Artificial Intelligence 47 (1991) 185–250
Elsevier

On the thresholds of knowledge*

Douglas B. Lenat
MCC, 3500 W. Balcones Center, Austin, TX 78759, USA

Edward A. Feigenbaum
Computer Science Department, Stanford University, Stanford, CA 94305, USA

Received September 1987
Revised November 1989

Abstract

Lenat, D.B. and E.A. Feigenbaum, On the thresholds of knowledge, Artificial Intelligence 47 (1991) 185–250.

We articulate the three major findings and hypotheses of AI to date:

(1) The Knowledge Principle: If a program is to perform a complex task well, it must know a great deal about the world in which it operates. In the absence of knowledge, all you have left is search and reasoning, and that isn't enough.

(2) The Breadth Hypothesis: To behave intelligently in unexpected situations, an agent must be capable of falling back on increasingly general knowledge and analogizing to specific but superficially far-flung knowledge. (This is an extension of the preceding principle.)

(3) AI as Empirical Inquiry: Premature mathematization, or focusing on toy problems, washes out details from reality that later turn out to be significant. Thus, we must test our ideas experimentally, *falsifiably*, on large problems.

We present evidence for these propositions, contrast them with other strategic approaches to AI, point out their scope and limitations, and discuss the future directions they mandate for the main enterprise of AI research.

1. Introduction

For over three decades, our field has pursued the dream of the computer that competently performs various difficult cognitive tasks. AI has tried many approaches to this goal and accumulated much empirical evidence. The evidence suggests the need for the computer to have and use domain-specific knowledge. We shall begin with our definition of intelligence:

* This article was originally prepared in May 1987 for the MIT Workshop on Foundations of AI the following month, and issued then as MCC Technical Report AI-126-87. A very much shortened version was given as an invited paper at IJCAI-87 in Milan, August 1987. It was edited in 1989 in preparation for this publication.

Definition. *Intelligence* is the power to rapidly find an adequate solution in what appears a priori (to observers) to be an immense search space.

So, in those same terms, we can summarize the empirical evidence: "Knowledge is Power" or, more cynically "Intelligence is in the eye of the (uninformed) beholder". The *knowledge as power* hypothesis has received so much confirmation that we now assert it as:

Knowledge Principle (KP). A system exhibits intelligent understanding and action at a high level of competence primarily because of the *knowledge* that it can bring to bear: the concepts, facts, representations, methods, models, metaphors, and heuristics about its domain of endeavor.

The word *knowledge* in the KP is important. There is a tradeoff between knowledge and search; that is, often one can either memorize a lot of very detailed cases, or spend time applying very general rules. Neither strategy, carried to extremes, is optimal. On the one hand, *searching* is often costly, compared to the low cost of just not forgetting—of preserving the knowledge for future use. Our technological society would be impossible if everyone had to rediscover everything for themselves. On the other hand, even in a relatively narrow field, it's impractical if not impossible to have a pre-stored database of all the precise situations one will run into. Some at least moderately general knowledge is needed, rules which can be applied in a variety of circumstances. Since *knowledge* includes control strategies and inference methods, one might ask what is *excluded* by the KP. The answer is that we exclude unbalanced programs: those which do not contain, and draw power from, a mixture of explicit and compiled knowledge, and we advocate programs in which the balance is tipped toward the explicit, declarative side. Section 2 discusses the Knowledge Principle in more detail, and Section 3 provides experimental evidence for it.

The KP suggests that any system which is to perform intelligently incorporate both particular facts and heuristic rules. But how far-ranging must such knowledge be? Consider the brittleness of current knowledge-based systems. They have a plateau of competence, but the edges of that plateau are steep descents into complete incompetence. Evidence for how *people* cope with novelty is sparse and unreliable. Still, there is suggestive evidence supporting their reliance on general "commonsense" knowledge, and their reliance on partial or analogical matching. This leads us to a plausible extension of the Knowledge Principle:

Breadth Hypothesis (BH). Intelligent performance often requires the problem solver to fall back on increasingly general knowledge, and/or to analogize to specific knowledge from far-flung domains.

Are we, of all people, advocating the use of weak methods? Yes, but only in the presence of a breadth of knowledge far afield of the particular task at hand. We are adding to the KP here, not contradicting it. Much of the power still derives from a large body of task-specific expertise (cases and rules). We are adding to the KP a new speculation, namely that intelligent problem solvers cope with novel situations by analogizing and by drawing on "common sense". Section 4 examines the brittleness of current expert systems, and Section 5 presents evidence in support of the Breadth Hypothesis. That evidence comes from considering the limits of what AI can do today, in areas such as natural language understanding and machine learning.

The natural tendency of any search program is to slow down (often combinatorially explosively) as additional assertions are added and the search space therefore grows. All our real and imagined intelligent systems must, at some level, be *searching* as they locate and apply general rules and as they locate and perform analogical (partial) matches. Is it inevitable, then, that programs must become less intelligent in their previously-competent areas, as their KBs grow? We believe not. The key to avoiding excess search is to have a little meta-knowledge to guide and constrain the search. Hence, the key to preserving effective intelligence of a growing program lies in judicious adding of meta-knowledge along with the addition of object-level knowledge. Some of this meta-knowledge is in the form of meta-rules, and some of it is encoded by the ontology of the KB; these are, respectively, the dynamic and static ways of effectively preserving whatever useful bundlings already existed in the KB. (Of course, meta-rules can and should be represented explicitly, declaratively, as well as having a procedural form. That way, meta-meta-knowledge can apply to *them*; and so on.) This is a prescription for one to gradually add and refine categories and predicates (types of slots) as one grows the KB. This is why we believe the KP works "in the large", why we can scale up a KB to immense size without succumbing to the combinatorial explosion.

There is an additional element in our paradigm of AI research, which says that intelligence is still so poorly understood that Nature still holds most of the important surprises in store for us. This leads, in Section 6, to our central *methodological* tenets:

Empirical Inquiry Hypothesis (EH). The most profitable way to investigate AI is to embody our hypotheses in programs, and gather data by running the programs. The surprises usually suggest revisions that start the cycle over again. Progress depends on these experiments being able to *falsify* our hypotheses. Falsification is the most common and yet most crucial of surprises. In particular, these programs must be capable of behavior not expected by the experimenter.

Difficult Problems Hypothesis. There are too many ways to solve simple

problems. Raising the level and breadth of competence we demand of a system makes it *easier* to test—and raise—its intelligence.

The Knowledge Principle is a mandate for humanity to concretize the knowledge used in solving hard problems in various fields.[1] This *might* lead to faster training based on explicit knowledge rather than apprenticeships. It has *already* led to thousands of profitable expert systems.

The Breadth Hypothesis is a mandate to spend the resources necessary to construct one immense knowledge base spanning human consensus reality, to serve as scaffolding for specific clusters of expert knowledge.

The Empirical Inquiry Hypothesis is a mandate to actually try to build such systems, rather than theorize about them and about intelligence. AI is a science when we use computers the way Tycho Brahe used the telescope, or Michaelson the interferometer—as a tool for looking at Nature, trying to test some hypothesis, and quite possibly getting rudely surprised by finding out that the hypothesis is false. There is quite a distinction between using a tool to gather data about the world, and using tools to, shall we say, merely fabricate ever more beautiful crystalline scale models of a geocentric universe.

In Section 7, the various principles and hypotheses above combine to suggest a sweeping three-stage research program for the main enterprise of AI research:

(1) Slowly hand-code a large, broad knowledge base.
(2) When enough knowledge is present, it should be faster to acquire more from texts, databases, etc.
(3) To go beyond the frontier of human knowledge, the system will have to rely on learning by discovery, to expand its KB.

Some evidence is then presented that stages (1) and (2) may be accomplished in approximately this century; i.e., that artificial intelligence is within our grasp. Lenat's current work at MCC, on the CYC program [28], is a serious effort to carry out the first stage by the mid-1990s.

We are betting our professional lives—the few decades of useful research we have left in us—on KP, BH, and EH. That's a scary thought, but one has to place one's bets somewhere, in science. It's especially scary because:

(a) the hypotheses are not obvious to most AI researchers,
(b) they are unpalatable in many ways even to us, their advocates!

Why are they not obvious? Most AI research focuses on very small problems, attacking them with machinery (both hardware and search methods) that overpower them. The end result is a program that "succeeds" with very little

[1] Russell and others started a similar codification in the 1920s, but that movement was unfortunately led astray by Wittgenstein (see [41]).

knowledge, and so KP, BH, and EH *are irrelevant*. One is led to them only by tackling problems in difficult "real" areas, with the world able to surprise and falsify.

Why are our three hypotheses (KP, BH, EH) not particularly palatable? Because they are unaesthetic! And they entail person-centuries of hard knowledge-entry work. Until we are forced to them, Occam's Razor encourages us to try more elegant solutions, such as training a neutral net "from scratch"; or getting an infant-simulator and then "talking to it". Only as these fail do we turn, unhappily, to the "hand-craft a huge KB" tactic.

Section 8 summarizes the differences between our position and that of some other schools of thought on AI research. Section 9 lists several limitations and problems. We do not see any of them as insurmountable. Some of the problems seem at first blush to be "in-principle limitations", and some seem to be pragmatic engineering and usability problems. Yet we lump them side by side, because our methodology says to approach them all as symptoms of gaps in our (and our programs') knowledge, which can be identified and filled in incrementally, by in-context knowledge acquisition. Several of these problems have, in the two years since the first draft of this paper was prepared, been adequately "solved". The quote marks around "solved" mean that we have found adequate ways of handling them, typically by identifying a large collection of special-case solutions that cover the vast majority of occurrences in actuality.

The biggest hurdle of all has already been put well behind us: the enormous local maximum of building and using *explicit-knowledge-free* systems. On the far side of that hill we found a much larger payoff, namely expert systems. We have learned how to build intelligent artifacts that perform well, using knowledge, on specialized tasks within narrowly defined domains. An industry has been formed to put this technological understanding to work, and widespread transfer of this technology has been achieved. Many fields are making that transition, from data processing to knowledge processing.

And yet we see expert systems technology, too, as just a local maximum. AI is finally beginning to move on beyond that threshold. This paper presents what its authors glimpse on the far side of the expert systems local-maximum hill: the promise of a large, broad KB serving as the nucleus of crystallization for programs which respond sensibly to novel situations because they can reason more by analogy than by perfect matching, and, ultimately, because, like us, they understand the meanings of their terms.

2. The Knowledge Principle

There is a continuum between the power of already knowing and the power of being able to search for the solution. In between those two extremes lie,

e.g., generalizing and analogizing and plain old observing (for instance, noticing that your opponent is Castling). Even in the case of having to search for a solution, the *method* to carry out the search may be something that you already know, or partial-match to get, or search for in some other way. This recursion bottoms out in things (facts, methods, etc.) that are *already known*. Though the knowledge/search tradeoff is often used to argue for the primacy of search, we see by this line of reasoning that it equally well argues for the primacy of knowledge.

2.1. Thresholds of competence

Before you can apply search *or* knowledge to solve some problem, though, you need to already know enough to at least state the problem in a well-formed fashion. For each task, there is some minimum knowledge needed for one to even formulate it—that is, so that one can recognize when one has solved the problem.

Beyond this bare minimum, today's expert systems also include enough knowledge to reach the level of a typical practitioner performing the task. Up to that "competence" level, the knowledge/search tradeoff is strongly tipped in favor of knowledge. That is, there is an ever greater "payoff" to adding each piece of knowledge, up to some level of competence (e.g., where a useful subset of the original NP-complete problem becomes polynomial). Some of the knowledge that competent practitioners have is the knowledge of which distinctions to make and which ones to ignore. As shown by Polya [39] and Amarel [2], the space one needs to search for a solution to a problem can become smaller and smaller as one incorporates more and more such knowledge into the representation.

Beyond that "practitioner" level is the "expert" level. Here, each piece of additional knowledge is only infrequently useful. Such knowledge deals with rare but not unheard-of cases. In this realm, the knowledge/search tradeoff is fairly evenly balanced. Sometimes it's worth knowing all those obscure cases, sometimes it's more cost-effective to have general models and "run" them.

Notice that we have not yet considered "the rest of human knowledge", all the facts, heuristics, models, etc., that are not known to be relevant to this particular task. This does not mean that all other knowledge is truly irrelevant and useless to the task; perhaps it will one day be seen to be relevant through new discoveries, perhaps it will be useful to analogize from (and thereby lead to a novel solution to some tough situation), etc. Of course, putting this into an expert system for just one particular task is even *less* cost-effective, per piece of knowledge added, so no one seriously considers doing it.

2.2. Why the Knowledge Principle works so frequently

The above arguments describe how the KP *might* work; but why *does* it work

so frequently? In other words, why is building even conventional expert systems a powerful and useful thing to do?

(1) Many useful real-world tasks are sufficiently narrow that the "practitioner level" and even some degree of "expert level" can be achieved by a system containing only, say, several hundred if/then rules—hence requiring only a few person-years of effort.

(2) These systems often embody only the delta, the *difference* between expert and non-expert. MYCIN may outperform general practitioners at deciding which kind of meningitis a patient has, but that's because the GPs have to know about thousands of varieties of medical problems, and relatively rarely encounter meningitis, while the program can assume that the other 99% of medicine has been judged to be irrelevant, and is free to focus on how to differentiate one type of meningitis from another.

(3) Conventional programs that perform a similar task lack most of the "if" parts of the corresponding expert system's rules. That is, they have compiled away much of the knowledge, in order to gain efficiency. The price they pay for this, though, is the high cost of integrating a new piece of knowledge into their program once it exists. To put this the other way, you can never be sure, in advance, how the knowledge already in the system is going to be used, or added to, in the future. Therefore, much of the knowledge in an intelligent system needs to be represented explicitly, declaratively, although compiled forms of it may of course also be present. We might call this the "Explicit Knowledge Principle". In other words, the experts in a field often do not yet have all the required knowledge explicitly codified (otherwise anyone could be proficient, and there wouldn't *be* recognized experts). Therefore, standard software design methodology may fail to build a program "in one pass" to perform the task. However, as the developing expert system makes mistakes, the experts can correct them, and those corrections incrementally accrete the bulk of the hitherto unexplicated rules. In this manner, the system incrementally approaches competence, and even expertise, where no traditional software solution would work.

(4) There is another benefit that accrues when knowledge—including procedural knowledge—is represented declaratively, as explicit objects, following the "Explicit Knowledge Principle", above. Namely, *meta*-rules can apply to it, e.g., helping to acquire, check, or debug other rules. Structured objects that represent knowledge can be more easily analogized to, and can enable generalizations to be structurally induced from them. So while we concede to procedural attachment (having opaque lumps of code here and there in the system) for efficiency reasons, we argue that there should be a declarative version of that also (i.e., a declarative structure containing the information which is encoded in the procedure).

2.3. Control in knowledge-based systems

What about the control structure of an intelligent system? Even granted that lots of knowledge is necessary, might we not need sophisticated as-yet-unknown reasoning methods?

Knowledge Is All There Is Hypothesis. No sophisticated, as-yet-unknown control structure is required for intelligent behavior.

On the one hand, we already understand deduction, induction, analogy, specialization, generalization, and so on, well enough to have *knowledge* be our bottleneck, not control strategies. This does not mean that we fully understand such processes, of course. To be sure, additional work still needs to be done there. But we have examined them enough to eliminate the gross inefficiencies in their execution, to devise data structures and algorithms for efficiently performing them in the most commonly occurring cases. (For instance, consider Stickel's non-clausal connection graph resolution theorem prover [44], which was a response to the known inefficiency of deduction.)

On the other hand, all such strategies and methods are themselves just pieces of knowledge. The control structure of the intelligent system can be *opportunistic*: select one strategy, apply it for a while, monitor progress, and perhaps decide to switch to another strategy (when some other piece of knowledge suggests it do so).

Carefully reading our wording in this section will reveal that we are making a pragmatic argument, involving choices for where to focus current AI research, rather than making a hypothesis we expect to hold true forever. We don't understand induction, analogy, etc., perfectly, but further progress on understanding them needs to be done in the context of a large knowledge base. So let's worry about getting that built, and then return to study these phenomena. Perhaps at that time we will see the need to develop some useful new control scheme.

