understanding that are known to be adequate in a scientific sense. It follows that he cannot know that certain people at certain times do *not* understand in Parry-or Eliza-like ways. That is to say, he has no way of knowing that we do not ourselves sometimes function by means of "clever tricks".

Finally, of course we *interpret* responses "in a manner which may indeed allow (us) to conclude that (we) are being 'understood' . . . " We do it with people, and we do it with machines, because that is what understanding is about, and how could the world be otherwise? The basic flaw in McLeod's position is that, like a lot of people, scientific and lay, he believes the (1) there really is some definitive process or feeling called UNDERSTANDING or BEING-UNDERSTOOD, and (2) that we can know for absolute certainty when we experience it, and (3) we can therefore contrast this feeling with one we have about a machine that "appears" to understand. These assumptions are, alas, false, at least from any scientific point of view, and the fact that Humbert Dreyfus has given a sophisticated philosophical defense [*What Computers Can't Do*, Harper and Row, New York, 1972] of a position very like that of (1)-(3) above, does not make it any more plausible to anyone who believes that the only serious test we can have is how a system *behaves*.

If one sticks to this simple, but firm, principle of machine performance, then McLeod's position will only make sense if and when he can tell us what it would be like to know of any machine that it *really* understood, and didn't just *seem* to do so. I do not believe that this distinction makes much sense, largely because (1)-(3) are false assumptions, yet they are the unexamined foundations of those who argue like Mr. McLeod.

ARTIFICIAL INTELLIGENCE MEETS NATURAL STUPIDITY

Drew McDermott
MIT AI Lab Cambridge, Mass 02139

As a field, artificial intelligence has always been on the border of respectability, and therefore on the border of crackpottery. Many critics <Dreyfus, 1972>, <Lighthill, 1973> have urged that we are over the border. We have been very defensive toward this charge, drawing ourselves up with dignity when it is made and folding the cloak of Science about us. On the other hand, in private, we have been justifiably proud of our willingness to explore weird ideas, because pursuing them is the only way to make progress.

Unfortunately, the necessity for speculation has combined with the culture of the hacker in computer science <Weizenbaum, 1975> to cripple our self-discipline. In a young field, self-discipline is not necessarily a virtue, but we are not getting any younger. In the past few years, our tolerance of sloppy thinking has led us to repeat many mistakes over and over. If we are to retain any credibility, this should stop.

This paper is an effort to ridicule some of these mistakes. Almost everyone I know should find himself the target at some point or other; if you don't, you are encouraged to write up your own favorite fault. The three described here I suffer from myself. I hope self-ridicule will be a complete catharsis, but I doubt it. Bad tendencies can be very deep-rooted. Remember, though, if we can't criticize ourselves, someone else will save us the trouble.

Acknowledgment-- I thank the AI Lab Playroom crowd for constructive play.

Wishful Mnemonics

A major source of simple-mindedness in AI programs is the use of mnemonics like "UNDERSTAND" or "GOAL" to refer to programs and data structures. This practice has been inherited from more traditional programming applications, in which it is liberating and enlightening to be able to refer to program structures by their purposes. Indeed, part of the thrust of the structured programming movement is to program entirely in terms of purposes at one level before implementing them by the most convenient of the (presumably many) alternative lower-level constructs.

However, in AI, our programs to a great degree are problems rather than solutions. If a researcher tries to write an "understanding" program, it isn't because he has thought of a better way of implementing this well-understood task, but because he thinks he can come closer to writing the *first* implementation. If he calls the main loop of his program "UNDERSTAND", he is (until proven innocent) merely begging the question. He may mislead a lot of people, most prominently himself, and enrage a lot of others.

What he should do instead is refer to this main loop as "G0034", and see if he can *convince* himself or anyone else that G0034 implements some part of understanding. Or he could give it a name that reveals its intrinsic properties, like NODE-NET-INTERSECTION-FINDER, it being the substance of his theory that finding intersections in networks of nodes constitutes understanding. If Quillian <1969> had called his program the "Teachable Language Node Net Intersection Finder", he would have saved us some reading. (Except for those of us fanatic about finding the part on teachability.)

Many instructive examples of wishful mnemonics by AI researchers come to mind once you see the point. Remember GPS? <Ernst and Newell, 1969> By now, "GPS" is a colorless term denoting a particularly stupid program to solve puzzles. But it originally meant "General Problem Solver", which caused everybody a lot of needless excitement and distraction. It should have been called LFGNS -- "Local-Feature-Guided Network Searcher".

