in a way independent of the way the facts might subsequently be used. It seemed then and still seems that humans communicate mainly in declarative sentences rather than in programming languages for good objective reasons that will apply whether the communicator is a human, a creature from Alpha Centauri or a computer program. Moreover, the advantages of declarative information also apply to internal representation. The advantage of declarative information is one of generality. The fact that when two objects collide they make a noise may be used in particular situations to make a noise, to avoid making noise, to explain a noise or to explain the absence of noise. (I guess those cars didn't collide, because while I heard the squeal of brakes, I didn't hear a crash).

Once one decides to build an AI system that represents information declaratively, one still has to decide what kind of declarative language to allow. The simplest systems allow only constant predicates applied to constant symbols, e.g. $on(Block1, Block2)$. Next one can allow arbitrary constant terms, built from function symbols, constants and predicate symbols, e.g. $location(Block1) = top(Block2)$. Prolog databases allow arbitrary Horn clauses that include free variables, e.g. $P(x, y) \wedge Q(y, z) \supset R(x, z)$, expressing the Prolog in standard logical notation. Beyond that lies full first order logic including both existential and universal quantifiers and arbitrary first order formulas. Within first order logic, the expressive power of a theory depends on what domains the variables are allowed to range. Important expressive power comes from using set theory which contains expressions for sets of any objects in the theory.

Every increase in expressive power carries a price in the required complexity of the reasoning and problem solving programs. To put it another way, accepting limitations on the expressiveness of one's declarative information allows simplification of the search procedures. Prolog represents a local optimum in this continuum, because Horn clauses are medium expressive but can be interpreted directly by a logical problem solver.

One major limitation that is usually accepted is to limit the derivation of new facts to formulas without variables, i.e to substitute constants for variables and then do propositional reasoning. It appears that most human daily activity involves only such reasoning. In principle, Prolog goes slightly beyond this, because the expressions found as values of variables by Prolog programs can themselves involve free variables. However, this facility is rarely used except for intermediate results.

What can't be done without more of predicate calculus than Prolog allows is universal generalization. Consider the rationale of canning. We say that a container is sterile if it is sealed and all the bacteria in it are dead. This can be expressed as a fragment of a Prolog program as follows.

$$sterile(X) \text{:-} sealed(X), not alive\text{-}bacterium(Y, X).$$
$$alive\text{-}bacterium(Y, X) \text{:-} in(Y, X), bacterium(Y), alive(Y).$$

However, a Prolog program incorporating this fragment directly can sterilize a container only by killing each bacterium individually and would require that some other part of the program successively generate the names of the bacteria. It cannot be used to discover or rationalize canning — sealing the container and then heating it to kill all the bacteria at once. The reasoning rationalizing canning involves the use of quantifiers in an essential way.

My own opinion is that reasoning and problem solving programs will eventually have to allow the full use of quantifiers and sets and have strong enough control methods to use them without combinatorial explosion.

While the 1958 idea was well received, very few attempts were made to embody it in programs in the immediately following years, the main one being F. Black's Harvard PhD thesis of 1964. I spent most of my time on what I regarded as preliminary projects, mainly LISP. My main reason for not attempting an implementation was that I wanted to learn how to express common sense knowledge in logic first. This is still my goal. I might be discouraged from continuing to pursue it if people pursuing nonlogical approaches were having significant success in achieving generality.

(McCarthy and Hayes 1969) made the distinction between epistemological and heuristic aspects of the AI problem and asserted that generality is more easily studied epistemologically. The distinction is that the epistemology is completed when the facts available have as a consequence that a certain strategy is appropriate to achieve the goal, while the heuristic problem involves the search that finds the appropriate strategy.

Implicit in (McCarthy 1959) was the idea of a general purpose common sense database. The common sense information possessed by humans would be written as logical sentences and included in the database. Any goal-seeking program could consult the database for the facts needed to decide how to achieve its goal. Especially prominent in the database would be facts about the effects of actions. The much studied example is the set of facts about the effects of a robot trying to move objects from one location to another. This led in the 1960s to the *situation calculus* (McCarthy and Hayes 1969) which was intended to provide a way of expressing the consequences of actions independent of the problem.

The basic formalism of the situation calculus is

$$s' = result(e, s),$$

which asserts that s' is the situation that results when event e occurs in situation s. Here are some situation calculus axioms for moving and painting blocks.