2.4. The manner in which knowledge boosts competence

Can we be more specific about the manner in which knowledge boosts competence? Can we give, say, an equation for how to measure the effective power of the knowledge in a system, when it's applied to a problem P? It is premature to even attempt to do so—it may *never* be possible to do so. It may never even be possible to give precise definitions for terms like "useful" and "competence". Nevertheless, this section speculates on what some of the terms in that equation would be.

Factor 1. Consider a heuristic H; e.g., "Drive carefully late at night". It has a characteristic curve of how powerful or useful it is, as a function of what

problem it's applied to. As detailed in [25], the area under this curve is often constant across heuristics. In simpler and more familiar terms, this is just the generality/power tradeoff: the more powerful a heuristic's "peak power" is, the narrower its domain of applicability is likely to be. A heuristic that only applies to driving on Saturday nights, in Austin, might be far more powerful than H, but its range of applicability is correspondingly narrower. As a first approximation to the power of the knowledge in the system, we might simply superpose all the "power curves" of the heuristics (and algorithms) that comprise the knowledge. That would give us an overall idea of the power of the system as a function of what problem it was applied to. If we're interested in applying the system to a particular problem P, we could then read off the value of this curve at point P. If we're going to apply the system to several problems, so that P is a large distribution, then we would weight the result by that distribution.

Factor 2. As a correction to this first rough guess, attempt to factor out some of the redundancy and dependence among the pieces of knowledge.

Factor 3. Weight each heuristic by how costly it is to run. "Cost" here includes literal CPU and memory resources used, and also includes the less tangible cost of asking questions of slow and busy human beings. Also included in this factor would be the downside risks of what might happen if the heuristic gave incorrect advice.

Factor 4. To be fair to the less-knowledge-based approaches, we should also deduct some amount which amortizes the effort we spent *acquiring* that rule or method.

Those represent just four of the factors in measuring the effective power of the knowledge in a system. We encourage further investigation in this direction. Recent work in nonmonotonic logic may bear on Factors 1 and 3; and [46] may bear on Factor 2.

3. Evidence for the Knowledge Principle

Half a century ago, before the modern era of computation began, Turing's theorems and abstract machines gave a hint of the fundamental idea that the computer could be used to model the symbol-manipulating processes that make up that most human of all behaviors: thinking.

Thirty years ago, following the 1956 Dartmouth Summer Conference on AI, the work began in earnest. The founding principle of the AI research paradigm is really an article of faith, first concretized by Newell and Simon: (See [35] for more details.)

Physical Symbol System Hypothesis. The digital computer has sufficient means

for intelligent action; to wit: representing real-world objects, actions, and relationships internally as interconnected structures of symbols, and applying symbol manipulation procedures to those structures.

The early dreaming included intelligent behavior at very high levels of competence. Turing speculated on wide-ranging conversations between people and machines, and also on expert-level chess playing programs. Newell and Simon also wrote about champion chess programs, and began working with Cliff Shaw toward that end. McCarthy wrote about the Advice Taking program. Gelernter, Moses, Samuel, and many others shared the dream.

Lederberg and Feigenbaum chose, in 1964, to pursue the AI dream by focusing on scientific reasoning tasks. With Buchanan and Djerassi, they built Dendral, a program that solved structure elucidation problems at a high level of competence. Many years of experimenting with Dendral led to some hypotheses about what its source of power might be, how it was able to solve chemical structure problems from spectral data. Namely, the program worked because it had enough knowledge of basic and spectral chemistry.

Table 1 shows that as each additional source of chemical knowledge was added, the Dendral program proposed fewer and fewer candidates (topologically plausible structures) to consider (see [7]). The fifth and final type of rule of thumb were rules for interpreting nuclear mass resonance (NMR) data. With all five types of rule in the program, many problems—such as the one illustrated—resulted in only a single candidate isomer being proposed as worth considering! Threatened by an a priori huge search space, Dendral managed to convert it into a tiny search space. That is, Dendral exhibited intelligence.

When searching a space of size 1, it is not crucial in what order you expand the candidate nodes. If you want to speed up a blind search by a factor of 43 million, one could perhaps parallelize the problem and (say, by 1995) employ a 43-mega-processor; but even back in 1965 one could, alternatively, talk with the human experts who routinely solve such problems, and then encode the knowledge they bring to bear to avoid searching. There is a cost associated with making the generator "smarter" in this fashion (i.e., there is inferencing

Table 1
Dendral at work: Finding all atom-bond graphs that could have the formula $C_{20}H_{43}N$. The sources given are cumulative; thus, the final "1" refers to Dendral with all five types of rules running in it.

Information source	Number of structures generated
Topology (limits of 3D space)	42,867,912
Chemical topology (valences)	14,715,814
Mass spectrography (heuristics)	1,284,792
Chemistry (first principles)	1,074,648
NMR (interpretation rules)	1

going on inside the generator, to utilize the knowledge it now contains) but that cost is insignificant compared to the seven orders of magnitude reduction in the size of the search space it permits.

Obvious? Perhaps, in retrospect. But at the time, the prevailing view in AI ascribed power to the reasoning processes, to the inference engine and not to the knowledge base. (E.g., consider LT and GPS and the flurry of work on resolution theorem provers.) The *knowledge as power* hypothesis, supported by Feigenbaum (Dendral), McCarthy (Advice Taker), and a few others, stood as a *contra*-hypothesis. It stood awaiting further empirical testing to either confirm it or falisfy it.

The 1970s were the time to start gathering evidence for or against the Knowledge Principle. Medical and scientific problem solving provided the springboard.

- Shortliffe's MYCIN program formed the prototype for a large suite of expert-level advisory systems which we now label "expert systems" [12]. Its reasoning system was simple (exhaustive backward chaining) and ad hoc in parts.
- DEC has been using and extending R1 program (EXCON) since 1981; its control structure is also simple: exhaustive forward chaining [32].
- Over a period of two decades, Bledsoe was led to incorporate more and more heuristics into his theorem provers, ultimately rejecting resolution entirely and opting for knowledge-guided natural deduction [3, 4].
- The INTERNIST program [40] got underway at nearly the same time as MYCIN. By now it has grown to a KB of 572 diseases, 4500 manifestations, and many hundreds of thousands of links between them.
- The AM [9] and EURISKO [25] programs, 15 years old by now, demonstrated that several hundred heuristic rules, of varying levels of generality and power, could adequately begin to guide a search for plausible (and often interesting) new concepts in many domains, including set theory, number theory, naval wargaming tactics, physical device design, evolution, and programming. These experiments showed how scientific discovery—a very different sort of intelligent behavior from most expert systems' tasks—might be explained as rule-guided, knowledge-guided search. Not all of the AM experiments were successful; indeed, the ultimate limitations of AM as it was run longer and longer finally led to EURISKO, whose ultimate empirical limitations [27] led to CYC, of which more later.

In the past decade, thousands of expert systems have mushroomed in engineering, manufacturing, geology, molecular biology, financial services, machinery diagnosis and repair, signal processing, and in many other fields. From the very beginning, these expert systems could interact with professionals in the jargon of the specialty; could explain their line of reasoning by displaying annotated traces of rule-firings; and had subsystems (such as

MYCIN's TEIRESIAS [9] and XCON's SALT [29]) which aided the acquisition of additional knowledge by guiding the expert to find and fix defects in the knowledge (rule) base.

Very little ties these areas together, other than that in each one, some technical problem solving is going on, guided by heuristics: experimental, qualitative rules of thumb—rules of good guessing. Their reasoning components are weak and simple; in their knowledge bases lies their power. The evidence for the various propositions we made in Section 2 lies in their details—in the details of their design, development, and performance.

In the 1980s, many other areas of AI research began making the shift over to the knowledge-based point of view. It is now common to hear that a program for understanding natural language must have extensive knowledge of its domain of discourse. Or, a vision program must have an understanding of the "world" it is intended to analyze scenes from. Or even, a machine learning program must start with a significant body of knowledge which it will expand, rather than trying to learn from scratch.

4. The Breadth Hypothesis

A limitation of past and current expert systems is their brittleness. They operate on a high plateau of knowledge and competence until they reach the extremity of their knowledge; then they fall off precipitously to levels of ultimate incompetence. People suffer the same difficulty, too, but their plateau is much broader and their slope is more gentle. Part of what cushions the fall are layer upon layer of weaker, more general models that underlie their specific knowledge.

For example, if engineers are diagnosing a faulty circuit they are unfamiliar with, they can bring to bear: circuit analysis techniques; their experiences with the other products manufactured by the same company, published handbook data for the individual components, and commonsense rules of thumb for water circuits (looking for leaks, or breaks), for electrical devices (turn it off and on a few times), and for mechanical devices in general (shake it or smack it a few times). Engineers might analogize to the last few times their automobile engine failed, or even to something as distant as a failed love affair. Naturally, the more different the causality of the thing they analogize to, the less likely it will be to apply in the electronic circuit diagnosis situation.

Domain-specific knowledge represents the distillation of experience in a field, nuggets of compiled hindsight. In a situation similar to the one in which they crystallized, they can powerfully guide search. But when confronted by a *novel* situation, human beings turn to reasoning strategies like generalizing and analogizing in real time and (even better) *already having* more general rules to fall back on. This leads to the Breadth Hypothesis (BH), which we stated in Section 1.

4.1. Falling back on increasingly general knowledge

Each of us has a vast storehouse of general knowledge, though we rarely talk about any of it explicitly to one another; we just assume that other people already know these things. If they're included in a conversation, or an article, they confuse more than they clarify. Some examples are:

- water flows downhill,
- living things get diseases,
- doing work requires energy,
- people live for a single, contiguous, finite interval of time,
- most cars today are riding on four tires,
- each tire a car is riding on is mounted on a wheel,
- if you fall asleep while driving, your car will start to head out of your lane pretty soon,
- if something big is between you and the thing you want, you probably will have to go around it.

It is *consensus reality* knowledge. Lacking these simple commonsense concepts, expert systems' mistakes often appear ridiculous in human terms. For instance, when a car loan authorization program approves a loan to a teenager who put down he'd worked at the same job for twenty years; or when a skin disease diagnosis program concludes that my rusted out decade-old Chevy has measles; or when a medical system prescribes an absurd dosage of a drug for a maternity patient whose weight (105) and age (35) were accidentally swapped during the case's type-in.

As we build increasingly complex programs, and invest them with increasing power, the humor quickly evaporates.

4.2. Reasoning by analogy

Reasoning by analogy involves *partial*-matching from your current situation to another one. There are two independent dimensions along which analogizing occurs, vertical (simplifying) and horizontal (cross-field) transformation.

- *Vertical*: When faced with a complex situation, we often analogize to a much simpler one. Of course simplification can be overdone: "the stock market is a seesaw"; "medication is a resource" (this leads many patients to overdose).
- *Horizontal*: Cross-field mapping is rarer but can pay off: "curing a disease is like fighting a battle" may help doctors devise new tactics to try (e.g., viruses employed to perform the analogue of propaganda) and may help soldiers devise new military tactics (e.g., choosing missions which function like vaccination).

Successful analogizing often involves components of both vertical and

horizontal transformation. For instance, consider reifying a country as if it were an individual person: "Russia is angry". That accomplishes two things: it simplifies dealing with the other country, and it also enables our vast array of first-hand experiences (and lessons learned) about inter-personal relations to be applied to international relations.

Do not make the mistake we did, of thinking of this reasoning method as little more than a literary device, used for achieving some sort of emotional impact. It can be used to help discover solutions to problems, and to flesh out solutions; and it can be argued that analogy pervades human communication and perhaps almost all of human thought! (see [22]). Even conceding that analogy is powerful, and often applies, still two questions linger: "*Why* does such an unsound problem-solving method work well?", and "Why does it work so often?"

There is much common causality in the world; that leads to similar events A and B; people (with our limited perception) then notice a little bit of that shared structure; finally, since we *know* that human perception is often limited, people come to rely on the following rule of thumb:

Analogical Method. If A and B appear to have some unexplained similarities, then it's worth your time to hunt for additional shared properties.

This rule is general but inefficient. There are many more specialized versions for successful analogizing in various task domains, in various user-modes (e.g., by someone in a hurry, or a child), among analogues with various epistemological statuses, depending on how much data there is about A and B, and so on. These are some of the n dimensions of analogy space; we can conceive having a special body of knowledge—an expert system—in each cell of that n-dimensional matrix, to handle just that sort of analogical reasoning.

Why focus on causality? If cause(A) and cause(B) have no specific common generalization, then similarities between A and B are more likely to be superficial coincidences, a metaphor useful perhaps as a literary device but not as a heuristic one.

Analogy in mathematics, where there is no clear notion of causality, operates similarly to "genuine" analogy, with the weaker relation of material implication substituting for causality. In that case, what one is often finding is a connection between two instances of a not-yet-conceptualized generalization. Also, much of analogizing in the doing of mathematics [39] is analogizing between the current problem-solving situation and a past one, i.e., between two search processes, not between two mathematical entities. And the act of trying to solve math problems is indeed frought with causality and, hence, opportunities for the above sort of "genuine" analogizing.

The above paragraphs are really just a rationalization of how analogy *might* work. The reason this unsound reasoning method *frequently* succeeds has to do

with three *moderation properties* that happen to hold in the real world:

(1) The moderate distribution of causes with respect to effects. If there were a vast number of unrelated kinds of causes, or if there were only one or two distinguishable causes, then analogy would be less useful.
(2) The moderately high frequency with which we must cope with novel situations, and the moderate degree of novelty they present. Lower frequency, or much higher (volatility of the world in which the problem solver must perform), would decrease the usefulness of trying to analogize. Why? In a world with essentially no surprises, memory is all you need; and in volatile world, matching to past occurrences is more of a hindrance than a help.
(3) The obvious metric for locating relevant knowledge—namely, "closeness of subject matter"—is just a moderately good predictor of true relevance. Far-flung knowledge and imagery *can* be useful. If we already understood all the connections, we'd always know when X was relevant; and if we had no attributes of knowledge to match to, we'd have no idea of how to generate (let alone flesh out) an analogy.

Analogizing broadens the relevance of the entire knowledge base. It can be used to construct interesting and novel interpretations of situations and data; to retrieve knowledge that has not been stored the way that it is now needed; to guess values for attributes; to suggest methods that just might work; and as a device to help students learn and remember. It can provide access to powerful methods that might work in this case, but which might not otherwise be perceived as "relevant". E.g., Dirac analogized between quantum theory and group theory, and very gingerly brought the group theory results over into physics for the first time, with quite successful results.

Today, we suffer with laborious manual knowledge entry in building expert systems, carefully codifying knowledge and placing it in a data structure. Analogizing may be used in the future not only as an inference method inside a program, but also as an aid to adding new knowledge to it.

5. Evidence for the Breadth Hypothesis

If we had as much hard evidence about the BH as we do for the KP, we would be calling it the Breadth *Principle*. Still, the evidence is there, if we look closely at the limits of what AI programs can do today. Most of the current AI research we've read about is currently stalled. As Mark Stefik recently remarked in a note to us, "*Progress will be held back until a sufficient corpus of knowledge is available on which to base experiments.*" For brevity, we will focus on natural language understanding (NL) and machine learning (ML), but similar results are appearing in most other areas of AI as well.

5.1. The limits of natural language understanding

To understand sentences in a natural language, one must be able to disambiguate which meaning of a word is intended, what the referent of a pronoun probably is, what each ellipsis means, and so on. These are knowledge-intensive skills.

1. I saw the Statue of Liberty flying over New York.
2. The box is in the pen. The ink is in the pen.
3. Mary saw a dog in the window. She wanted it.
4. Napolean died on St. Helena. Wellington was saddened.

Fig. 1. Sentences presume world knowledge furiously.

Consider the first sentence in Fig. 1. Who's flying, you or the statue? Clearly we aren't getting any clues from English to do that disambiguation; we must know about people, statues, passenger air travel, the size of cargo that is shipped by air, the size and location of the Statue of Liberty, the ease or difficulty of seeing objects from a distance, and numerous other consensus reality facts and heuristics. What if we'd said "I saw the Statue of Liberty standing in New York Harbor." It's not fair to say the verb (flying *versus* standing) decides it for you; consider, e.g., "I saw the Statue of Liberty standing at the top of the Empire State Building." See [45] for similar examples.