Compare the mnemonics in Planner <Hewitt,1972> with those in Conniver <Sussman and McDermott, 1972>:

Planner	Conniver
GOAL	FETCH & TRY-NEXT
CONSEQUENT	IF-NEEDED
ANTECEDENT	IF-ADDED
THEOREM	METHOD
ASSERT	ADD

It is so much harder to write programs using the terms on the right! When you say (GOAL . . .), you can just feel the enormous power at your fingertips. It is, of course, an illusion.

Of course, Conniver has some glaring wishful primitives, too. Calling "multiple data bases" CONTEXTS was dumb. It implies that, say, sentence understanding in context is really easy in this system.

LISP's mnemonics are excellent in this regard. <Levin *et. al.*, 1965> What if atomic symbols had been called "concepts", or CONS had been called ASSOCIATE? As it is, the programmer has no debts to pay to the system. He can build whatever he likes. There are some minor faults; "property lists" are a little risky; but by now the term is sanitized.

Resolution theorists have been pretty good about wishful mnemonics. They thrive on hitherto meaningless words like RESOLVE and PARAMODULATE, which can only have their humble, technical meaning. There are actually quite few pretensions in the resolution literature. <Robinson, 1965> Unfortunately, at the top of their intellectual edifice stand the word "deduction". This is *very* wishful, but not entirely their fault. The logicians who first misused the term (e.g., in the "deduction" theorem) didn't have our problems; pure resolution theorists don't either. Unfortunately, too many AI researchers took them at their word and assumed that deduction, like payroll processing, had been tamed.

Of course, as in many such cases, the only consequence in the long run was that "deduction" changed in meaning, to become something narrow, technical, and not a little sordid.

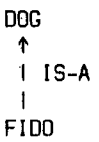
AI MEETS NS

As AI progresses (at least in terms of money spent), this malady gets worse. We have lived so long with the conviction that robots are possible, even just around the corner, that we can't help hastening their arrival with magic incantations. Winograd <1971> explored some of the complexity of language in sophisticated detail; and now everyone takes "natural-language interfaces" for granted, though none has been written. Charniak <1972> pointed out some approaches to understanding stories, and now the OWL interpreter includes a "story-understanding module". (And, God help us, a top-level "ego loop". <Sunguroff, 1975>)

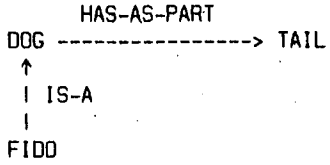
Some symptoms of this disease are embarrassingly obvious once the epidemic is exposed. We should avoid, for example, labeling any part of our programs as an "understander". It is the job of the text accompanying the program to examine carefully how much understanding is present, how it got there, and what its limits are.

But even seemingly harmless mnemonics should be handled gingerly. Let me explore as an example the ubiquitous "IS-A link", which has mesmerized workers in this field for years. <Quillian, 1968, Fahlman, 1975, Winograd, 1975> I shall take examples from Fahlman's treatment, but what I say is criticism of calling the thing "IS-A", not his work in particular.

An IS-A link joins two nodes in a "semantic net" (a by-now emasculated misnomer), thus:



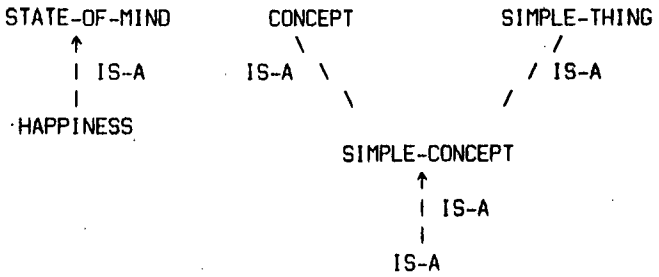
which is presumably meant to express "Fido is a dog". However, the *intrinsic* description of this link is "indicator-value pair inheritance link". That is, if the piece of network



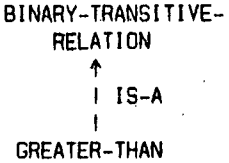
is present, then implicitly, "Fido has [a] tail" is present as well. Here HAS-AS-PART is the indicator, TAIL the value.