Qualified Result-of-Action Axioms

$$\forall xls.clear(top(x), s) \wedge clear(l, s) \wedge \neg tooheavy(x) \supset loc(x, result(move(x, l), s)) = l$$

$$\forall xcs.color(x, result(paint(x, c), s)) = c.$$

Frame Axioms

$$\forall xyls.color(y, result(move(x, l), s)) = color(y, s).$$

$$\forall xyls.y \neq x \supset loc(y, result(move(x, l), s)) = loc(y, s).$$

$$\forall xycs.loc(x, result(paint(y, c), s)) = loc(x, s).$$

$$\forall xycs.y \neq x \supset color(x, result(paint(y, c), s)) = color(x, s).$$

Notice that all qualifications to the performance of the actions are explicit in the premisses and that statements (called frame axioms) about what doesn't change when an action is performed are explicitly included. Without those statements it wouldn't be possible to infer much about $result(e2, result(e1, s))$, since we wouldn't know whether the premisses for the event $e2$ to have its expected result were fulfilled in $result(e1, s)$.

Futhermore, it should be noticed that the situation calculus applies only when it is reasonable to reason about discrete events, each of which results in a new total situation. Continuous events and concurrent events are not covered.

Unfortunately, it wasn't very feasible to use the situation calculus in the manner proposed, even for problems meeting its restrictions. In the first place, using general purpose theorem provers made the programs run too slowly, since the theorem provers of 1969 (Green 1969) had no way of controlling the search. This led to STRIPS (Fikes and Nilsson 1971) which reduced the use of logic to reasoning within a situation. Unfortunately, the STRIPS formalizations were much more special than full situation calculus. The facts that were included in the axioms had to be delicately chosen in order to avoid the introduction of contradictions arising from the failure to delete a sentence that wouldn't be true in the situation that resulted from an action.

5 NONMONOTONICITY

The second problem with the situation calculus axioms is that they were again not general enough. This was the *qualification problem*, and a possible way around it wasn't discovered until the late 1970s. Consider putting an axiom in a *common sense database* asserting that birds can fly. Clearly the axiom must be *qualified* in some way since penguins, dead birds and birds whose feet are encased in concrete can't fly. A careful construction of the axiom might succeed in including the exceptions of penguins and dead birds, but clearly we can think up as many additional exceptions like birds with their feet encased in concrete as we like. Formalized nonmonotonic reasoning (see (McCarthy 1980, 1986), (Doyle 1977), (McDermott and Doyle 1980) and (Reiter 1980)) provides a formal way of saying that a bird can fly unless there is an abnormal circumstance and reasoning that only the abnormal circumstances whose existence follows from the facts being taken into account will be considered.

Non-monotonicity has considerably increased the possibility of expressing general knowledge about the effects of events in the situation calculus. It has also provided a way of solving the *frame problem*, which constituted another obstacle to generality that was already noted in (McCarthy and Hayes 1969). The frame problem (The term has been variously used, but I had it first.) occurs when there are several actions available each of which changes certain features of the situation. Somehow it is necesary to say that an action changes only the features of the situation to which it directly refers. When there is a fixed set of actions and features, it can be explicitly stated which features are unchanged by an action, even though it may take a lot of axioms. However, if we imagine that additional features of situations and additional actions may be added to the database, we face the problem that the axiomatization of an action is never completed. (McCarthy 1986) indicates how to handle this using *circumscription*, but Lifschitz (1985) has shown that circumscription needs to be improved and has made proposals for this.

Here are some situation calculus axioms for moving and painting blocks taken from (McCarthy 1986).

Axioms about Locations and the Effects of Moving Objects

$$\forall xes.\neg ab(aspect1(x,e,s)) \supset loc(x,result(e,s)) = loc(x,s)$$

asserts that objects normally do not change their locations. More specifically, an object does not change its location unless the triple consisting of the object, the event that occurs, and the situation in which it occurs are abnormal in $apect1$.

$$\forall xls.ab(aspect1(x,move(x,l),s))$$

However, moving an object to a location in a situation is abnormal in $aspect1$.

$$\forall xls.\neg ab(aspect3(x,l,s)) \supset loc(x,result(move(x,l),s)) = l$$

Unless the relevant triple is abnormal in $aspect3$, the action of moving an object to a location l results in its being at l.

Axioms about Colors and Painting

$$\forall xes.\neg ab(aspect2(x,e,s)) \supset color(x,result(e,s)) = color(x,s)$$

$$\forall xcs.ab(aspect2(x,paint(x,c),s))$$

$$\forall xcs.\neg ab(aspect4(x,c,s)) \supset color(x, result(paint(x,c),s)) = c$$

Thes three axioms give the corresponding facts about what changes the color of an object.