On line 2, in Fig. 1, the first "pen" is a corral, the other is a writing implement. But how do you know that? It has to do with storage of solids and liquids, of how big various objects are, with your ability to almost instantly and subconsciously consider *why* one might place a box in each kind of pen, *why* one might put ink inside each kind of pen, and choose the plausible interpretation in each case. This ability can of course be misled, as for example in one category of jokes.

On line 3, does "it" refer to the dog or the window? What if we'd said "She *smashed* it", or "She pressed her nose up against it"?

A program which *understood* line 4 should be able to answer "Did Wellington hear of Napoleon's death?" Often, we communicate by what *isn't* said, in between one sentence and the next one. And of course we should then be able to draw the obvious conclusions from those inferred assertions; e.g., being able to answer the question "Did Wellington outlive Napoleon?"

For any particular chosen text, an NL program can incorporate the small set of necessary twentieth century Americana, the few commonsense facts and scripts that are required for semantic disambiguation, question answering, anaphoric reference, and so on. But then one turns to a new page, and the new text requires more semantics (pragmatics) to be added.

In a sense, the NL researchers *have* cracked the language understanding problem. But to produce a general Turing-testable system, they would have to provide more and more domain-specific information, and the program's semantic component would more and more resemble the immense KB mandated by the Breadth Hypothesis. As Norvig [38] concludes: "the complexity [has been shifted] from the algorithm to the knowledge base, to handle examples that other systems could do only by introducing specialized algorithms."

Have we overstated the argument about how NL programs must ultimately have a large, real-world knowledge base to draw upon? Hardly; if anything we have drastically *under*stated it! Look at almost any newspaper story, e.g., and attend to how often a word or concept is used in a clearly metaphorical, non-literal sense. Once every few minutes, you might guess? No! Reality is full of surprises. The surprise here is that almost every sentence is packed with metaphors and analogies [22]. An unbiased sample: here is the first article we saw today (April 7, 1987), the lead story in the *Wall Street Journal* [50]:

> Texaco lost a major ruling in its legal battle with Pennzoil. The Supreme Court dismantled Texaco's protection against having to post a crippling $12 billion appeals bond, pushing Texaco to the brink of a Chapter 11 filing.

Lost? Major? Battle? Dismantled? Posting? Crippling? Pushing? Brink? The example drives home the point that, far from overinflating the need for real-world knowledge in language understanding, the usual arguments about disambiguation barely scratch the surface. (Drive? Home? The point? Far? Overinflating? Scratch? Surface? oh no, I can't call a halt to this! (call? halt?)) These layers of analogy and metaphor eventually "bottom out" at physical—*somatic* – primitives: up, down, forward, back, pain, cold, inside, seeing, sleeping, tasting, growing, containing, moving, making noise, hearing, birth, death, strain, exhaustion, . . . , and calling and halting.

NL researchers—and dictionaries—usually get around analogic usage by allowing several meanings to a word. Definition #1 for "war", say, is the literal one, and the other definitions are various common metaphorical uses of "war" (such as "an argument", "a commercial competition", "a search for a cure for", etc.).

There are many millions (perhaps a few hundred million) of things we authors can assume you readers know about the world: the number of tires an auto has; who Ronald Reagan is; what happens if you fall asleep when driving—what we called consensus reality. To use language effectively, we select the best consensus image to quickly evoke in the listener's mind the complex thought we want to convey. If our program doesn't already know most of those millions of shared concepts (experiences, objects, processes, patterns, . . .), it will be awkward for us to communicate with it in NL.

It is common for NL researchers to acknowledge the need for a large semantic component nowadays; Schank and others were saying similar things a

decade ago! But the first serious efforts have only recently begun to try to actually build one: at MCC (in conjunction with CYC), and at EDR, the Japanese Electronic Dictionary Research project [10]. We shall have to wait a few more years until the evidence is in.

5.2. *The limits of machine learning (induction)*

Machine learning is a second area where research is stalled owing to insufficiently broad knowledge bases. We will pick on AM and EURISKO because they exemplify the extreme knowledge-rich end of the current ML spectrum. Many experiments in machine learning were performed on them. We had many surprises along the way, and gained an intuitive feel for how and why heuristics work, for the nature of their power and their brittleness. Lenat and Brown present many of those surprises in [27]; some are listed in Fig. 2.

1. It works. Several thousand concepts, including some novel concepts and heuristics, from several domains, were discovered.
2. Most of the interesting concepts could be discovered in several different ways.
3. Performing the top N tasks on the Agenda in simulated-parallel provided only about a factor of 3 speedup even when N grew as large as 100.
4. Progress slows down unless the program learns new heuristics (compiles its hindsight) often.
5. Similarly, progress slowed down partly because the programs could not competently learn to choose, switch, extend, or invent different representations.
6. These programs are sensitive to the assumptions woven into their representations' semantics; e.g., "What does it *mean* for Jane to appear as the value on the spouse slot of the Fred frame?"
7. Some of their apparent power is illusory, only present in the mind of the intelligent observer who recognizes concepts which the program defines but does not properly appreciate.
8. Structural mutation works iff syntax mirrors semantics: represent heuristics using many small if- and many small then-parts, so the results of point mutation can be more meaningful.
9. In each new domain, there would be a flurry of plausible activities, resulting in several unexpected discoveries, followed by a period of decreased productivity, and finally lapsing into useless thrashing. The above techniques (e.g., 4, 5, 8) only delayed this decay.

Fig. 2. Some of the major surprises of the "discovery guided by heuristic rules" experiments performed with the AM and EURISKO programs, during the decade 1975–1984.

Despite their relative knowledge-richness, the ultimate limitations of these programs derived from their small size. Not their small number of methods, which were probably adequate, but the small initial knowledge base they had to draw upon. One can analogize to a campfire that dies out because it was too small, and too well isolated from nearby trees, to start a major blaze. The heuristics, the representations chosen, etc., provide the kindling and the spark, but the real fuel must come from without.

In other words, AM and other machine learning programs "ran down" because of insufficient knowledge. EURISKO partially solved this, by having new heuristics be learned simultaneously with the object-level learning, but this merely delayed the inevitable. The point here is that one can only learn something—by discovery or by being told—if one almost knows it already. This leads to Piagetian stages in young children, courses of study in young adults, and suggests the need for large knowledge bases in AI programs.

Marvin Minsky cites a variant of this relationship in his afterword to *True Names* [49]: "The more you know, the more (and faster) you can learn." The inverse of this enabling relationship is a disabling one, and that's what ultimately doomed AM and EURISKO:

Knowledge Facilitates Learning (Catch 22). If you don't know very much to begin with, don't expect to learn a lot quickly.

This is the standard criticism of pure Baconian induction. As philosophers are wont to say, "To get ahead, get a theory." Without one, you'll be lost. It will be difficult (or time-consuming) to determine whether or not each new generalization is going to be useful. In hindsight, perhaps we shouldn't have been surprised at this. After all, learning can be considered a task; and, like other tasks, it is subject to the Knowledge Principle.

This theme (knowledge facilitates learning) is filtering into ML in the form of explanation-based generalization (EBG) and goal-based learning. Unfortunately, EBG requires well-defined theories (too strong a requirement) and works by logically deducing (too restrictive a process) the explanandum. Hence we would expect this method of getting machines to learn to have some— but limited—success, which seems to be empirically what ML researchers report. E.g., Mostow [34] concludes that "scaling up for harder learning problems . . . is likely to require integrating [additional] sources of knowledge."

Don't human beings violate this Catch, starting as we do "from nothing"? Maybe, but it's not clear *what* human infants start with. Evolution has produced not merely physically sophisticated structures, but also brains whose architecture make us well suited to learning many of the simple facts that are worth learning about the world. Other senses, e.g., vision, are carefully tuned as well, to supply the brain with data that is already filtered for meaning (edges, shapes, motion, etc.) in the world in which we do happen to live. The exploration of those issues is beyond the scope of this paper, and probably

beyond the scope of twentieth century science, but one thing is clear: neonatal brains are far from *tabula rasae*.

Besides starting from well-prepared brain structures, humans also have to spend a lot of time learning. It is unclear what processes go on during infancy. Once the child begins to communicate by speaking, *then* we are into the symbolic sort of learning that AI has traditionally focused on. One theory of why it's difficult to remember one's infancy and young childhood is that we radically reorganize our knowledge once or twice during our early life, and the memory structures we built as infants are not interpretable by the retrieval and reasoning methods we use as an adult [33].

6. The Empirical Inquiry Hypothesis

We scientists have a view of ourselves as terribly creative, but compared to Nature we suffer from a poverty of the imagination; it is thus much easier for us to uncover than to invent. As we state elsewhere in this paper, experimentation must be hypothesis-driven; we are not advocating the random mixture of chemicals in the hope that lead transmutes to gold. But there is a difference between having theories as one's guide versus as one's master. Premature adherance to a theory keeps Nature's surprises hidden, washing out details that later turn out to be significant (i.e., either not perceiving them at all, or labeling them as anomalies [20] and then not attending to them). E.g., contrast the astonishing early empirical studies by Piaget (*Stages of Development*) with his subsequent five decades of barren attempts to mathematize them.

This attitude leads to our central methodological hypothesis, our paradigm for AI research: the Empirical Inquiry Hypothesis (EH). We stated it in Section 1, and repeat it here:

Empirical Inquiry Hypothesis (EH). Intelligence is still so poorly understood that Nature still holds most of the important surprises in store for us. So the most profitable way to investigate AI is to embody our hypotheses in programs, and gather data by running the programs. The surprises usually suggest revisions that start the cycle over again. Progress depends on these experiments being able to *falsify* our hypotheses. Falsification is the most common and yet most crucial of surprises! In particular, these programs must be capable of behavior not expected by the experimenter.

What do we mean by "a surprise"? Surely we wouldn't want to increase surprises by having more naive researchers, less careful thought and planning of experiments, sloppier coding, unreliable machines, etc. We have in mind astronomers getting surprised by what they see (and "see") through telescopes; i.e., things surprising to the professional. Early AI programs often surprised

their builders in this fashion; e.g., Newell, Simon, and Shaw's LT program [36] and Gelernter's geometry program [15]. Then fascination with axiomatizing and proving set in, and surprises from "the real world" became rare.

We have no objection to experimentation and theorizing proceeding hand in hand, we object only to the nearly exclusive doing of one of those activities and ignoring the other. As Genesereth and Nilsson argue in the preface of [17], having a good understanding of the theoretical issues can enable one to be a better experimenter.

The inverse to the EH is cruel:

Inverse to the Empirical Inquiry Hypothesis. If one builds programs which cannot possibly surprise him/her, then one is using the computer either

(a) as an engineering workhorse, or
(b) as a fancy sort of word processor (to help articulate one's hypothesis), or
(c) as a (self-)deceptive device masquerading as an experiment.

Most expert systems work falls into the first category; DART's use of MRS exemplifies the middle [16]; PUP5 (by the young Lenat [24]) and HACKER (by the young Sussman [47]) exemplify the latter category.

6.1. PUP5: a bad example

To illustrate this point, we will use some of our own earlier work. The PUP5 program [24] used a community of about one hundred Beings (similar to what have since been called actors and blackboard knowledge sources) to cooperate and synthesize a long LISP program, namely a variant of the Arch-learning program that Patrick Winston had written for his thesis a few years earlier.

That was the program that PUP5 was built to synthesize, the target it was to hit. We chose that target first, and wrote a clean version of the program in INTERLISP. Next, we wrote down an English dialogue in which a user talked to an idealized automatic program synthesis program which then gradually wrote the target program. Next, we analyzed the script of that dialogue, writing down the specific knowledge needed on the part of the synthesizer to handle each and every line that the user typed in. Finally, we encoded each of those pieces of knowledge, and bundled up the related ones into little actors or Beings.

Given this methodology, it should come as no surprise that PUP5 was then able to carry on that exact dialogue with a user, and synthesize that exact Arch program. We still firmly believe in the paradigm of multiple cooperating knowledge sources, it's just that our methodology ensured that there wouldn't be any surprises when we ran PUP5. Why? All along the way, there were numerous chances to cut corners, to consciously or unconsciously put down knowledge in a very specific form: just the knowledge that was needed, and in

just the form that it would be needed during the dialogue we know was going to be run. There wasn't much else PUP5 could do, therefore, besides hit its target, and there wasn't much that we learned about automatic programming or intelligence from that six-month exercise.

There was one crucial *meta*-level lesson we did learn: You can't do science if you just use a computer as a word processor, to illustrate your ideas rather than test them. That's the coarse form of the Empirical Inquiry Hypothesis. We resolved, in late 1974, to choose a task that eliminated or minimized the chance of building a wind-up toy like PUP5. We did not want a program whose target behavior was so narrow, so precisely defined, that it could "succeed" and yet teach us nothing. The AM program, written during 1975, was the direct result of Lenat's violent recoil from the PUP5 project.

There was no particular target behavior that AM was designed with; rather, it was an experiment: What would happen if a moderate-sized body of a few hundred math heuristics (about what were plausible directions to go in, about when something was and wasn't interesting) were applied in an agenda-managed best-first search, given an initial body of a hundred or so simple math concepts. In this sense, AM's task was less constrained than any program's had ever been: to explore areas of mathematics and do interesting things (gather data, notice regularities, etc.), with no preconceptions about what it might find or by what route it would find it. (Actually, we did have a few examples in mind for what AM might do, involving simple lattice theory and abstract algebra, but it never did those!)

Unlike PUP5, AM provided hundreds of surprises, including many experiments that led to the construction of EURISKO. EURISKO ran for several thousand CPU hours, in half a dozen varied domains (see Fig. 2, above). And again the ultimate limitation was not what we expected (CPU time), or hoped for (the need to learn new representations of knowledge), but rather something at once surprising and daunting: the need to have a large fraction of consensus reality already in the machine. In this case, the data led Lenat to the next project to work on—CYC—an undertaking we would have shied away from like the plague if the empirical evidence hadn't forced us to it. It has similarly led Feigenbaum to undertake his current line of research, namely building a large KB of engineering and scientific knowledge.

Thus, progress along our personal "paths of evolution" was due to running large experiments. As the Difficult Problems Hypothesis said in Section 1, *There are too many ways to solve simple problems. Raising the level and breadth of competence we demand of a system makes it* easier *to test and raise its intelligence.*

Much research in cognitive psychology, e.g., traditionally sidesteps hard-to-quantify phenomena such as scientific creativity or reading and comprehending a good book, in favor of very simple tasks such as remembering nonsense syllables. If a "messy" task *is* studied, then usually either (1) it is abstracted

and simplified almost beyond recognition [23], or (2) the psychologist focuses on (and varies) one specific variable, so "respectable" statistical tests for significance can be run.

6.2. Paradigms for AI research

Much of the confusion about AI methodology may be due to our casual mixing together of two quite different things: AI *goals* and AI *strategies* for achieving those goals. The confusion arises because many entries appear on both lists. Almost any strategy can apply toward any goal. Consider just one example: (1) An *expert system strategy* for a *language understanding goal* might be to build a rule-based system containing rules like "If a person gets excited, they break more grammatical rules than usual." By contrast: (2) A *language understanding strategy* for an *expert system goal* might be to build a restricted-English front end that helps an expert enter and edit rules.

All scientific disciplines adopt a paradigm: a list of the problems that are acceptable and worthwhile to tackle, a list of the methods that can and should be tried, and the standards by which the results are judged. Adopting a paradigm is done for reasons of cognitive economy, but each paradigm is one narrow view. Adding to the confusion, some paradigms in AI have grown up both around the various goals *and* around the various strategies! See Appendix A for a more detailed look into AI goals and strategies.

Finer distinctions can be drawn, involving the *tactical* choices to be made, but this turns out to be misleading. In what way misleading? Tactics that appear to be superficially different may share a common source of power: E.g., predicate calculus and frames both rely on a judicious dividing up of the world. Much of the "scruffies' " recent work on plausible reasoning by heuristic rules overlaps (but with very little shared vocabulary!) the "neats' " recent work on nonmonotonic reasoning and circumscription.