Most readers will think it extreme to object to calling this IS-A. Indeed, a self-disciplined researcher will be safe. But many people have fallen into the following IS-A traps:

Often, a programmer will shut his mind to other interpretations of IS-A, or conclude that IS-A is a very simple concept. Then he begins to write nonsensical networks like

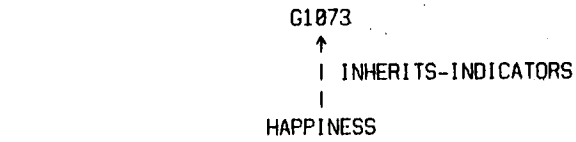


or suspiciously wishful networks like



This is an illustration of "contagious wishfulness": because one piece of a system is labeled impressively, the things it interacts with inherit grandiosity. A program called "THINK" is likely inexorably to acquire data structures called "THOUGHTS".

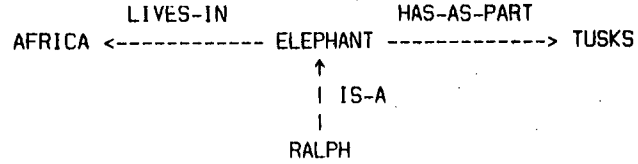
A good test for the disciplined programmer is to try using gensyms in key places and see if he still admires his system. For example, if STATE-OF-MIND is renamed G1073, we might have:



which looks much more dubious.

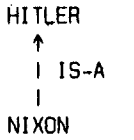
Concepts borrowed from human language must shake off a lot of surface-structure dust before they become clear. (See the next section of this paper.) "Is" is a complicated word, syntactically obscure. We use it with great facility, but we don't understand it well enough to appeal to it for clarification of anything. If we want to call attention to the "property inheritance" use, why not just *say* INHERITS-INDICATORS? Then, if we wish, we can prove from a completed system that this captures a large part of what "is a" means.

Another error is the temptation to write networks like this:



which people do all the time. It is clear to them that Ralph lives in Africa, the same Africa as all the other elephants, but his tusks are his own. But the network doesn't say this. Woods <1975> discusses errors like this in detail.

People reason circularly about concepts like IS-A. Even if originally they were fully aware they were just naming INHERITS-INDICATORS with a short, friendly mnemonic, they later use the mnemonic to conclude things about "is a". For example, although he is aware of complexities, Fahlman proposes that a first cut at representing "Nixon is a Hitler" is:



It worked for Fido and Dog, didn't it? But we just can't take stuff out of the IS-A concept that we never put in. I find this diagram worse than useless.

Lest this all seem merely amusing, meditate on the fate of those who have tampered with words before. The behaviorists ruined words like "behavior", "response", and especially, "learning". They now play happily in a dream world, internally consistent but lost to science. And think on this: if "mechanical translation" had been called "word-by-word text manipulation", the people doing it might still be getting government money.

Unnatural Language

In this section I wish to rail against a pervasive sloppiness in our thinking, the tendency to see in natural language a natural source of problems and solutions. Many researchers tend to talk as if an internal knowledge representation ought to be closely related to the "corresponding" English sentences; and that operations on the structure should resemble human conversation or "word problems". Because the fault here is a disregard for logic, it will be hard for my criticism to be logical and clear. Examples will help.

A crucial problem in internal representation is effective naming

AI MEETS NS

of entities. Although every entity can be given a primary name of some kind, much information about it will be derived from knowledge about roles it plays. If two persons marry and have children, then they play the role of parents in whatever data structure encodes knowledge of the family. Information about them (such as, "parents are older than their children") will be in terms of PARENT-1 and PARENT-2 (or "mother" and "father" if they are of opposite sexes). The naming problem is to ensure that information about PARENT-1 is applied to the primary name G0073 when it is discovered that G0073 shares a family with G0308.

The "natural-language fallacy" appears here in the urge to identify the naming problem with the problem of resolving references in English-language discourse. Although the two problems must at some remote intersection meet, it seems to me to be a waste of time to focus on their similarities. Yet it is hard to avoid the feeling that our ability to understand "the mother" to mean "Maria" is the same as the internal function of "binding" PARENT-1 to G0073. But it can't be.

The uses of reference in discourse are not the same as those of naming in internal representation. A good reason to have differing referential expressions in natural language is to pick out an object to talk about with the least amount of breath. After all, the speaker already knows exactly what he wants to refer to; if he says, "the left arm of the chair" in one place, "the arm" in another, and "it" in a third, it isn't because he thinks of this object in three different ways. But internally, this is exactly the reason for having multiple names. Different canonical data structures with different names for the constituent entities come to be instantiated to refer to the same thing in different ways. The internal user of such a structure must be careful to avoid seeing two things where one is meant.