This treats the qualification problem, because any number of conditions that may be imagined as preventing moving or painting can be added later and asserted to imply the corresponding *ab aspect* .... It treats the frame problem in that we don't have to say that moving doesn't affect colors and painting locations.

Even with formalized nonmonotonic reasoning, the general commonsense database still seems elusive. The problem is writing axioms that satisfy our notions of incorporating the general facts about a phenomenon. Whenever we tentatively decide on some axioms, we are able to think of situations in which they don't apply and a generalization is called for. Moreover, the difficulties that are thought of are often *ad hoc* like that of the bird with its feet encased in concrete.

6 REIFICATION

Reasoning about knowledge, belief or goals requires extensions of the domain of objects reasoned about. For example, a program that does backward chaining on goals used them directly as sentences, e.g. $on(Block1, Block2)$, i.e. the symbol *on* is used as a predicate constant of the language. However, a program that wants to say directly that $on(Block1, Block2)$ should be postponed until $on(Block2, Block3)$ has been achieved, needs a sentence like $precedes(on(Block2, Block3), on(Block1, Block2))$, and if this is to be a sentence of first-order logic, then the symbol *on* must be taken as a function symbol, and $on(Block1, Block2)$ regarded as an object in the first order language.

This process of making objects out of sentences and other entities is called *reification*. It is necessary for expressive power but again leads to complications in reasoning. It is discussed in (McCarthy 1979).

7 FORMALIZING THE NOTION OF CONTEXT

Whenever we write an axiom, a critic can say that the axiom is true only in a certain context. With a little ingenuity the critic can usually devise a more general contex in which the precise form of the axiom doesn't hold. Looking at human reasoning as reflected in language emphasizes this point. Consider axiomatizing "on" so as to draw appropriate consequences from the information expressed in the sentence, "The book is on the table". The critic may propose to haggle about the precise meaning of "on" inventing difficulties about what can be between the book and the table or about how much gravity there has to be in a spacecraft in order to use the word "on" and whether centrifugal force counts. Thus we encounter Socratic puzzles over what the concepts mean in complete generality and encounter examples that never arise in life. There simply isn't a most general context.

Conversely, if we axiomatize at a fairly high level of generality, the axioms are often longer than is convenient in special situations. Thus humans find it useful to say, "The book is on the table" omitting reference to time and precise identifications of what book and what table. This problem of how general to be arises whether the general common sense knowledge is expressed in logic, in program or in some other formalism. (Some people propose that the knowledge is internally expressed in the form of examples only, but strong mechanisms using analogy and similarity permit their more general use. I wish them good fortune in formulating precise proposals about what these mechansims are).

A possible way out involves formalizing the notion of context and combining it with the circumscription method of nonmonotonic reasoning. We add a context parameter to the functions and predicates in our axioms. Each axiom makes its assertion about a certain context. Further axioms tell us that facts are inherited by more restricted context unless exceptions are asserted. Each assertions is also nonmonotonically assumed to apply in any particular more general context, but there again are exceptions. For example, the rules about birds flying implicitly assume that there is an atmosphere to fly in. In a more general context this might not be assumed. It remains to determine how inheritance to more general contexts differs from inheritance to more specific contexts.

Suppose that whenever a sentence p is present in the memory of a computer, we consider it as in a particular context and as an abbreviation for the sentence $holds(p, C)$ where C is the name of a context. Some contexts are very specific, so that Watson is a doctor in the context of Sherlock Holmes stories and a baritone psychologist in a tragic opera about the history of psychology.

There is a relation $c1 \leq c2$ meaning that context $c2$ is more general than context $c1$. We allow sentences like $holds(c1 \leq c2, c0)$ so that even statements relating contexts can have contexts. The theory would not provide for any "most general context" any more than Zermelo-Frankel set theory provides for a most general set.

A logical system using contexts might provide operations of *entering* and *leaving* a context yielding what we might call *ultra-natural deduction* allowing a sequence of reasoning like

$holds(p, C)$

$ENTER\ C$

p

$\vdots$

q

$LEAVE\ C$

$holds(q, C)$.

This resembles the usual logical natural deduction systems, but for reasons beyond the scope of this lecture, it is probably not correct to regard contexts as equivalent to sets of assumptions — not even infinite sets of assumptions.

All this is unpleasantly vague, but it's a lot more than could be said in 1971.

References

Black, F., (1964). *A Deductive Question Answering System*, Doctoral Dissertation, Harvard University.