The KP and BH and EH are all *strategic* statements. Each could be prefaced by the phrase "*Whichever of the ultimate goals for AI you are pursuing . . .*". The strategic level is, apparently, the level where one needs to take a stand. This is rarely stated explicitly, and it is rarely taken into account by news media or by conference organizers.

We have an abiding trust in our chosen paradigm—empirical inquiry—in doing science the same way as the early Piaget, Newell, Simon, and Gelernter. The number of states that a brain or a computer can be in is immense; both those numbers are so huge as to be almost unimaginable. Turing's Hypothesis likens them to each other; the only other system we're familiar with with that degree of complexity is Nature itself. Mankind has made progress in studying natural phenomena only after centuries of empirically studying those phenomena; there is no reason to expect intelligence to be exempt. Eventually, many of the natural sciences advanced to the point that theory now often precedes

experiment by years or decades, but we have far to go in AI before reaching that stage.

James Wilkinson was asked in 1974 why *he* was the first to discover the truncation errors of early twentieth century integration methods. After all, Wilkes at Cambridge, and others, had access to equal or better machines at the same time. He replied that at the National Physical Laboratory, the Pilot Ace machine was sitting out, available to all to use and to watch. He was fascinated by the rows of blinking lights, and often stood mesmerized by them while his programs ran. Soon he began to recognize patterns in the lights—patterns where there should *not* have been patterns! By contrast, the Cambridge computer was screened off from its users, who got one-day turnaround on their card decks, but who were denied access to the phenomenon that Wilkinson was allowed to observe.

The point is that while having a theory is essential, it is equally important to examine data and be driven by exceptions and anomalies to revise, criticize, and if necessary reject one's theory. To take one last example: a computer-simulated Newtonian world really would *be* Newtonian; only by hooking up actual telescopes and interferometers and such (or the data from them) can the non-Newtonian nature be perceived.

7. A mandate for AI research: mapping the human memome

AI must somehow get to that stage where—as called for by KP and BH—learning begins to accelerate due to the amount already known. Induction will not be an effective means to get to that stage, unfortunately; we shall have to hand-craft that large "seed" KB one piece at a time. In terms of the graph in Fig. 3, all the programs that have ever been written, including AM and EURISKO, lie so far toward the left edge of the x-axis that the learning rate is more or less zero. Several of the more successful recent additions to the suite of ML techniques can be interpreted as pushes in the direction of adding more knowledge from which to begin the learning.

The graph in Fig. 3 shows learning by induction (DISCOVERY) constantly accelerating: the more one knows, the faster one can discover still more. Once you speak fluently, learning by talking with other people (LANGUAGE) is more efficient than rediscovery, until you cross the frontier of what humanity already knows (the vertical line at $x = F$), at which point there is no one to tell you the next piece of knowledge.

"Learning by discovery" is meant to include not only scientific research (e.g., cancer research), but also the many smaller-scale events in which someone formulates a hypothesis, gathers data to test it, and uses the results to adjust their "theory". That small-scale case can occur in a (good) classroom; or just by driving the same route to work over various different times of the day

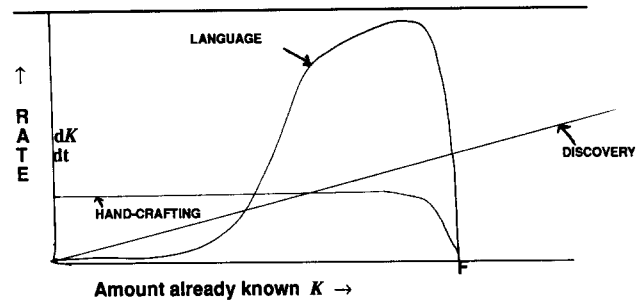

Fig. 3. The rate at which one can learn new knowledge. One can also integrate these three curves with respect to time, to see how the total amount known might grow over time.

(and hypothesizing on rush hour patterns). It involves defining new concepts (at least in principle), formulating new heuristics, and even adjusting or changing one's representation of knowledge. Figure 3 illustrates two more things. Learning by discovery is much *slower* than other forms of learning—such as being told something in natural language—but it is the chief method that extends the boundary F of human knowledge.

By contrast, the rate of hand-coding of knowledge is fairly constant, though it, too, drops to zero once we cross the boundary of what is already known by humanity. The hand-coding rate may slope down a bit, since the time to find related concepts will increase perhaps as the log of the size of the KB. Or, instead, the hand-coding rate may slope *up* a bit, since copy and edit is a powerful technique for knowledge entry, and, as the KB grows, there will be more chance that some very similar concept is already present.

This is an example of EH (the Empirical Inquiry Hypothesis which was presented in Section 1): Only by trying to hand-code the KB will we see which of those two counteracting factors outweighs the other, and by how much. Only by continued work on NL and ML will we determine whether or not there is a region, near where all three curves meet, where ML temporarily surpasses NL as a way to grow the KB. And only much further in the future, after our program crosses the frontier F will we find out if the discovery curve begins to slope up or down.

Figure 3 suggests a sweeping three-stage research program for the coming three decades of AI research:

- Slowly hand-code a large, broad knowledge base.
- When enough knowledge is present, it will be faster to acquire more through reading, assimilating databases, etc.
- To go beyond the frontier of human knowledge, the system will have to rely on learning by discovery, carrying out research and development projects to expand its KB.

Three decades? What are the scales on the axes of Fig. 3? Why do we think it's

not a three-*century* or three-*millenia* program? Even if the vague shapes of the curves are correct, and even if we are near the left edge, how far over to the right is that place where language understanding meets and then surpasses the hand-coding level? Might we need a trillion things in our knowledge base, in order to get analogy and generalization to pay off? The usefulness and timeliness of the Breadth Hypothesis rest on the following quantitative assumption:

Breadth Is within Our Grasp. A KB of about a million "frames" will provide a significant performance increase, due to generalization and analogy; this will consume ~2 person-centuries of time, ~$50 million, and ~1 decade. Why such a "small size"? That's about all that people know!

"Just one million 'frames'! Where did that number come from? What an insult!" you may say. "You just argued that the world is a complicated place. Surely we human beings each know an effectively infinite number of things! It's hopeless to try to represent an appreciable fraction of that, so we may as well settle on writing programs that know only 10–1000 specific things."

What goes on during the 200,000 hours between birth and age 21? Certainly most of it is spent gathering experiences, building up long-term memories; some conscious time (and perhaps some sleep time) is spent generalizing and organizing one's memories. Much of what we're learning is quite specific to ourselves, our home, our family, our friends, and our culture. The result of thrusting someone into a different culture is often tragic or comic; consider, e.g., Crocodile Dundee, A Connecticut Yankee, The Beverly Hillbillies, and The Gods Must Be Crazy.

Three recent estimates of the number of concepts (frames) needed for full breadth of knowledge all came up with a figure of approximately one million:

(1) Alan Kay: 30,000 articles × 30 frames per article.[2]
(2) EDR: 200k words × 1 frame for each of a few languages.[3]
(3) Marvin Minsky: 4 LTM entries/hour from birth to adulthood.[4]

Two other ways for bounding the "bits" a human brain can store lead to much larger numbers: (1) counting neurons and synapses; but it's unclear how memories are stored in them; (2) counting pixels in our "mental images"; but controversy rages in cognitive psychology over whether mental imagery is just an illusion caused by the consistency and regularity in the world that lets us fill in missing pieces of memories—and of dynamic sensory experiences—with

[2] Based on research performed at Atari Research Labs, in conjunction with *Encyclopaedia Britannica*, during 1983.

[3] Reported at the First Workshop on Electronic Dictionaries, Tokyo, November 1988; proceedings available.

[4] Back of the envelope calculation performed for Bob Kahn, at NRI planning meeting, 1985.

default values (see, e.g., [13]). And humans who are blind from birth are not particularly less intelligent for want of those terrabytes of stored mental images. So it's unclear what those larger numbers signify. (Also, though it's clearly an over-simplification, having a million entries means that there can be a trillion one-step inferences involving pairs of them. And it would surprise no one to discover that one-step inference is going on unconsciously in our minds constantly.)

Here again is a situation in which one should apply the EH. Various theories give various estimates, and the way to settle the issue—and, perhaps, much more importantly, achieve the goal of having the KB we want—is to go off and try to build the large KB. Along the way, it will no doubt become clear how big it is growing and what the actual obstacles are that must be overcome.

Lenat started the CYC project in late 1984 for this very purpose. It is now halfway through its ten-year time frame, and, most surprisingly, it is still on schedule. A book describing the project and its philosophy has been published [28], and the interested reader is referred there for details. Here, we shall just very briefly list a few of the surprises that actually trying to build this immense KB has engendered:

(1) The need for more formality, for a more principled representation language. In a typical expert system application, much of the meaning of an entry on a slot of a frame can be idiosyncratic to that particular application; but CYC, which might be used for any application, cannot afford such sloppiness. E.g., consider placing "IceCream" on the "likes" slot of the "Fred" frame. Does this mean that that's all he likes? Does he like all ice cream? In what sense does he like it? Has he liked it from birth onward (and does it mean he'll like it until he dies), or is there some temporal sub-abstraction of Fred that likes it? etc.

(2) The search for a use-neutral control structure and use-neutral representation is not unlike the search for a single universal carpenter's tool. The pragmatic global *effect* of use-neutrality arises by having a large set of tools that complement each other (and sometimes overlap) and easily work together to get most common jobs done. On very, very rare occasions, a new tool may have to get invented; use the existing ones to fabricate it then.

(3) In the case of control structure, CYC has by now amassed two dozen separate inference engines: inheritance, inverse slots, automatic classification, Horn clause rules, transfersThrough, etc. One lesson is that it is cost-effective to write and fine-tune a separate truth (actually, justification) maintenance system (TMS) for each feature, rather than relying on any one general (but of necessity inefficient) TMS algorithm.

(4) In the case of representation, besides frames, we now have numerous other "tools". One of them is a powerful constraint language which is

essentially predicate calculus. This is because much of the knowledge in the system is inherently constraint-like. Consider "The number of children that Joe and Sam have are equal." We could define a new slot sameNumberOfChildrenAs, and such tactics might well get us through any one application, but that's hardly a scalable solution. In general, though, we wanted, needed, and developed a general constraint language. The constraint language is superficially second-order; in almost all real uses, any quantification over predicates (slot names) can be mechanically reduced to first-order. Several dozen of the most common sorts of constraints (e.g., the domain and range of slots) have been "slotized"; i.e., special slots (in this case, makesSenseFor and entryIsA) have been created and optimized, but still the general language is there to fall back on when needed. From time to time, when numerous constraints of the same form have been entered, we "slotize" that form by defining a new slot. For instance, we could create sameNumberOfChildrenAs if there were really heavy use of that sort of constraint.

(5) There are almost ten times as many "frames" required as we had originally expected; luckily, our rate of knowledge entry is also that much faster, so we still hope to "finish" by 1994. In the search for eliminating ambiguity, the knowledge being entered must be more precise than we are used to being in everyday conversation. E.g., the meaning of "Japan" or "water" varies depending on the context of the conversation. Each separate meaning (e.g., political Japan during the 1890s) has its own frame, which is why there are more than we expected. But balancing that, it is relatively easy to build a knowledge entry tool which assists the user in copying and editing an entire cluster of related frames at once. So the two order-of-magnitude increases are not unrelated. By the way, "finishing by 1994" means approaching the crossover point (see Fig. 3), where it will be more cost-effective to continue building CYC's KB by having it read online material, and ask questions about it, than to continue the sort of manual "brain-surgery" approach we are currently employing.

8. Differences with other positions

8.1. Our position regarding the aesthetes

There is a *methodological* difference between our "scruffy" way of doing AI and the aesthetes' "neat" way. This in turn stems from a difference of opinion about whether the world must admit an elegant and simple formalization of intelligence.

If only there were a *secret ingredient* for intelligence—Maxwell's equations

of thought. If only we could axiomatize the world in a small set of axioms, and deduce everything. If only our learning program could start from scratch. If only our neural nets were big or cerebellar or hyperlinear enough. If only the world were like that. But it isn't. The *evidence indicates* that almost all the power is in the bulk knowledge. As Whitehead remarked, "God is in the details."

Following the Difficult Problems Hypothesis, we are firmly convinced that the AI researcher must make a major time commitment to the domain(s) in which his/her programs are to be competent; e.g., the two years that Stefik and Friedland spent to learn about molecular biology before doing MOLGEN [14]; the decade-long time frame for CYC. This is in contrast to, e.g., research that uses the Fifteen Puzzle, or cryptarithmetic, as its domain. Even in physics, where Nature so far *has* been remarkably elegant, it is still strongly cost-effective to expend the enormous time and money to build and use a SLAC or a CERN or a superconducting super-collider.

We may be exhausting the range of potent experimental AI theses that can be carried out in two years, by a student starting more or less from scratch; witness the trend to give the Computers and Thought Award to increasingly less recent graduates. The presence of a large, widely-accessable "testbed" KB should enable a new round of important theses.

People do prefer—and *should* prefer—the simplest consistent hypothesis about any phenomenon. That doesn't make the hypotheses correct, of course [20]. A few examples:

- Early astronomers with poor instruments had no problem with a geocentric model. When one friend of Wittgenstein's ridiculed them for making this error, he replied "Ah, yes, how foolish. But I wonder what it would have looked like if the sun *did* go around the earth?"
- Biologists, who are unable to perform experiments on evolution, or even get precise data on it, can still believe it operates so quickly as it does, using nothing more than random mutation, random generate-and-test—a simple, elegant, appealing, but (as we in AI found out empirically) woefully inadequate problem-solving method.
- James Wilkinson, a fellow of the Royal Society and one of the world's leading numerical analysts, spoke at Stanford in 1974 about bugs in early twentieth century methods of numerical integration. These algorithms had "proofs", and were used (to a couple iterations only) by human beings armed only with pencil and paper. Those people, by the way, were called "computers". The introduction of the high-speed electronic digital computer provided the next round of "criticism"—namely, truncation errors made the algorithms unstable—which led to the next round of improvement in that field.
- Lakatos [21] presents the historical series of mathematicians' retreats from

> the initial form of the Euler–Descartes conjecture to increasingly longer, less elegant versions, with more and more terms getting added to take obscure cases and exceptions into account.

A quarter century ago, Simon provided the spectacular image of an ant on a beach: though its path is complex, that is due to the complexity of the beach, not the ant. It is a beautiful hypothesis that thinking might be based on such simple mechanisms; i.e., that the apparent complexity of the human mind is an illusion, reflecting a very simple, very general problem solver thrust into a complex environment. With apologies to Wittgenstein, and to Newell and Simon, we reply, "Ah, yes, but I wonder what it would look like if the mind *were* filled with—and dependent upon—knowledge?"

Eventually, we *will* want layers of increasing "neatness". E.g., in physics, students learn each year that last year's equations were a special case. We always try to reason at the highest, most superficial, most efficient level at which we can, and delve down one level deeper when we are forced to. But devoting so much effort to the attempt at "neatness" today just drains time and intellectual energy away from the prime enterprise of the field.

Many AI researchers quest for an elegant solution in a desperate desire for scientific respectability. The name of our field—artificial intelligence—invites everyone to instantly form an opinion. Too bad it wasn't called quantum cognodynamics. But perhaps, by interposing a layer of mathematical formalism, we can come to be accepted as hard scientists. Hence the physics-envy!

Formalizing has never driven any early science along. In designing new drug molecules, the biochemist knows it's too inefficient to apply Schrödinger's wave equation to compute the energy minimizations, hence from his/her point of view the fact that such a deep understanding even exists *is irrelevant* to the task at hand. S/he relies on crude design heuristics, and the drug companies using this methodology occasionally are enriched. As Minsky remarked about the A* algorithm in 1970: "Just because it's mathematical doesn't mean it deserves to be taught."

There are actually three separate extreme "aesthete" positions we are arguing against here! In caricature, the first one is the CMU "Soar" (and MIT Robotics) school of letting simple mechanisms give rise to slightly less simple behavior, and then claiming that that will scale up and eventually give rise to human-level intelligence. In that world view, one does not need to assemble a large knowledge base, the machine learner will acquire it automatically. Rod Brooks at MIT goes one step further, denying that one needs even a representation scheme for knowledge [6].