In discourse, a speaker will introduce a hand and easily refer to "the finger". Frame theorists and other notation-developers find it marvelous that their system practically gives them "the finger" automatically as a piece of the data structure for "hand". As far as I can see, doing this automatically is the worst way of doing it. First, of course, there are four or five fingers, each with its own name, so "the finger" will be ambiguous.

Second, a phrase like "the finger" can be used in so many ways that an automatic evaluation to FINGER 109 will be wasteful at best. There are idioms to worry about, as in, "He raised his hand and gave me the finger". (Are we to conclude that the "default finger in the hand frame" is the middle finger?) But even ignoring them, there are many contexts where "the" just doesn't mean what we would like it to. For example, "He removed his glove and I saw the finger was missing". This is like, "The barn burned to the ground five years ago and was completely rebuilt". There are logics in which the same BARN 1051 can have different "denotations" in different time periods, but do we really want this clumsiness in the heart of our internal representation?

It seems much smarter to put knowledge about translation from natural language to internal representation in the natural language processor, not in the internal representation. I am using "in" loosely; my intent is to condemn an approach that translates language very superficially (using a little syntax and morphology) and hands it to the data base in that form. Instead, the language routine must draw on knowledge about all parts of the sentence in translating "the finger". Its output must be a directly useful internal representation, probably as remote as possible from being "English-like".

These problems stem from a picture of a program constructed of cooperating modules that "talk to" each other. While this may be a reasonable metaphor in some ways, anyone who has actually written such a program knows that "talking" is a very poor model of the communication. Yet many researchers (most extremely <Stansfield, 1975> and <Hawkinson,1975>) find English to be the ideal notation in which to encode messages. They are aware that message-passing channels are the most frustrating bottleneck through which intelligence must pass, so they wish their way into the solution: let the modules speak in human tongues! Let them use metaphor, allusion, hints, polite requests, pleading, flattery, bribes, and patriotic exhortations to their fellow modules!

It is hard to say where they have gone wronger, in underestimating language or overestimating computer programs. Language is only occasionally a medium of communication of information; even when it is, the ratio of information to packaging is low. The problem of a language speaker is to get the directed attention of an unprepared hearer and slide some information into his mind in a very short time. Since the major time sink is moving his mouth, the language sacrifices everything else to brevity; forcing the hearer to do much quick thinking to compensate. Furthermore, since the speaker doesn't quite know the organization of his hearer's mind, his phrasing of information and packaging must, except for the most stereotyped conversations, be an artwork of suggestiveness and insight.

Communication between computer programs is under completely different constraints. At the current stage of research, it is ridiculous to focus on anything but raw communication of information; we are unable to identify where more devious, Freudian intercourse might occur. Packaging and encoding of the information are usually already done. Ambiguity is avoidable. Even brevity is unimportant (at least for speed), since a huge structure can be transmitted by passing an internal name or pointer to it shared by sender and receiver. Instead, the whole problem is getting the hearer to notice what it has been told. (Not "understand", but "notice". To appeal to understanding at this low level will doom us to tail-chasing failure.) The new structure handed to the receiver should give it "permission" to make progress on its problem. If the sender could give more detailed instructions, it could just execute them itself. Unfortunately, the latitude this leaves the receiver is wasted if it is too "narrow-minded" to see the usefulness of what it has received. (The <1962> paper by Newell on these topics is still the best.)

Everyone who has written a large AI program will know what I am talking about. In this communication effort, the naming problem can be irritating, since the sender must make sure the receiver understands its terms. But there are so many approaches to solving the problem (for example, by passing translation tables around), which are not open to conversing humans, that it recedes quickly into the background. The frustrations lie elsewhere.

Reference is not the only "unnatural language" problem. A related one is the feeble analysis of concepts like "the" and "a" by most AI researchers. There is a natural inclination to let "the" flag a definite description and "a" an existential quantifier (or occasionally a description). Except for *Dick, Jane and Sally*, and some of Bertrand Russell's work, this approach is not even an approximation.

First the typical noun phrase is not directly translated into the internal representation at all, and does not wind up as an object name. For example, "Despite the peripatetic nature of American students and their families . . ., there remain wide gaps and serious misconceptions in our understanding of other peoples and cultures". (*Media and Methods 11*, No. 2, (1974), p. 43.) Translating this sentence (whose meaning is transparent) is problematic in the extreme. The author means to allude to the fact that Americans travel a lot, as a way of getting around to the claim that they don't travel enough or well enough. Why? We don't know yet why people talk this way. But translation methods that worked on "the

AI MEETS NS

big red block" will not succeed, and *cannot be extended to succeed*, on "the . . . nature of American students".