Buchanan, B.G. and Shortliffe, E.H, eds., (1984). *Rule-based expert systems: the MYCIN experiments of the Stanford Heuristic Programming Project.*

Davis, Randall; Buchanan, Bruce; and Shortliffe, Edward (1977). Production Rules as a Representation for a Knowledge-Based Consultation Program, *Artificial Intelligence*, Volume 8, Number 1, February.

Doyle, J. (1977). Truth Maintenance Systems for Problem Solving, *Proc. 5th IJCAI*, p. 247.

Ernst, George W. and Allen Newell (1969). *GPS: A Case Study in Generality and Problem Solving*, Academic Press.

Fikes, R, and Nils Nilsson, (1971). STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, *Artificial Intelligence*, Volume 2, Numbers 3,4, January, pp. 189-208.

Friedberg, R.M. (1958). A Learning Machine, *IBM Journal of Research*, Volume 2, Number 1, January, pp. 2-13.

Friedberg, R.M., B. Dunham and J.H. North (1959). A Learning Machine, Part II, *IBM Journal of Research*, Volume 3, Number 3, July, pp. 282-287.

Green, C.(1969). Theorem-proving by Resolution as a Basis for Question Answering Systems, in *Machine Intelligence 4*, pp. 183-205 (eds. Meltzer, B. and Michie, D.). Edinburgh: Edinburgh University Press.

Laird, John E., Allen Newell and Paul S. Rosenbloom (1987). *Soar: An Architecture for General Intelligence*, *Artificial Intelligence 33*, pp. 1–64.

Lifschitz, Vladimir (1985). Computing Circumscription, in *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, Volume 1, 1985, pp. 121–127.

McCarthy, John (1959). Programs with Common Sense, in *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, London: Her Majesty's Stationery Office. (Reprinted in McCarthy 1990).

McCarthy, John and Patrick Hayes (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence, in B. Meltzer and D. Michie (eds), *Machine Intelligence 4*, Edinburgh University. (Reprinted in B. L. Webber and N. J. Nilsson (eds.), *Readings in Artificial Intelligence*, Tioga, 1981, pp. 431–450; also in M. J. Ginsberg (ed.), *Readings in Nonmonotonic Reasoning*, Morgan Kaufmann, 1987, pp. 26–45; also in McCarthy 1990.)

McCarthy, John (1979). First Order Theories of Individual Concepts and Propositions, in Michie, Donald (ed.) *Machine Intelligence 9*, Ellis Horwood. (Reprinted in McCarthy 1990.)

McCarthy, John (1984). Some Expert Systems Need Common Sense, in *Computer Culture: The Scientific, Intellectual and Social Impact of the Computer*, Heinz Pagels, ed. vol. 426, Annals of the New York Academy of Sciences. (Reprinted in McCarthy 1990.)

McCarthy, John (1980). Circumscription — A Form of Nonmonotonic Reasoning, *Artificial Intelligence*, Volume 13, Numbers 1,2. (Reprinted in B. L. Webber and N. J. Nilsson (eds.), *Readings in Artificial Intelligence*, Tioga, 1981, pp. 466–472; also in M. J. Ginsberg (ed.), *Readings in Nonmonotonic Reasoning*, Morgan Kaufmann, 1987, pp. 145–152; also in McCarthy 1990.)

McCarthy, John (1986). Applications of Circumscription to Formalizing Common Sense Knowledge *Artificial Intelligence 28*, pp. 89–116. (Reprinted in M. J. Ginsberg (ed.), *Readings in Nonmonotonic Reasoning*, Morgan Kaufmann, 1987, pp. 153–166; also in McCarthy 1990.)

McCarthy, John (1990). *Formalizing Common Sense*, Ablex 1990.

McDermott, D. and Doyle, J. (1980). Nonmonotonic Logic I. *Artificial Intelligence*, Volume 13, Numbers 1,2, pp. 41-72.

Newell, A., J. C. Shaw and H. A. Simon (1957). *Preliminary Description of General Problem Solving Program-I (GPS-I)*, CIP Working Paper #7, Carnegie Institute of Technology, Dec.

Newell, A., J. C. Shaw and H. A. Simon, (1960). A variety of intelligent learning in a General Problem Solver, in *Self-Organizing Systems*, Yovits M.C., and Cameron, S. eds., Pergammon Press, Elmford, N.Y., pp 153-189.

Newell, A. and H. A. Simon (1972). *Human Problem Solving*, Prentice-Hall.

Reiter, Raymond (1980). A Logic for Default Reasoning, *Artificial Intelligence*, Volume 13, Numbers 1,2, April.