Again in caricature, the second one is the Stanford (and elsewhere) Logic Group school of formalizing some aspect of reasoning (e.g., analogy), programming it to run on a few simple examples, and claiming that that will scale up, that that obviates the need for a huge KB, and besides the important thing is the theorems isn't it? Actually, to be fair, several of these researchers, such

as John McCarthy, Mike Genesereth, and Pat Hayes, have been strong advocates of the need for a large KB eventually; they just believe that there are various tools required to build it which we don't have at present, formal tools which they are researching.

The third, increasingly popular type of aestheticism is the trend to highly parallel (e.g., connectionistic) and ever faster devices. The trouble is that most difficult tasks in knowledge-rich areas can't be highly parallelized. If we were to set a billion people to work trying to find a cure for cancer, we wouldn't find one in 0.2 seconds. Each cancer experiment takes months or years to perform, and there are only a moderate number of promising experiments to do at any one time; their *results* will determine what the next round of promising experiments should be.

Parallelism is useful at one extreme for implementing very carefully engineered algorithms (e.g., systolic algorithms), and at the other extreme for allowing a community of meaningfully-individuated agents act independently, asynchronously. For most technical tasks, until we understand the task very well, the size of such an actor community that we can design is typically only ~100.

The time to perform a task often increases exponentially with its size (e.g., looking ahead n moves in chess). Taking a microcoding approach or a parallelizing approach cuts off a constant factor; taking a knowledge-based approach may add a constant overhead but more importantly, for the long run, it may chip at the *exponent*. On the other hand, it is worth remarking that there are some special tasks where the desired level of performance (x-coordinate) is fixed: just barely beating all humans at chess, just barely understanding spoken words in real time, tracking the space shuttle in real time, etc. In such a case, getting a large enough constant factor speedup really could solve the problem, with no need to apply the KP, BH, or EH. As our ambition to attack ever more difficult problems grows, though, the exponential nature of the search hurts worse.

8.2. Our position regarding expert systems

The KP underlies the current explosion of work on expert systems (ESs). Still, there are additional things our position argues for, that are not yet realized in today's ESs. Knowledge space *in toto* is not a homogeneous solid surface, but more like a set of self-supporting buttes, and one ought to be able to hop from one to its neighbors. But current ESs are too narrow, too independent, and too informal, as we discuss below.

One major power source for ESs, the reason they can be so readily constructed, is the synergistic additivity of many rules. Using a blackboard [11] or partitioned rule sets, it is possible to combine small packets of rules into mega-rules: knowledge sources for one large expert system.

The analogue at the next higher level would be to hook hundreds of large

ESs together, and achieve even greater synergy. That dream repeatedly fails to materialize. Why? As we increase the domain of each "element" we are trying to couple together, the "semantic glue" we need gets to be larger and more sophisticated. The "gluing" or communicating is made all the more difficult by the unstated and often ambiguous semantics that typically exist in a single ES. We discussed, earlier, how the CYC project at MCC has been driven toward increased formality and precision as they have labored to build that large system. It seems to us that it will require the construction of such a system, as mandated by the Breadth Hypothesis, and built not haphazardly but with a clean and formalized semantics, before the true potential of ES technology will be realized.

Plateau-hopping requires breadth

To harness the power of a large number of disparate expert systems will require something approaching full consensus reality—the millions of abstractions, models, facts, rules of thumb, representations, etc., that we all possess and that we assume everyone else does. Moreover, the ESs will need to be coded in a clean, formal representation, and integrated into a global ontology of knowledge.

The INTERNIST program is carefully engineered to do a good job of diagnosing diseases from symptoms. But consider coupling it to a machine learning program, which tries to speculate on new disease mechanisms for epidemiology. The knowledge in INTERNIST isn't stored in "the right way", and much of the needed mechanism knowledge has *already* been compiled away, condensed into numeric correlations. Clancey encountered similar difficulties when he tried to adapt MYCIN's diagnostic KB to *teach* medical students [8].

As we try to combine ESs from various tasks, even somewhat related tasks, their particular simplifications and idiosyncracies prevent synergy. The simplifying was done in the interests of highly efficient and competent problem solving; breadth was not one of the engineering goals.

This naturally results in each ES being a separate, simplified, knowledge universe. When you sit down to build an ES for a task—say scheduling machines on a factory floor—you talk to the experts and find out the compiled knowledge they use, the ways they finesse things. For instance, how do they avoid general reasoning about time and belief? Probably they have a very simple, very specialized data structure that captures just the bits of information about time and belief that they need to solve their task. How do they deal with the fact that this milling machine M has a precise location, relative to all the others; that its base plate is a solid slab of metal of such and such a size and orientation; that its operator is a human; that only one operator at a time can use it; etc.?

If someone accidentally drills a hole through the base plate, most human beings would realize that the machine can still be used for certain jobs but not

for others—e.g., it's okay if you want to mill a very large part, but not a very small one that might fall through the hole! People can fluidly shift to the next more detailed grain size, to reason out the impact of the hole in the base plate, even if they've never thought of it happening before; but the typical ES would have had just one particular level built in to it, so it couldn't adapt to using the crippled milling machine.

Sometimes the ES's precipitous fall into incompetent behavior is obvious, but sometimes its explanations remain dangerously plausible. Meta-rules about the system's area of competence can guard against this accidental misuse, but that is just a patch. A true solution would be to provide a broad KB so that (1) the plateau sloped off gently on all sides, and (2) we could hop from one ES's plateau or butte to another.

This brings up a point which is appropriate both to ESs and to the aesthetes (Section 8.1) as well. Both positions tacitly assume a kind of global consistency in the knowledge base. Inconsistencies may exist for a short period, but they are errors and must be tracked down and corrected. We expect, however, that this is just an idealized, simplified view of what will be required for intelligent systems. Namely, we advocate:

The Local Consistency Hypothesis. There is no need—and probably not even any possibility—of achieving a *global* consistent unification of several expert systems' KBs (or, equivalently, for one very large KB). Large systems need *local consistency*.

The Coherence Hypothesis. Moreover, whenever two large internally consistent chunks C_1, C_2 are similar, their heuristics and analogies should *cohere*; e.g., if the "going up" metaphor usually means "getting better" for C_1, then it should again mean "getting better" for C_2, or else it should not apply at all there.

As regards local consistency, consider how physics advanced for many decades with inconsistent particle and wave models for light. Local consistency is what permits each knowledge-space butte to be independent of the others; as with large office buildings, independent supports should make it easier for the whole structure to weather tremors such as local anomalies. In a locally consistent system, inferring an inconsistency is only slightly more serious than the usual sort of "dead-end" a searcher runs into; the system should be able to back up a bit and continue on. Intelligent behavior derives not from the razor's edge of absolute true versus absolute false—from perfect matching—but rather is suggested by plausibility heuristics and supported by empirical evidence.

Coherence is what keeps one from getting disoriented in stepping from one KB butte to its neighbor. Having the metaphors line up coherently can make the hops so small that one is unaware they have hopped at all: "Her academic

career, her mood, and her prospects were all going up." See [22] for many more examples, and a more detailed discussion of this phenomenon. Coherence applies at the conceptual level, not just at the word level. It is not so much the *words* "going up" as the concept, the *script* of moving upwards, that applies coherently in so many situations.

9. Problems and solutions

Problem 1. *Possible "in-principle" limitations.* There are several extremes that one can point to where the Knowledge Principle and Breadth Hypothesis might be inapplicable or even harmful: perceptual and motor tasks; certain tasks which must be performed in small pieces of real time; tasks that involve things we don't yet know how to represent well (the word "yet" is very important here); tasks for which an adequate algorithm exists; tasks so poorly understood that no one can do it well yet; and (until our proposed large KB becomes a reality) tasks involving large amounts of common sense.

Just as we believe that language faculties will require a large consensual reality KB, we expect it to be invaluable in most of the image understanding process (beyond retina-level edge detection and similar operations).

Our response—in principle and in CYC—is to describe perception, emotion, motion, etc., down to some level of detail that enables the system to understand humans doing those things, and/or to be able to reason simply about them. As discussed under Problem 2, below, we let a large body of examples dictate what sorts of knowledge, and to what depth, are required.

A similar answer applies to all the items which we don't yet know very clearly how to represent. In building CYC, e.g., a large amount of effort in the first five years was spent on capturing an adequate body of knowledge (including representations and problem-solving strategies) for time, space, belief, substances, and so on. We did not set out to do this, the effort was driven completely empirically, completely by need, as we examined snippets of encyclopedia and newspaper articles and had to develop machinery to represent them and answer questions about them. Our response is a tactical hypothesis, not a strategic one; we would find it interesting if it is falsified, but the effect would be negligible on our overall research strategy.

Tasks which can be done without knowledge, or which require some that no one yet possesses, should be shied away from. One does not use a hammer to type with.

The huge KB mandated by the Breadth Hypothesis is AI's "mattress in the road". Knowing that we can go around it one more time, AI researchers build a system in six months that will perform adequately on a narrow version of task X; they don't pause for a decade to pull the mattress away. This research

opportunity is finally being pursued; but until CYC or a similar project succeeds, the knowledge-based approach must shy away from tasks that involve a great deal of wide-ranging common sense or analogy.

The remainder of the problems in this section are primarily pragmatic, engineering problems, dealing with the mechanics of constructing systems and making them more usable. As can be seen from our response to the in-principle limitations, we personally view Problem 1 in that very same category! That is a view based on the EH, of course.

Problem 2. *How exactly do we get the knowledge?* Knowledge must be extracted from people, from databases, from the intelligent systems' KBs themselves (e.g., thinking up new analogies), and from Nature directly. Each source of knowledge requires its own special extraction methods.

In the case of the CYC project, the goal is to capture the full breadth of human knowledge. To drive that acquisition task, Lenat and his team examine pieces of text (chosen from encyclopediae, newspapers, advertisements, and so on), sentence by sentence. They aren't just entering the facts stated, but—much more importantly—are encoding what the writer of that sentence assumed the reader already knew about the world. These are the facts and heuristics and simplified models of the world which one would need in order to understand the sentence, things which would be insulting or confusing for the writer to have actually stated explicitly (e.g., if coke is commercially consumed to turn ore into metal, then coke and ore must both be worth less than metal). They also generalize each of these as much as possible (e.g., the products of commercial processes are more valuable than their inputs). Another useful place they focus is the inter-sentential gap: in a historical article, what actions should the reader infer have happened between each sentence and the next one? Yet another focus: what questions should anyone be able to answer having just read that article? These foci drive the extraction process. Eventually, CYC itself began helping to add knowledge, by proposing analogues, extending existing analogies, and noticing gaps in nearly symmetric structures.

This methodology will collect, e.g., all the facts and heuristics about "Water" that newspaper articles assume their readers already know. This is in contrast to, for instance, naive physics and other approaches that aim to somehow capture a deeper theory of "Water" in all its various forms.

Problem 3. *How do we adequately represent it?* Human experts choose or devise representations that enable the significant features of the problem to remain distinguished, for the relevant connections to be quickly found, etc. Thus, one can reduce this to a special case of Problem 2, and try to elicit appropriate representations from human experts. CYC takes a pragmatic approach: when something proves awkward to represent, add new kinds of

slots to make it compactly representable. In extreme cases, add a whole new representation language to the toolkit. Besides frames and "rules" and our formal constraint language (described above), we use stored images and neural nets as representation schemes. Images are useful for users to point at; e.g., to say something about the strike plate of a door lock—if you don't happen to know what it's called, but you could pick it out instantly given a photo of a door lock. Statistical space partitioning (neural nets) may be useful for certain kinds of user modeling (e.g., gesture level), and the CYC group is currently training one on examples of good analogizing, so as to suggest promising "hunches" of new analogies to investigate, an activity which CYC will then do symbolically.

The quality of the solutions to many of these "Problems", including this one, depend on the quality of our system's emerging ontology. What category boundaries are drawn; what individuals get explicitly represented; what is the vocabulary of predicates (slots) with which to describe and interrelate them, etc.? Much of the 1984–89 work on CYC has been to get an adequate global ontology; i.e., has been worrying about ways to represent knowledge; most of the 1990–94 work will be actually representing knowledge, entering it into CYC. That is why we have "only" a million entries of CYC's KB today, but expect dozens of times that many in 1994.

Problem 4. *How will inference be done in CYC?* The representation chosen will of course impact on what inference methods are easy or difficult to implement. Our inclination was, once again, to apply EH: when we encountered some kind of operation that needed to be performed often, but it was very inefficient, then we adjusted the representation or the inference methods available, or both. As with Problem 3, there is a temptation to early specialization: it is a local optimum, like swerving around a mattress in the road. Pulling this mattress aside means assembling a large repertoire of reasoning methods, and heuristics for choosing, monitoring, and switching among them. When we first prepared this article, such a toolkit of methods was merely an expectation; today (as described earlier), we have two dozen such inference engines, each with its own optimized justification maintenance system (and each capable of running "forward" or "backward").

To illustrate one of those inference methods briefly, consider "transfersThrough". If I tell you that Michael's last name is Douglas, and Michael's father is Kirk, then you infer that Kirk's last name is Douglas. If you see a car with snazzy wire wheels, and I tell you that Fred owns that car, then you infer that Fred owns those wheels. One could represent such inferencing by general if–then rules; for instance, "If x's last name is y, and x's father is z, then guess that z's last name is y." So many such rules were added to CYC, though, that we defined a new inference template (predicate, slot, . . .) called "transfersThrough". There is a frame representing the transfersThrough relationship,

and one of its slots contains a definition in our constraint language, of the form:

```
(ForAll slots s1, s2) ((transfersThrough s1 s2)⇔
  [(ForAll x, y, z) (s1 x y) and (s2 x z)⇒(s1 x z)])
```

We're often asked how we expect to efficiently "index"—find relevant partial matches—as the KB grows larger and larger. The implicit assumption behind that question is that the problem gets worse and worse as the KB size grows. Our answer therefore often appears startling at first glance: wait until our programs *are* finding many, far-flung analogies, but inefficiently, i.e. only through large searches. Then investigate what additional knowledge *people* bring to bear, to eliminate large parts of the search space in those cases. Codify the knowledge so extracted, and add it to the system. This is a combined application of the Difficult Problems Hypothesis and the EH. It is a claim that the true nature of the indexing problem will only become apparent—and solvable—in the context of a large problem running in an already very large KB.

Earlier, we sketched an opportunistic (nonmonolithic) control structure which utilizes items in the control-strategy region of the KB. As with partial matching, we expect that meta-level control mechanism to be more, and more easily, fleshed out as the system grows.

In other words, the large and increasing size of the KB makes certain tasks less difficult, due to having a large and representative sample of cases one ought to try to make efficient. That holds for choosing specialized inference engines, for meta-level control, and for partial matching.

Problem 5. *How can someone interact "naturally" with KB systems?* Knowledge-based systems built so far share with their knowledge-free predecessors an intolerant rigidity of stylistic expression, vocabulary, and concepts. They rarely accept synonyms and pronouns, never metaphors, and only acknowledge users willing to wear a rigid grammatical straitjacket. The coming few years should witness the emergence of systems which begin to overcome this problem. As is only fitting, they will overcome it with knowledge: knowledge of the user, of the system's domain, of discourse, of metaphor. They will employ pictures, gestures, and sound as well as text, as means of input and output. Many individual projects (such as CYC) and expert system tools (such as KEE) are already moving in this direction.

Problem 6. *How can you combine several enterers'/systems' knowledge?* One solution is to sequentialize the entry, but it's not a good solution. Many EMYCIN-based programs designated someone to be the knowledge base czar, with whom all the other experts would discuss the knowledge to be entered.

EURISKO, built on RLL, tried *explicitly* enforced semantics. Each slot was given a description of its intended use, constraints that could be checked statically or dynamically (e.g., each rule's If-maybe-relevant slot should take less CPU time to execute than its If-truly-relevant slot). When someone enters rules that violate that constraint, the system can complain to them, to get everyone back on track using the same semantics again. CYC extends this to *implicitly* enforced semantics: having such a large existing KB that copy and edit is the clearly favorite way of entering new knowledge. When one copies and edits an existing frame, virtually all of its slots' semantics (and even most of their values!) carry right over.