Second, the difference between "the" and "a" is not the difference between "definite" and "indefinite", except vacuously. For example, what is the difference in meaning between

"Due to the decrease in the American birthrate in the 1960's, our schools are underutilized".

"Due to a decrease in the American birthrate in the 1960's, our schools are underutilized".

In most respects, they "mean" exactly the same thing, since there can have been only one decrease in the birthrate in the 1960's, and each sentence presupposes that it occurred. But in one the author is assuming we know it already; in the other, he is more casual about whether we do or not. We have no theory at all about what difference this difference makes.

It is unfortunate that a logical back seepage has caused people to see words like "the", "a", "all", "or", "and", etc. as being embellished or ambiguous versions of "iota", "∃", "∀", "∨", and "∧". To cure yourself of this, try examining two pages of a book for ten-year olds, translating the story as you go into an internal representation. (I found <Kenny, 1963>, pp. 14-15 useful.) If you can do this without difficulty, your case is hopeless.

The obsession with natural language seems to have caused the feeling that the human use of language is a royal road to the cognitive psyche. I find this analogous to preoccupation with imagery as a way of studying vision. Most AI researchers react with amusement to proposals to explain vision in terms of stored images, reducing the physical eye to the mind's eye. But many of the same people notice themselves talking to themselves in English, and conclude that English is very close to the language of thought.

Clearly, there must be some other notation, different in principle from natural language, or we will have done for the ear what imagery theory does for the eye. No matter how fascinating the structure of consciousness is, it is dangerous to gaze too long into its depths. The puzzles we find there can be solved only by sneaking up on them from behind. As of now, we have no idea at all why people experience their thoughts the way they do, in pictures and words. It will probably turn out to be quite different, even simpler, than what we think now, once we understand why and how people experience their thoughts at all.

In the meantime, for many people, natural language has become the preferred means of stating problems for programs to solve. For example, research that began as a study of visual recognition becomes a study of how people come up with an animal that is white, has hooves, and one horn in the middle of its head. People can do this (and get "unicorn"), but the fact that they can obviously has nothing to do with visual recognition. In visual recognition, the main problems are guessing that you're looking at an animal in the first place, deciding that thing is a horn and that it belongs to the head, deciding whether to look for hooves, etc. The problem as stated in natural language is just not the same. (For example, the difficulties raised by the fact that I omitted presence or absence of wings from my description are different from the corresponding visual problems.)

Linguists have, I think, suffered from this self-misdirection for years. The standard experimental tool of modern linguistics is the eliciting of judgments of grammaticality from native speakers. Although anyone can learn how to make such judgments fairly quickly, it is plainly not a skill that has anything to do with ability to speak English. The real parser in your head is not supposed to report on its inputs' degree of grammaticality; indeed, normally it doesn't "report" at all in a way accessible to verbalization. It just tries to aid understanding of what it hears as best it can. So the grammaticality judgment task is completely artificial. It doesn't correspond to something people normally do.

Linguists, of course, have a place in their ontology for these judgments. They are a direct road to the seat of linguistic "competence". AI people find this notion dubious. They would be just as suspicious if someone claimed a good way to measure "visual recognition competence" was to measure the ability of a subject to guess where the cubes were in a scene presented to him as an English description of intensity contours. ("A big steep one in the corner, impetuous but not overbearing".)

Eventually, though, we all trick ourselves into thinking that the statement of a problem in natural language is natural. One form of this self-delusion that I have had difficulty avoiding is the "information-retrieval fixation". It dates from Winograd's <1971> analysis of questions like, "Do I like any pyramids?" as a simple PLANNER program like (THAND (THGOAL (LIKE WINOGRAD ?X)) (THGOAL (IS PYRAMID ?X))). This was entirely justified in the context he was dealing with, but clearly a stopgap. Nonetheless, nowadays, when someone invents a representation or deduction algorithm, he almost always illustrates it with examples like this, couched either in natural language or a simple translation like (THAND . . .).

This tight coupling of internal and external problem statements, if taken seriously, reduces the chance of progress on representation and retrieval problems. If a researcher tries to think of his problem as natural-language question answering, he is hurt by the requirement that the answers be the results of straightforward data-base queries. Real discourse is almost never of the literal-minded information-retrieval variety. In real discourse, the context leading up to a question sets the stage for it, and usually affects its meaning considerably. But, since the researcher is not really studying language, he cannot use the natural-language context. The only version of natural language he can have in mind must exclude this example of a conversation between two programmers on a system with six-letter file names:

"Where is the function TRY-NEXT defined?"

"In the file TRYNXT >". (pronounced TRY-NEXT)

"How do you spell 'TRY-NEXT'?"

"Omit the *e*".

Such contextual and intentional effects are distracting at best for the designer of a data base; presumably they are normally helpful to humans.

The other course is to concentrate on handling the query after it has been translated into (THAND . . .), but if this formula is still thought of as a direct translation of an English question, the approach ignores whatever framework a system might use to focus its computational attention. Generally a program builds or prunes its data structure as it goes, organizing it in such a way that most queries worth making at all can be handled with reasonable efficiency. Just picking the THAND problem out of the blue throws this organization away. This is what happens with the naive natural-language information-retrieval paradigm. A researcher who designs his retrieval algorithm around the case of a completely unmotivated formal query is likely to become preoccupied with problems like the efficient intersection of lists of likable objects and pyramids. <Nevins, 1974, Fahlman, 1975> In the design of programs whose knowledge is organized around problems, such issues are not nearly as important.

Someone must still work on the context-free English query problem, but there is no reason to expect it to be the same as the data-base retrieval problem. Besides, it might turn out that natural language is not the best notation for information retrieval requests. Perhaps we should postpone trying to get computers to speak English, and try programming librarians in PL/1!

In this section I have been harsh toward AI's tendency to oversimplify or overglorify natural language, but don't think that my

AI MEETS NS

opinion is that research in this area is futile. Indeed, probably because I am an academic verbalizer, I feel that understanding natural language is the most fascinating and important research goal we have in the long run. But it deserves more attention from a theoretical point of view before we rush off and throw together "natural-language" interfaces to programs with inadequate depth. We should do more studies of what language is for, and we should develop complex programs with a need to talk, before we put the two together.

"Only a Preliminary Version of the Program was Actually Implemented"**

A common idiocy in AI research is to suppose that having identified the shortcomings of Version I of a program is equivalent to having written Version II. <McDermott, 1974a, Sussman, 1975, Goldstein, 1974> Of course, the sincere researcher doesn't think of his actions this way. From my own experience, the course of a piece of research is like this:

Having identified a problem, the ambitious researcher stumbles one day upon a really good idea that neatly solves several related subproblems of it at once. (Sometimes the solution actually comes before the problem is identified.) The idea is formally pretty and seems to mesh smoothly with the way a rational program ought to think. Let us call it "sidetracking control structure" for concreteness. The researcher immediately implements an elegant program embodying automatic sidetracking, with an eye toward applying it to his original problem. As always, implementation takes much longer than expected, but matters are basically tidy.

However, as he develops and debugs this piece of code, he becomes aware that there are several theoretical holes in his design; and that it doesn't work. It doesn't work for good and respectable reasons, most of them depending on the fact that the solution to the problem requires more than one good idea. But, having gotten a framework, he becomes more and more convinced that those small but numerous holes are where the good ideas are to fit. He may even be right.

Here, however, he begins to lose his grip. Implementing Version I, whose shortcomings are all too obvious, was exhausting; it made him feel grubby for nothing. (Not at all like the TECO macros he took time out for along the way!) He feels as though he's paid his dues; now he can join the theoreticians. What's more, he *should*. Implementation details will make his thesis dull. The people want *epistemology*.

Simultaneously, he enjoys the contradictory feeling that the implementation of Version II would be easy. He has reams of notes on the holes in Version I and how to fill them. When he surveys them, he feels their master. Though a stranger to the trees, he can talk with confidence about the forest. Indeed, that is precisely what he does in his final document. It is full of allusions to a program he seems to be claiming to have written. Only in a cautious footnote does he say, "the program was never actually finished", or, "a preliminary version of the program was actually written".

This final report can have interesting quirks. It is likely to be titled *A Side-Tracking Control Structure Approach to Pornographic Question-Answering*, because the author's fondness for sidetracking never quite left him. However, sidetracking is the only part of the solution he really understands, so he is likely to be quite diffident about it. He feels much better about the multitude of patch mechanisms which he describes. He designed them as solutions, not problems; he wisely avoided implementing them and spoiling the illusion, so he can talk at length about how each one neatly ties up a loose end of sidetracking.