We are not talking about just text-editing here, but rather a problem-solving process in its own right, which CYC should monitor and assist with. Already, CYC makes guesses about which "slots" will exist on the new "frame", which entries on the value will carry over, which will need to be changed, if they're to be changed then will the new entries be idiosyncratic (e.g., monarch) or predictable based on other information about this new frame (e.g., majorExports).

Although this discussion has assumed that inconsistency should be detected and stamped out, there is a much more fundamental long-range solution to the problem of inconsistent KBs: live with them! Problem 7 describes this position:

Problem 7. *How should the system cope with inconsistency?* View the knowledge space, and hence the KB, not as one rigid body, but rather as a set of independently supported buttes. Each butte should be locally consistent, and neighboring buttes should be maximally coherent. These terms are described in Section 8.2. The power of such systems should derive, then, not from perfect matching, but rather from partial matching, heuristic guidance, and (ultimately) confirming empirical evidence. Systems such as we are describing must encompass several points of view; they are "open" in the sense of Hewitt [18]. It should be possible for new knowledge to compatibly and safely flow among them. At a much more exotic level, one can imagine mental immune systems providing (in the background) constant cross-checking, healthy skepticism, advice, and criticism.

Problem 8. *How can the system builder, and the system user, not get lost?* "Getting lost" is probably the right metaphor to extend here, because what they need to do is to successfully navigate their way through knowledge space, to find and/or extend the relevant parts. Many systems, including CYC, are experimenting with various exploration metaphors and orientation tools: helicoptering through semantic nets; exploring a museum with Alician entry into display cases and posters, etc. Both of these are physical spatial metaphors, which allow us to use kinesthetic memory to some extent, as the enterer or user gets more and more familiar with the layout of the KB.

On a typical day in mid-1989, ten to thirty people are logged into CYC's Knowledge Server, all actively adding to its KB simultaneously. Thus, one's world sometimes changes out from under one a bit, adding to the relevance of the (dis)orientation metaphor. For more elaborately scripted interface metaphors, see *True Names* [49], *Riding the Torch* [42], or *Knoesphere* [26]. For instance, the latter suggests clip-on filters to shade or highlight certain aspects of what was seen; models of groups and each individual user; and simulated tour-guides with distinct personalities.

Problem 9. *How big a fraction of "consensus reality" do you need to represent, before the "crossover" occurs and language understanding is a better knowledge entry paradigm?* We believe the answer is around 30–50%. Why? When communicating with an intelligent entity, having chosen some concept X, we would expect the "listener" to be familiar with X; if it fails several times in a row—often!—then it is missing too much of consensus reality. A similar argument applies to analogizing, and to generalizing. Now to have a 30% chance for the chosen analogue to be already known by the listener, he/she/it might have to know 30% of the concepts that are analogized to. But how *uniformly* are good analogues distributed in concept space? Lacking more data, we assume that they are uniformly distributed, which means the system should embody 30% of the full corpus of consensus reality. The distribution is quite possibly *not* uniform, which is why (the EH again) we need to build the KB and see.

10. Conclusion: Beyond local maxima

Our position includes the statements:

- One must include *domain-specific* knowledge to solve difficult problems effectively.
- One must also include both *very general* knowledge (to fall back on) and very *wide-ranging* knowledge (to analogize to), to cope with novel situations.
- We already have plenty of theories about mechanisms of intelligence; we need to proceed empirically: go off and build large testbeds for performing, analogizing, ML, NL, ...
- Despite the progress in learning, language understanding, and other areas of AI, *hand-crafting* is still the fastest way to get the knowledge into the program for at least the next several years.
- With a large KB of facts, heuristics, and methods, the fastest way will, after some years, tip toward NL (reading online textual material), and then eventually toward ML (learning by discovery).

- The hand-crafting and language-based learning phases may each take about one decade, partially overlapping (ending in 1994 and 2001, respectively, although the second stage never quite "ends"), culminating in a system with human-level breadth and depth of knowledge.

Each of those statements is more strongly believed than the one following it. There is overwhelming evidence for the KP and EH. There is strong evidence in favor of the BH. There is a moderate basis for our three-stage program. And there is suggestive evidence that it may be possible to carry out the programs this century.

As a partial application of the Breadth Hypothesis, consider the task of building a knowledge-based system covering most of engineering design. Interestingly, this task was chosen independently by the Japanese EDR project, by Bob Kahn's fledgling National Research Institute, and by Feigenbaum at Stanford. All three see this task as a moderate-term (~1994) goal. It is certainly much broader than any single expert system, yet much narrower than the universal knowledge base mandated by the BH.

Slightly narrower "lawyers' workstations" or "botanists' workstations", etc., are similar sorts of compromises (partial applications of BH) worth working on. They would possess a crown of very general knowledge, plus their specific field's next level of generalities, useful representations, etc., and some detailed knowledge including, e.g., methods for extracting and using entries in that field's online databases. These have the nice side effect of enticing the experts to use them, and then modify them and expand them.

The impact of systems mandated by the KP and BH cannot be overestimated. Public education, e.g., is predicated on the *un*availability of an intelligent, competent tutor for each individual for each hour of their life. AI will change that. Our present entertainment industry is built largely on passive viewing; AI will turn "viewers" into "doers". What will happen to society as the cost of wisdom declines, and society routinely applies the best of what it knows? Will a *knowledge utility* arise, like the electric utility, and how might it (and other AI infrastructures) effect what will be economically affordable for personal use?

When we give talks on expert systems, on commonsense reasoning, or on AI in general, we are often asked about the ethical issues involved, the *mental* "environmental impact" it will have, so to speak, as well as the direct ways it will alter everyday life. We believe that this technology is the analogue of language. We cannot hold AI back any more than primitive man could have suppressed the spread of speaking. It is too powerful a technology for that. Language marks the start of what we think of as civilization; we look back on pre-linguistic cultures as uncivilized, as comprised of intelligent apes but not really human beings yet. Can we even imagine what it was like when people couldn't talk with each other? Minsky recently quipped that a century from

now people might look back on us and wonder "Can you imagine when they used to have libraries where the books didn't talk to each other?" Our distant descendants may look back on the synergistic man–machine systems that emerge from AI, as the natural dividing line between "real human beings" and "animals". We stand, at the end of the 1980s, at the interstice between the first era of intelligent systems (competent, thanks to the KP, but quite brittle and incombinable) and the second era, the era in which the Breadth Hypothesis will finally come into play.

Man–Machine Synergy Prediction. In that "second era" of knowledge systems, the "system" will be reconceptualized as a kind of colleagular relationship between intelligent computer agents and intelligent people. Each will perform the tasks that he/she/it does best, and the intelligence of the system will be an *emergent* of the collaboration.

The interaction may be sufficiently seamless and natural that it will hardly matter to anyone which skills, which knowledge, and which ideas resided where (in the head of the person or the knowledge structures of the computer). It would be inaccurate to identify Intelligence, then, as being "in the program". From such man–machine systems will emerge intelligence and competence surpassing the unaided human's. Beyond that threshold, in turn, lie wonders which we (as unaided humans) literally cannot today imagine.

Appendix A. Goals and strategies for AI research

In Section 6, we briefly touched on the common confusion between AI goals and AI strategies. The next two sections list nine of each. As we mentioned in the body of the paper, much confusion in our field stems from several entries appearing on both lists. If one researcher chooses, say, the ultimate goal of language understanding, then they could approach that strategically in several ways. E.g., humans first learn language by discovery, by imitating others' sounds, noting correlations and inducing simple vocabulary and grammar rules. Later, as we enter school, we further improve our language abilities by taking English classes, i.e., by discussing in natural language the fine points of English vocabulary and grammar and composition.

Scientific disciplines not only adopt a paradigm, in the early stages they are partitioned into subfields by paradigm. If more than one paradigm remains viable for any length of time, it will soon come to see itself as a different discipline altogether and split off; AI faced this around 1970 with cognitive psychology, and is facing this again now with robotics and vision. People can't mentally focus on too much at once, and paradigms provide some of the obligatory cognitive simplifying.

All 9×9 pairs of the form $\langle$goal, strategy$\rangle$ could in principle be separate paradigms. Today there are only a small fraction of that number, and the groupings that have developed are in many cases poorly matched; e.g., all pairs of the form $\langle x, \text{Learning}\rangle$ unioned with $\langle \text{Learning}, x\rangle$, come together for machine learning workshops every year. This leads to confusion and miscommunication—often unrealized by both parties! Let's give an illustration of this phenomenon, still in the domain of machine learning. When they say they are working on analogy, they might mean either one of the following:

(1) They are using analogy as a strategy to pursue some other AI goal G. For instance, they might be building a program whose goal is to parse or disambiguate English sentences by analogy.
(2) They are using some other strategy S, such as knowledge engineering, as the power source in a program whose task is to discover and flesh out analogies. In that case their program's final output would be a data structure that humans somehow recognize as symbolizing an analogy; but that data structure might be built by a set of if–then rules, or a neutral net, or by talking with a human being, etc.

The trouble is that, today, they are equally likely to mean they're pursuing analogy as a strategy or as a goal.

A.1. Nine ultimate goals of AI

We share, or are sympathetic to, almost all of these:

- *Understand human cognition.* The goal is to understand how people think, not to have machine artifacts to put to work. Try for a deeper knowledge of human memory, problem-solving abilities, learning, decision making in general, etc.
- *Cost-effective automation.* The goal is to replace humans at various tasks requiring intelligence. This goal is met by programs that perform as well as the humans currently on the job; it doesn't matter if the programs think like people or not. The harder the problems it can solve, and the faster it solves them, the smarter it is.
- *Cost-effective intelligence amplification.* The goal is to build mental prostheses that help us think better, faster, deeper, more clearly, ... Science's goal—and measure of success—is how much it augments human being's leg muscles, immune system, vocal cords, and (in this case) brain. This goal further divides depending on whose performance is being so amplified: do we want to amplify the average person's ability to diagnose disease, or the average GP's ability, or the world's best diagnostician's?
- *Superhuman intelligence.* The goal is to build programs which exceed human performance. Crossing that particular threshold could lead to an explosion of progress: technological innovation in manufacturing, theoreti-

cal breakthroughs, superhuman teachers and researchers (including AI researchers), and so on.

- *General problem solving.* Be able to solve—or at least plausibly attack—a broad range of problems, including some from fields you've never even heard of before. It doesn't matter if the programs perfectly fit human performance data, nor does it matter if they are at an expert's level. The point is that intelligent creatures can get somewhere on almost any problem; intelligence is flexibility and breadth of mind, not depth in some narrow area.
- *Coherent discourse.* This is similar to the Turing test. The goal is to competently communicate with people, using complete sentences in some natural human language. A system is intelligent iff it can carry on a coherent dialogue.
- *Autonomy.* This goal holds that a system is intelligent iff it can, on its own initiative, do things in the real world. This is to be contrasted with, say, merely planning in some abstract space, or "performing" in a simulated world, or advising a human who then goes off and does things. The idea is that the real world is always so much more complex than our models of it, that it is the only fair test of the programs we claim to be intelligent.
- *Learning* (*induction*). This goal is to have a program that chooses what data to gather and how; gathers it; generalizes (or otherwise converts) its experiences into useful new beliefs, methods, heuristics and representations; and reasons analogically.
- *Information.* Having stored lots of facts about a wide range of topics. This is more of a "straw man" view than the others, as it could be satisfied by an online textual encyclopedia, or even by a hardcopy one! The other views all require the intelligent entity not merely possess information but also use it appropriately.

A.2. Broad strategies for achieving those goals

Most of these are not *our* strategies:

- *Duplicate low-level cognitive performance.* Get your program to duplicate even micro-level measurements that psychologists have gathered from human subjects, such as memory storage and recall times, STM size, forgetting rate, errors, etc. Hopefully, if you do that, then your program's internal mechanisms will be similar to humans', and your program will be able to scale up the same way that human low-level mechanisms scale up (even though we don't know how that is, we won't have to know if we get the lowest level built the same way). One variation is to use slightly less low-level mechanisms (such as Soar's chunking), but still the idea is that repeated application of simple IPS processes let intelligence emerge.

- *Duplicate low-level structure.* Mimicking the human brain's architecture will lead to mimicking its functionality. This strategy traditionally makes the further assumption that McCulloch–Pitts threshold logic is the right level at which to abstract brain cell structure (rather than, e.g., at the chemical and enzymatic levels). It gained attention as perceptrons, and now enjoys a renaissance due to the promise that VLSI technology holds for soon producing parallel neural nets of immense size.
- *Simulate a society of mind.* This is yet another variant on the "duplicate and hope" strategy, but this one is not so low-level as either of the previous two strategies. Build a program that consists of hundreds of specialized mental beings or actors—think of them as kludged knowledge sources—and marshall them to solve problems by cooperating and communicating among themselves. This is how Nature managed to evolve us, and it may be the easiest way for us to in turn evolve AI.
- *Knowledge engineering.* Talk with human experts who perform the task, and extract from them the facts, representations, methods, and rules of thumb that they employ in doing the task. Encode these in a running prototype system, and then extract more and more knowledge, as the program runs and makes mistakes which the expert can easily translate—in context—into additional pieces of knowledge that should have been in the system all along. Have faith that this incremental knowledge acquisition will attain an adequate level of competence.
- *Natural language understanding.* Have a program talk with people, read articles, etc. People achieve intelligence that way; so can machines!
- *Learning (induction).* Build a program that can learn. Then let it. People get to be smart by learning, starting from a tabula rasa; so can machines.
- *Formalizing and advanced reasoning.* Marshall a toolkit of sophisticated deductive procedures for maintaining consistency and inferring new assertions. Having such a set of snazzy mechanisms will be necessary and sufficient. The strong version of this view says "It worked for physics; we must strive to find the 'Maxwell's equations of thought'." The mild version is more conservative: "As you formalize, you find the gaps in your understanding."
- *Intelligence amplification.* Build some intelligent interfaces that allow us to write programs more easily, or synthesize ideas more rapidly, etc. Then let these improved man–machine systems loose on the problem of achieving AI, whichever goal we choose to define it. In other words, instead of tackling the AI task right away, let's spend time getting prostheses that let us be smarter, then we'll come back to working on "real" AI.
- *Superhuman intelligence.* An extreme form of the previous strategy. Build a program that does AI research just slightly better than people do, and then go sit on a beach while it investigates low-level cognition, or language understanding, or whatever your chosen AI goal is.

References

[1] G. Abrett and M.H. Burstein, The KREME knowledge editing environment, in: *Proceedings Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Banff, Alta. (1986).

[2] S. Amarel, On representations and modelling in problem solving and on future directions for intelligent systems, RCA Labs Scientific Rept. No. 2, Princeton, NJ (1967).

[3] W.W. Bledsoe, Non-resolution theorem proving, *Artif. Intell.* **9** (1977) 1–35.

[4] W.W. Bledsoe, I had a dream: AAAI Presidential Address, *AI Mag.* **7** (1) (1986) 57–61.

[5] A. Borning and S. Weyer, A prototype electronic encyclopedia, *ACM Trans. Off. Inf. Syst.* **3** (1) (1985) 63–88.

[6] R.A. Brooks, Intelligence without representation, *Artif. Intell.* **47** (1991) 139–159, this volume.

[7] B.G. Buchanan, G. Sutherland and E.A. Feigenbaum, Heuristic Dendral: a program for generating explanatory hypotheses in organic chemistry; in: B. Meltzer and D. Michie, eds., *Machine Intelligence* **4** (American Elsevier, New York, 1969) 209–254.

[8] W.J. Clancey, Dialogue management for rule-based tutorials, in: *Proceedings IJCAI-79*, Tokyo (1979) 155–161.

[9] R. Davis and D.B. Lenat, *Knowledge Based Systems in Artificial Intelligence* (McGraw-Hill, New York, 1982).

[10] EDR (Japan Electronic Dictionary Research Institute, LTD), Tokyo, Japan, Personal communication (1987).

[11] L. Erman, F. Hayes-Roth, V. Lesser and D.R. Reddy, Hearsay-II speech understanding system, *Comput. Surv.* **12** (1980) 224–225.