The final report usually pleases most people (more people than it should), impressing them but leaving them a little hungover. They are likely to be taken with sidetracking, especially if a theorem about it is proved, but the overall approach to the real problem lacks definition. Performance and promise run together like the colors of a sunset. The happy feeling is kindled in the reader that indefinite progress has already started. On the other hand, they usually know the author's approach won't solve everything; he avoids claiming this. So the document fails to stimulate or challenge; it merely feeds the addict's desire for reassurance that AI is not standing still, and raises his tolerance a little.

This muddle finally hurts those following in the researcher's path. Long after he has his Ph.D. or his tenure, inquiring students will be put off by the document he has left behind. He seems to have solved everything already, so the report says, yet there is no tangible evidence of it besides the report itself. No one really wants to take up the problem again, even though the original research is essentially a partial success or even a failure! If a student decides sidetracking is a good idea, and wants to study it, people will assume he is "merely implementing" an already fully designed program. (No Ph.D. for that!) He would be willing or even eager to start from a smoothly running Version II and write Version III, incorporating a new theoretical idea like Syntactic Network Data Bases, but there is no Version II. Even a Version I would help, but it isn't really working very well and its author has no desire for it to be publicized.

Of course, the student can turn his back on sidetracking, and develop an entirely new approach to Pornographic Question Answering. But this will only antagonize people. They thought they understood sidetracking; they had convinced themselves it could be made to work. Disagreeing will only confuse them. Besides, it probably could have been made to work. If only its inventor had left it an open question!

This inflationary spiral can't go on forever. After five theses have been written, each promising with fuzzy grandeur a different solution to a problem, people will begin to doubt that the problem has any solution at all. Five theses, each building on the previous one, might have been enough to solve it completely.

The solution is obvious: insist that people report on Version I (or possibly "I 1/2"). If a thorough report on a mere actual implementation were required, or even *allowed*, as a Ph.D. thesis, progress would appear slower, but it would be real.

Furthermore, the program should be user-engineered enough and debugged enough so that it can be run by people besides its author. What people want to know about such a program is how far they can diverge from the examples given in the thesis before it fails. Think of their awe when they discover that the hardest cases it handles weren't even mentioned! (Nowadays, the cases mentioned are, at the very best, the *only* ones the program handles.)

When a program does fail, it should tell the explorer why it failed by behavior more illuminating than, e.g., going into an infinite loop. Often a program will begin to degrade in time or accuracy before it fails. The program should print out statistics showing its opinion of how hard it had to work ("90,265 sidetracks"), so the user will not have to guess from page faults or console time. If he wishes to investigate further, a clearly written, up-to-date source program should be available for him to run interpretively, trace, etc. (More documentation should not be necessary.) In any other branch of computer science, these things are taken for granted.

My proposal is that thesis research, or any other two-year effort, should be organized as follows:

As before, a new problem, or *old problem with partial solution*, should be chosen. The part of the problem where most progress could be made (a conceptual "inner loop") should be thought about

AI MEETS NS

hardest. Good ideas developed here should appear in a research proposal.

The first half of the time allotted thereafter should be applied to writing Version n+1, where n is the version number you started with (0 for virgin problems). (Substantial rewriting of Version n should be anticipated.) The second half should be devoted to writing the report and improving Version n+1 with enough breadth, clean code, and new user features to make it useful to the next person that needs it.

The research report will then describe the improvements made to Version n, good ideas implemented, and total progress made in solving the original problem. Suggestions for further improvements should be included, in the future subjunctive tense.

The standard for such research should be a partial success, but AI as a field is starving for a few carefully documented failures. Anyone can think of several theses that could be improved stylistically and substantively by being rephrased as reports on failures. I can learn more by just being told why a technique won't work than by being made to read between the lines.

Benediction

This paper has focussed on three methodological and substantive issues over which we have stumbled. Anyone can think of more. I chose these because I am more guilty of them than other mistakes, which I am prone to lose my sense of humor about, such as:

1. The insistence of AI people that an action is a change of state of the world or a world model, and that thinking about actions amounts to stringing state changes together to accomplish a big state change. This seems to me not an oversimplification, but a false start. How many of your actions can be characterized as state changes, or are even performed to effect state changes? How many of a program's actions in problem solving? (Not the actions it strings together, but the actions it *takes*, like "trying short strings first", or "assuming the block is where it's supposed to be".)
2. The notion that a semantic network is a network. In lucid moments, network hackers realize that lines drawn between nodes stand for pointers, that almost everything in an AI program is a pointer, and that any list structure could be drawn as a network, the choice of what to call node and what to call link being arbitrary. Their lucid moments are few.
3. The notion that a semantic network is semantic.
4. Any indulgence in the "procedural-declarative" controversy. Anyone who hasn't figured this "controversy" out yet should be considered to have missed his chance, and be banned from talking about it. Notice that at Carnegie-Mellon they haven't worried too much about this dispute, and haven't suffered at all. The first half of <Moore and Newell, 1974> has a list of much better issues to think about.
5. The idea that because *you* can see your way through a problem space, your program can: the "wishful control structure" problem. The second half of <Moore and Newell, 1974> is a great example.

In this paper, I have criticized AI researchers very harshly. Let me express my faith that people in other fields would, on inspection, be found to suffer from equally bad faults. Most AI workers are responsible people who are aware of the pitfalls of a difficult field and produce good work in spite of them. However, to say anything good about anyone is beyond the scope of this paper.

References

Bobrow, D. G. and A. M. Collins (1975) (eds) *Representation and Understanding*, New York: Academic Press.

Charniak, E. (1972) "Toward a Model of Children's Story Comprehension", Cambridge: MIT AI Lab TR 266.

Dreyfus, H. L. (1972) *What Computers Can't Do: A Critique of Artificial Reason*, New York: Harper & Row.

Ernst, G. W. and A. Newell (1969) *GPS: A Case Study in Generality and Problem-Solving*, New York: Academic Press.

Fahlman, S. (1975) "Thesis Progress Report: A System for Representing and Using Real-World Knowledge", Cambridge: MIT AI Lab Memo 331.

Goldstein, I. (1974) "Understanding Simple Picture Programs", Cambridge: MIT AI Lab TR 294.

Hawkinson, L. (1975) "The Representation of Concepts in OWL", Cambridge: Project MAC Automatic Programming Group Internal Memo 17.

Hewitt, C. (1972) "Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot", Cambridge: MIT AI Lab TR-258.

Kenny, K. (1963) *Trixie Belden and the Mystery of the Blinking Eye*, Racine, Wisconsin: The Western Publishing Company, Inc.

Levin, M. I., J. McCarthy, P. W. Abrahams, D. J. Edwards, and T. P. Hart (1965) *LISP 1.5 Programmer's manual*, (Second edition) Cambridge: The M.I.T. Press.

Lighthill, J. (1973) "Artificial Intelligence: A General Survey", in *Artificial Intelligence: a Paper Symposium*, Science Research Council

McDermott, D. (1974a) "Assimilation of New Information by a Natural Language-Understanding System", Cambridge: MIT AI Lab TR 291.

McDermott, D. (1974b) "Advice on the Fast-Paced World of Electronics", Cambridge: MIT AI Lab Working Paper No. 71.

Minsky, M. (1968) *Semantic Information Processing*, Cambridge: MIT Press.

Moore, J. and A. Newell (1974) "How Can Merlin Understand?" in Gregg, L. (ed.) *Knowledge and Cognition*, Potomac, Maryland: Lawrence Erlbaum Associates.

Nevins, A. (1974) "A Relaxation Approach to Splitting in an Automatic Theorem Prover", Cambridge: MIT AI Lab Memo 302.

Newell, A. (1962) "Some Problems of Basic Organization in Problem-Solving Programs", in Yovitts, M., G. T. Jacobi, and G. D. Goldstein (eds.) *Self-Organizing Systems--1962*, New York: Spartan.

Quillian, M. R. (1968) "Semantic Memory", in Minsky <1968>.

Quillian, M. R. (1969) "The Teachable Language Comprehender", *Comm. ACM 12*, p. 459.

Robinson, J. A. (1965) "A Machine-oriented Logic Based on the Resolution Principle", *JACM 12*.

Stansfield, J. L. (1975) "Programming a Dialogue Teaching Situation", unpublished Ph.D. thesis, University of Edinburgh.

Sussman, G. J. and D. V. McDermott (1972) "From PLANNER to CONNIVER -- A Genetic Approach", *(Proc. FJCC 41*, p. 1171.

Sunguroff, A. (1975) Unpublished paper on the OWL system.

Sussman, G. J (1975) *A Computer Model of Skill Acquisition*, New York: American Elsevier.

Weizenbaum, J. (1975) *Computer Power and Human Reason*, Wm. Freeman Company.

Winograd, T. (1971) "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language", Cambridge: MIT AI Lab TR 84.

Winograd, T. (1975) "Frame Representations and the Declarative/ Procedural Controversy", in Bobrow and Collins <1975>.

Woods, W. A. (1975) "What's in a Link: Foundations for Semantic Networks", in Bobrow and Collins <1975>.