[12] E.A. Feigenbaum, The art of artificial intelligence: themes and case studies in knowledge engineering, in: *Proceedings IJCAI-77*, Cambridge, MA (1977).

[13] J.A. Fodor and Z.W. Pylyshyn, How direct is visual perception?, *Cognition* **9** (1981) 139–196.

[14] P. Friedland, Knowledge-based experiment design in molecular genetics, in: *Proceedings IJCAI-79*, Tokyo (1979) 285–287.

[15] H. Gelernter, J.R. Hansen and D.W. Loveland, Empirical exploration of the geometry theorem machine, in: *Proceedings Western Joint Computer Conference* (1960) 143–147.

[16] M.R. Genesereth, The use of design descriptions in automated diagnosis, *Artif. Intell.* **24** (1984) 411–436.

[17] M.R. Genesereth and N.J. Nilsson, *Logical Foundations of Artificial Intelligence* (Morgan Kaufmann, Los Altos, CA, 1987).

[18] C. Hewitt, Open systems, AI Memo 691, MIT, Cambridge, MA (1982).

[19] Japan Electronic Dictionary Research Institute, Concept dictionary, Tech. Rept. TR-009, EDR, Tokyo (1988).

[20] T.S. Kuhn, *The Structure of Scientific Revolutions* (University of Chicago Press, Chicago, IL, 2nd ed., 1970).

[21] I. Lakatos, *Proofs and Refutations* (Cambridge University Press, Cambridge, 1976).

[22] G. Lakoff and M. Johnson, *Metaphors We Live By* (University of Chicago Press, Chicago, IL, 1980).

[23] P. Langley, J. Zytkow, H.A. Simon and G. Bradshaw, The search for regularity: four aspects of scientific discovery, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell, eds., *Machine Learning* **2** (Tioga, Palo Alto, CA, 1985).

[24] D.B. Lenat, Beings: Knowledge as interacting experts, in: *Proceedings of IJCAI-75*, Tblisi, USSR (1975).

[25] D.B. Lenat, The nature of heuristics, *Artif. Intell.* **19** (1982) 189–249.

[26] D.B. Lenat, A. Borning, D. McDonald, C. Taylor and S. Weyer, Knoesphere: expert systems with encyclopedic knowledge, in: *Proceedings IJCAI-83*, Karlsruhe, FRG (1983).

[27] D.B. Lenat and J.S. Brown, Why AM and EURISKO appear to work, *Artif. Intell.* **23** (1983) 269–294.

[28] D.B. Lenat and R.V. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project* (Addison-Wesley, Reading, MA, 1990); portions also available as: The world according to CYC, MCC Tech. Rept. ACA-AI-300-88, Austin, TX (1988).

[29] S. Marcus, J. McDermott and T. Wang, Knowledge acquisition for constructive systems, in: *Proceedings IJCAI-85*, Los Angeles, CA (1985).
[30] J. McCarthy, Programs with common sense, in: *Proceedings Symposium on Mechanisation of Thought Processes*, Teddington, England (H.M. Stationery Office, London, 1959); reprinted in: M. Minsky, ed., *Semantic Information Processing* (MIT Press, Cambridge, MA, 1968).
[31] J. McCarthy, We need better standards for AI research, *AI Mag.* **5** (3) (1984) 7–8.
[32] J. McDermott, R1: an expert in the computer systems domain, in: *Proceedings AAAI-80*, Stanford, CA (1980) 269–271.
[33] M. Minsky, *Society of Mind* (Simon and Schuster, New York, 1985).
[34] J. Mostow, Searching for operational concept descriptions in BAR, MetaLEX, and EBG, in: *Proceedings Fourth International Workshop on Machine Learning*, Irvine, CA (1987) 376–382.
[35] A. Newell, The knowledge level, *AI Mag.* **1** (1) (1980).
[36] A. Newell and J.C. Shaw, Programming the logic theory machine, in: *Proceedings Western Joint Computer Conference* (1957) 230–240.
[37] A. Newell and H.A. Simon, *Human Problem Solving* (Prentice-Hall, Englewood Cliffs, NJ, 1972).
[38] P. Norvig, Inference in text understanding, in: *Proceedings AAAI-87*, Seattle, WA (1987) 561–565.
[39] G. Polya, *How to Solve It* (Princeton University Press, Princeton, NJ, 1957).
[40] H. Pople, Heuristic methods for imposing structure on ill-structured problems: the structuring of medical diagnosis, in: P. Szolovitz, ed., *Artificial Intelligence in Medicine* (Westview Press, Boulder, CO, 1982) 119–190.
[41] B. Russell, *Human Knowledge: Its Scope and Limitations* (Unwin, 1948).
[42] N. Spinrad, *Riding the Torch* (Bluejay Books, New York, 1984).
[43] M. Stefik, The knowledge medium, *AI Mag.* **7** (1) (1986) 34–46.
[44] M. Stickel, A non-clausal connection graph resolution theorem-proving program, in: *Proceedings AAAI-82*, Pittsburgh, PA (1982) 229–233.
[45] W. Strunk and E.B. White, *The Elements of Style* (Macmillan, New York, 3rd ed., 1979).
[46] D. Subramanian, A theory of justified reformulations, Ph.D. Thesis, Computer Science Department, Stanford University, Stanford, CA (1988).
[47] G.J. Sussman, *A Computer Model of Skill Acquisition* (American Elsevier, New York, 1975).
[48] A. Tversky, P. Slovic and D. Kahneman, eds., *Judgment under Uncertainty: Heuristics and Biases* (Cambridge University Press, Cambridge, 1982).
[49] V. Vinge, *True Names* (Bluejay, New York, 1984).
[50] What's News, *Wall Street J.* **LXXIX** (65) (April 7, 1987) 1.

Reply to Brian Smith*

We have chosen to write a separate response to Brian Cantwell Smith's review of our position paper, rather than to respond by making changes and additions to the text of our article. Corresponding to each of Cantwell Smith's five sections, we highlight the issues upon which he and we appear to disagree; make a judgment about whether this is (a) just a misunderstanding, or (b) a genuine difference of opinion; and attempt a clarification.

1. Introduction

1.a. Mistakenly attributed beliefs[1]

In his long footnote 2, Smith accuses us of "an astounding reversal" in the last four years, a change in goals and methods from "coding up everything in the encyclopedia" to "the complement of the encyclopedia". All this reflects is his finally listening to what we've been saying for the past six years (since before the CYC project began). E.g., as we wrote in an article published in January 1986:

> "General knowledge" can be broken down into a few types. First, there is real world factual knowledge, the sort found in an encyclopedia. Second, there is common sense, the sort of knowledge that an encyclopedia would assume the reader know without being told (e.g., an object can't be in two places at once). [13]

And, as we went on to state there—and in all our talks and articles since 1984—looking at articles, advertisements, snippets of conversations, etc., is useful as a *tool* to drive human knowledge enterers to introspect concretely on that unstated underlying commonsense knowledge. For example, CYC knowledge enterers often make use of absurd supermarket tabloid headlines and articles, not because we want CYC to believe those things (!), but rather because they provide a natural introspective ice-breaker, namely asking oneself "Now why do I find that headline hard to believe?" Indeed, in a much earlier

* See article by B.C. Smith on pp. 251–288 of this volume.

[1] As just explained, this section—1.a—is a collection of remarks to "set the record straight" in cases where we feel that Section 1 of Smith has misrepresented or misunderstood our position. We regret the necessity to include any of this "a" type material, and endeavor to be as brief as possible in the "a" subsections.

article written about Alan Kay's Knoesphere project (which in some ways was the precursor to CYC), our position was already clear:

> Something which is absent for a typical encyclopedia but must be present in the Knoesphere KB is commonsense knowledge. This includes everyday physics, models of human interaction . . . as well as facts and heuristics about teaching, question-answering, imagery, analogy, etc. We intend to spend much of the coming decade in research trying to build such a core [of everyday concepts] [10].

In that same long footnote, Smith accuses us of being "considerably more optimistic" four years ago than today, about the chances for success in getting something like CYC to succeed. Although we don't discuss such things in our article, quite the reverse trend has occurred. When CYC began, back in late 1984, we estimated it had a low (less than one in ten) chance of succeeding. Year by year, our optimism has grown; we now put its chances at better than 50–50. Yes, we spent most of the early years in thrashing out a representation language and ontology, and now we're spending most of the effort using that (rather than fighting it) to do knowledge entry. Smith interprets that as a negative indicator, but we interpret it as an extremely positive and encouraging pattern. The time for pessimism—or perseverence—was 5 years ago, not today. We chose perseverence, and it has paid off.

In that same footnote, Smith claims our estimate of the size of the required KB has increased over the years. That's true (it correlates with our decreasing naivete), but even back in 1983 [10] our estimate was 300k frames for factual knowledge, and a similar volume for commonsense knowledge. Assuming about 100 individual assertions per concept, that number (600k frames, 60 million assertions) is not so far away from our best guess today (1 million frames, 100 million assertions). Perhaps some of the confusion came from mixing the two units of measure: concepts (frames) and individual assertions.

Again in that footnote, Smith confuses the pragmatic necessity of having several inference engines (to do efficient justification maintenance) with the theoretical "plank" of ours which continues to state that sophisticated inference procedures alone won't solve all your problems (literally and figuratively) if you lack knowledge.

We're almost ready to leave Smith's footnote 2. Smith implies, near the end of it, that we believe that all the theoretical foundations of AI will be complete by 1994. We certainly do not believe that. In fact, a host of fundamental research questions may be uncovered by this work, and become seen as important. E.g., one can successfully build a bridge over a stream without much theoretical understanding of engineering and physics, and the enterprise of doing so is quite likely to reveal many new issues to begin to investigate, issues that eventually lead to the development of a theory. The same situation occurs when high energy experimental physicists gather data about collisions at new energy levels, etc., etc. It's foolish never to theorize, but it's commonplace

for empirical experiments and constructions to outstrip (and drive) the development of theory, especially in a field's first few centuries of life. In its early stages, a theory may be little more than a plausible generalization of a class of recently observed phenomena. Theory building must—and does—go on in the absence of complete sets of data to characterize; and experiments must—and do—go on in the absence of complete theories.

Finally, we can leave footnote 2! A few lines later, Smith mistakenly attributes to us the absurd claim that "just a million frames . . . could intelligently manifest the sum total of human knowledge". That is most definitely not what we believe, or claim, or hope. Rather, our hope is that that order of magnitude (i.e., about 1 million "frame-fulls" or about 100 million assertions) will suffice for crossing the point where knowledge acquisition could be more profitably done by natural language understanding (reading online texts and "discussing" the difficult parts) rather than continuing to build the KB manually, one assertion at a time. We may span much of the breadth of human knowledge, but of course not the depth—one of the main uses of such a KB will be as a substrate on which to build the next generation of knowledge-based systems which do go into depth in particular areas.

Moreover, we certainly don't restrict ourselves to frames, though the majority of the assertions can be cast as simple $P(x, y)$ statements. Our philosophy is to not flinch from building special-purpose machinery (for representation, for control, for interfacing, etc.) to handle the most commonly occurring cases, and to thus have a series of increasingly general (and inefficient) mechanisms even though most of the very general ones are rarely used.

In the case of representation, we use frames for most of the assertions in the KB, but we of course have to have a way to represent disjunctions, set-theoretic constraints, quantified statements, etc., and so we have a constraint language (similar to predicate calculus with equality) as well.

For most human users browsing through the KB and editing it, it's proven useful to present assertions $P(u, v, \ldots)$ which share a common first argument u clumped together—i.e., to have what appears to be a frame-based interface. There are by now half a dozen different editing and browsing tools, some of them quite un-framelike (built around semantic nets, or a metaphor to a museum floorplan, or predicate calculus). While many humans prefer frame-like anchorings, the most common interface to/from other (non-CYC-based) application programs has been straight constraint language expressions. Although this is not the place to discuss "standards" for knowledge representation, we expect that whatever interlingua develops and becomes adopted will likely be based around something which is similar to that. See, e.g., Genesereth's proposal for KIF [1].

In the case of control structure, we again see a series of increasingly general (and inefficient) rules of inference—inference engines—and once again the most specific ones are the most efficient and the most frequently used. E.g., we could express the fact that "children(x, y) iff parents(y, x)" using general

if–then rules, Horn clause rules, . . . , all the way down to the special-purpose mechanism "inverse" (i.e., by asserting inverse(children, parents)). The latter is not just shorter to state, it is much faster to "run" (e.g., to later retract, to not show up later in irrelevant situations while searching for a proof, etc.) because the maintenance of inverse(r, s) assertions has been thoroughly worked out ahead of time in CYC.

1.b. Genuine disagreements[2]

We do believe that clever control structures *alone* are no substitute for large amounts of cached knowledge. Of course some amount of effort—perhaps as high as 20% of the two person-centuries of effort in getting CYC to its mid-1990's "crossover point" must deal with inference—with symbol manipulating methods to do deduction and induction (including abduction, analogy, and so on).

We do believe that a decade of flat-out work will get us through "stage 1" in our research program. Smith is welcome to begin his "decades of debate"; meanwhile, we are happy to announce that CYC is halfway through its one decade lifetime and still on schedule. Yes, of course there is "a middle realm", Brian, but it is immense. A brief paper can do little more than tantalize, and we encourage the reader to go through [12] for a several hundred page foray into that middle realm.

2. Conceptual tunneling

2.a. Mistakenly attributed beliefs

We are as aware as anyone of the range of application of expert system technology today—and its limitations. We have written about that in numerous books and articles over the past fifteen years. An expert system can't be a nurse because of the heavy reliance on sight, hearing, etc., the need for frequent and subtle motor activity and hand–eye coordination, the need to provide inter-personal warmth and support, etc. Could an expert system in principle one day write a good textbook on nursing, or design a new device used in nursing? To that we would answer affirmatively, and astonishingly enough so would Smith. We are pleased in a way that Smith views our general principles as tautologous. Not so many years ago, and still today in many academic circles, they would be quite controversial.

His attributing a trivial Analogical Method to us is a bit unfair. Immediately

[2] As explained earlier, this section—1.b—discusses the genuine disagreements that we have with Section 1 of Smith, answers some of his objections to our position, and presents our disagreements with his position as stated therein.

after we state it, we explain why it's too general and weak to be useful, and how it can and should be specialized. We have a useful specification of the notion of causality, a family of "causes" relations, and a calculus for using them to predict (deductively), to explain (abductively), and to help us epistemologically to favor one explanation over another.

Yes, we know "these issues have been investigated for years". We've been doing some of the investigating! He accuses us of not being aware that "analogy requires a notion of relevant similarity", which is odd given the detail with which we discussed that in [9]. And as the previous paragraph indicated, we've proceeded a long way further than that in the intervening years.

2.b. Genuine disagreements

The need for formulation: Simple problems can of course be solved without explicitly formulating them—else the meta-meta-$\cdots$-level recursion would never end. But difficult ones require formulation. You may grasp your coffee mug unconsciously, but you probably don't design an airplane that way. At least not a plane I'd care to be the first to fly in. Raw perception and low-level muscle coordination are not part of what we were calling knowledge, though of course propositions about what you have perceived, about the act of your perceiving it, etc., are knowledge.

This ties in to his remark about children not having explicit mental models (formulations, representations). Two remarks are in order here. First, although Smith (and the colleagues he cites) tell us we don't need such things, they don't propose any alternative. Second, we don't care whether that's "really" how people solve problems and get around in the real world—we're AI scientists, not cognitive psychologists. And we feel that, in limited domains, the best computational scheme to get programs to duplicate human-level problem-solving behavior is through explicit formulation (and logical manipulation of same).

How dare we try to build this KB: We are not so pessimistic (or perhaps so perfectionistic) as Smith. In our opinion, AI has progressed to the point where it's worth trying to build the large, broad KB: we do know ways to adequately represent a vast variety of knowledge, we do know enough about ontology and ontological engineering to choose and debug an adequate set of collections, predicates (slots), and so on. If we fail, then the next set of important lessons for AI are likely to emerge by tackling this large empirical task, rather than by micro-experiments or sterile philosophical argument.

Are we sure about the one million frame number (100 million assertions)? Of course not. But we have lots of supportive evidence, based not just on the three estimates we gave in our article, but also based on the thousands of man-years of effort spent on building knowledge-based systems in the past fifteen years. Yes, by now we feel we do have the right to estimate how many "frames" it will take.

Smith chooses visual imagery as his examples (recognizing faces; the expression on a person's face gradually changing)—even though we explicitly claimed we would not tackle perception head-on. On the other hand, knowledge *about* facial expressions can and should be part of the large KB, and it is easy to see that a small number of assertions suffices to predict the change in expression when one goes from glee to horror, and a moderate number of assertions to predict the change in emotional state as one loses control of one's vehicle. Yes, one can sit around for decades and bemoan the impenetrable mystique of the human intellect, and make grand arguments for why it is unknowable, or one can sit down and try to penetrate it.

The heart of this disagreement is made clear over the issue of whether we in AI should use computers to test (i.e., verify or falsify) our hypotheses and surprise us (our view), versus merely as fancy word processors to articulate and clarify one's hypothesis (his view).

As Guha[3] reminds us,

> One of the things we have learnt from so many years of science is that given any hard problem it is wise to break it down into separate pieces, solve these, and put the solutions together. But for Brian Smith, perception, inference, representation are all one complex integrated (nondecomposable) problem that has to be dealt with at one shot. The idea that he (or anyone else) can do this seems rather optimistic. Divide and conquer is really a pretty good idea.

It's a bit much for Smith to presume that he knows what discouraged Winograd and we don't; but granting for the sake of argument it was the problem of "genuine semantics", we claim that this problem gets easier, not harder, as the KB grows. In the case of an enormous KB, such as CYC's, for example, we could rename all the frames and predicates as G001, G002, . . . , and—using our knowledge of the world—reconstruct what each of their names must be. While this does not guarantee that the genuine meanings of the concepts have been captured, it's good enough for us. After all, how does one guarantee that one's neighbor shares the same meanings for terms? The answer is that one doesn't, at least not formally or exhaustively. Rather, in practice, one defeasably assumes by default that everyone agrees, but one keeps in reserve the ubiquitous conflict resolution method that says "one may call into question whether they and their neighbor are simply disagreeing over the meaning of some terms".

Near the end of Section 2, Smith raises the question of how a set of symbols relates to the world. He, better than most people, knows how many person-centuries have been lost on this issue. Programmers were writing working programs long before people developed fancy semantics for programming

[3] Personal communication.

languages. Surely he drives a car and uses other devices without "really knowing" how they work. In a similar fashion, we hope to use symbol structures to represent things without "really knowing" the answer to this question.

There are two issues, then, as regards our disagreement with Smith about how "solved" the problems are about deduction and control. The first issue is how to—and whether to—come up with a formalism in which to state relevant information. The second issue is how to actually use these formalisms to state the axioms. We are claiming that AI has found at least reasonable candidates for the former, and that it's finally time to really start doing the latter. Another way to look at what we are saying is that at the current state of the field, the maximum gain/improvement can be obtained by building KBs. Not that we are going to have a fully human-level intelligent agent in 1994, but that better AI—a whole new and qualitatively different set of experiments—can be done in 1995, using as a substrate the large KBs constructed between now and then.

3. The structure of the middle realm

3.a. Mistakenly attributed beliefs

Question 1: Explicit representation

Smith seems to massively misunderstand what we meant by "explicit". E.g., he says "L&F are even more committed to explicit representation than adherents of logic", which to us is a non-sequitor. What we mean by "explicit" is a representation with a declarative semantics. Thus, we might say a program P represents X even if there is no data structure in P that means exactly X, so long as X follows from other data structures (where "follows from" is given shape by the declarative semantics).

We are pragmatists and engineers; tradesmen, not philosophers. We are happy to use any tool that helps us in some specialized ways, and that includes implicitly represented knowledge. Despite myriad examples of such (which Smith mentions in his footnote 13) that we use in our programs, the evidence still supports our expectation that the vast majority of the contents of CYC *will* be declarative, and we view the gradual translation of knowledge into increasingly declarative form as both inevitable and desirable. (There is also often an extra level of translation, from declarative into efficient "compiled" form, and this final stage is also probably inevitable and futile.)

Question 5: Multiple representations of knowledge

We are of course not arguing that any one narrow representation (such as frames and slots) is enough; see our earlier comments about our formal constraint language, etc. Nevertheless, it's important to remark that binary predicates (slots on frames) are—surprisingly often—quite adequate, and such awkwardnesses as arise in trying to represent some new situation *can* often be

remedied just by slotizing (creating specialized new slots). E.g., there are now about 50 slotized forms of constraints on slots—what sort of frames can legally have them, what sort of entries can legally fill them, how many entries they can have, and so on—and we almost never have to resort to writing a full-fledged constraint language expression. Indeed, most of those slotized constraints were introduced (as new slots) to eliminate the need for the various constraint language expressions we had to write. Smith might say that all this is still just one "grammar" for representation—that the frames and slots are a special case of the constraint language. In that case, we count this as a genuine disagreement rather than a mistakenly attributed belief.

Question 7

We could not understand Smith's words here (his use of "traditional" early on in this discussion, his use of "non-representational experience" late in the discussion, his "LISP" example about (LENGTH '(A B C)) which seems to be just a discourse example, etc.). So we're not sure if he misunderstood us or genuinely disagrees.

Resource-limited computation is an important part of our systems' design (e.g., the different GET levels in CYC), and we also rely on explicit meta-level reasoning about strategies, progress being made, time of day, models of the particular people using the system at present, etc.

Question 8

Smith cites Rosenschein's system as an example of "moving beyond logic's familiar representational assumptions". Our understanding of that system, though, is that it is built solidly on modal logic. It illustrates our assumption that one can reason adequately, using propositions, even about phenomena which people intuitively feel are somehow gestalt, mysterious, non-decomposable.

Question 9

A 200% incorrect misreading by Smith. Of course inference is important—it lets you just represent $\log n$ of the KB you'd otherwise have to represent if you tried to cache everything (n is the average depth of reasoning chain your system goes through). As mentioned in our article, CYC has dozens of specialized inference procedures, not just one (as do most AI programs) or zero (as Smith seems to think it does). Our point was not to advocate a 100-million-assertion KB in lieu of a small one plus some inference method—rather, it's to advocate a 100-million-assertion KB plus dozens of inference methods as being just barely enough to get us from stages 1 to 2 in our three-stage research program. It does appear, by the way, that most commonsense inference is rather shallow—2–6 "rule firings" deep—shallow compared to, say, playing master-level chess or proving a difficult theorem. Still, $n = 3$ means $\log^3$(total assertions) = 100 million. So it would be absurd to even

consider an equivalent $n = 0$ KB (it would have to have $10^{10^{10^{10^8}}}$ assertions in it!)

Why did we say 200% instead of 100%? Because he is also wrong in saying that reasoning is central to the logicist position. Some mathematical logicians might say it is, but most computer science logicians would say that knowledge is (see, e.g., McCarthy's Missouri Program; Pat Hayes' Second Naive Physics Manifesto [7]; and so on).

Question 10: Does meaning bottom out?

There are at least two senses in which we shout a negative answer; and one way in which we murmur an affirmative one. First, we have a very strong belief in the "gray box" view of knowledge and—hopefully—our large KB. "Gray box" means that one typically treats the thing as black box, as primitive, but when confronted by some novel problem, or the need to analogize, etc., one can open the black box and examine, modify, etc., the substructure that comprises it. A simple example of "gray boxing" is what we do with cars—we treat them as black boxes so long as they work. Other examples include the route we take to work every day; our use of an English dictionary which is itself written in English; etc. Of course, in practice, CYC and any such KB is of necessity finite. This does not concern us overmuch. Why? Sometimes, there are boxes which are (*currently*) still black to all of humanity—such as when we delve down to physical phenomena whose mechanisms are not yet understood; and most of us get by in the world quite well with a much larger fraction of black boxes than that.

The second way in which we claim meaning doesn't bottom out in an atomic base is illustrated by CYC's use of metaphorical sensibility. There are caches of popular metaphors, and in addition each slot P has a measure of how sensible (or common) it would be to say X when one actually meant $P(X)$. E.g., agent and physicalExtent have high metaphorical sensibility; one often says "The US did such and so" when they mean "Some agent of the US did such and so"; and one often says "Joe is huge" when one means "Joe's body—his physical extent—is huge." So one can state assertions like "Russia is angry", "Granadas guzzle gasoline" or "the relentless sun", and have them disambiguated and interpreted as (albeit much longer) legal (nonmetonomous) expressions.

The third way of looking at this issue, the one in which we murmur an affirmative answer to the question "Does meaning bottom out?", is to say that at any fixed level of abstraction, yes it does bottom out. We effectively drew on compositional semantics, above, to change this superficial Yes answer into a No, but we suspect that Smith does not believe in compositional semantics.

Question 11: Autonomous semantics

To the extent permitted by current sensors and effectors, the knowledge in CYC has autonomous semantics. CYC explicitly represents itself as a Program,

its current "run" as an Event, the users logged onto it as Human, their activity as KnowledgeEntering, etc. The frames about computer mice, mousing, and so on, tie in to actual events (as when a user moves their mouse or clicks a button; in such cases, CYC frames get created and/or modified). The user's actions cause revisions in the user's model (CYC frames), and that user model determines in a great many situations how CYC treats the user. The knowledge in the KB about people—such as eating and sleeping—is used to help guess why a user isn't responding at 12:30 pm or 12:30 am. And so on.

In areas where there is no meaningful overlap—such as ChurningCream IntoButter—the semantics are of necessity not autonomous. We do not believe that they need to be in order to understand and reason intelligently about those concepts (how many of us have ever made butter, after all?), and perhaps that is the crux of a genuine disagreement between Smith and us on this issue.

Question 12: Representing "as"

Far from ignoring the "'as' questions", the basic motivation for our paper and our current research (Feigenbaum's LSKB for engineering, and Lenat's CYC) is very much the brittleness of current systems. And much of that brittleness is due to what we have called the representation trap: using variable names pregnant with meaning—pregnant to the user, but barren to the system.

We choose to solve the "as" question empirically, by having our systems incrementally approach understanding. For instance, when a new piece of text is digested into CYC, a set of questions is raised, questions which "anyone ought to be able to answer." If CYC gets wrong answers, its KB is augmented. And the cycle repeats. Eventually, $DETENTE will not mean any less for the computer than "detente" means to us.

The final part of question 12 is the necessity of representing several different points of view for, say, the concept of detente. In CYC, we have a scheme for handling recursively nested propositional attitudes (e.g., Israel is afraid that Iran believes that Iraq expects that the USA will soon want to provoke a conflict with Russia in the Mid-East). To do this efficiently, we represent various sub-abstractions of the actors (e.g., Iran as Israel believes it to be) and have rules for projecting knowledge and beliefs, goals and dreads and expectations, from the "outer" world to the next "inner" world. One need only explicitly store the exceptions to what these projection rules would conclude. These rules are typically only run in a backwards direction, for efficiency reasons (this makes it increasingly costly to accurately simulate an actor with vastly different knowledge and reasoning methods than you have. Luckily, or perhaps by choice, we rarely have to deal with intelligent entities that don't share a large amount of knowledge with us.) And the various sub-abstractions are only created if we have something special to say about them, only if they have lasting importance. (Below, we discuss *temporal* sub-abstractions of actors and other objects.)

As stated in our general remark, above, this is the generous interpretation of his question 12. The less generous one would say that this is a genuine disagreement, and an irreconcilable one based on differences of faith. Even should we succeed in producing a generally acknowledged intelligent artifact, he might still refuse to acknowledge it on the grounds of this dimensional disagreement.

3.b. Genuine disagreements

We begin with a few basic problems with the material he presents before he begins discussing the various dimensions. The EC perspective Smith presents is too fuzzy to launch an attack against, so we shall restrict ourselves to particular "local" disagreements.

First of all, before he can criticize us or any AI paradigm for not having an adequate theory of representation, Smith has to define adequate. We claim that it's enough to have model theory, the theory of descriptions, etc.

The remark about "researchers rallying" around EC signifies nothing; we recall similar rallies in AI (e.g., around resolution), not to mention the numerous fads in philosophy. Actually, it signifies something a bit worse, given that the EC fad has been around longer than the CYC project, and has consumed a vastly larger annual budget. To wit, shouldn't they have something to show for all those years of work by now (i.e., some theoretical foundations built on a computational framework, embodied in programs)?

Smith seems to assume that the right way to go about developing a field (especially something like a logic) is to sit down, get all the foundations straight, and then start using it. (In this case, he's groping to try to erect a foundation that would compete with logic.) But that's not how the game is played. Consider what happened with logic. The big advance came with the *Principia Mathematica*, which was nothing but the CYC for mathematics. Building it exercised and honed logic. Tarski, Gödel et al. could then—a couple of decades later—set the foundations. So the best bet for Smith, and for what he calls EC, is to try something real with it. Forget the "decades of debate".

Question 1: Explicit representation

A trend that Smith begins here, and that we see throughout most of the twelve questions, is his equating us with the logic position, as if that somehow shows something bad about us. But there is nothing at odds between, say CYC and the proposed Advice Taker (as articulated in McCarthy's classic paper over thirty years ago; for a more recent update on this point of view, see [14]). The difference between the logic position and ours is principally one of focus: we think that our research time can best be spent actually trying to build big commonsense KBs, and they think it's not time yet, and so they continue building tools which eventually will be used to that end.

Another trend that keeps recurring in his treatment of these twelve questions, and which is first illustrated here, is the following. Smith mentions some very general and also very well-known problem (e.g., that explicit representation may result in unwarranted definiteness and premature categorization), and then slips in (as it were) the EC view toward eventually attacking that problem. The unstated parts of such an "argument" are (a) L&F have an approach to attacking it as well, and (b) the EC view is just that—one untried, fuzzy proposal which may or may not solve the problem. To the extent that the EC view is defined as "something which solves this problem", it's no wonder it's still unarticulated.

A third trend, which we will stoop to illustrate here, is ironically the sort of "tunneling" that he accuses us of. E.g., he goes from a true assertion ("interpreted code runs slower than compiled code") to an unfair generalization ("explicit representation leads to programs that are poor in general") and then back down to a few downright false specializations ("such programs are less effective"). But "effectiveness" means what can (ultimately) be derived from a program; if explicit representation has any effect on this attribute, it is to improve it, not decrease it! Similarly with his next target, "control flow". This means something like "how easy is it to decide, and include information affecting, what to do next, at each moment". Again, if explicit representation has any effect on this attribute, it is to improve it, not decrease it! His third target here is the negative effect this has on "overall system architecture". But that involves things like what knowledge the program can use, for what purposes. Given a fixed representation, an explicit one is likely to be usable by more types of architectures than an implicit one.

We shall give here just one more example of Smith's tunneling, and then try to restrict our attention to more substantive issues: Late in the question 8 discussion, Smith discusses the fact that traditional logic does not deal with certain issues (how to think creatively about the world, etc.). He then concludes that the problem is uncorrectable (we disagree), and logic is therefore woefully all wrong (we disagree), and therefore EC must be the answer (even if logic were wrong, this does not follow).

Question 2: Contextual content

Smith confuses the common practice of natural languages to economize (the meaning of "1989" or "tomorrow" is of course context-dependent) with the necessity of a KB to have this confusion. In CYC, e.g., there is a separate frame for 1987 in the Gregorian and Hebrew calendars, and a third frame for the word (so to speak) "1987", whose referents include both of those Event frames. There is little need, or benefit, in having the system itself confused about the meanings of "1987", any more than it's useful for a person not to understand the meanings—even if that person uses the word "1987" to mean one thing at one time and another thing at another time. So CYC in a way

appears to have "situated" knowledge, but that is—thankfully!—just a superficial phenomenon.

This is in a way Smith's main point, so let's restate the part of his position that we agree with. Consider linguistic utterances (such as "the time is 4 p.m."). The "meaning" of such utterances is highly dependent on context. I.e., one cannot expect the truth of the sentence (by itself) to be preserved if you transplant it from one conversation to another. So if our representation were going to consist of such (natural) linguistic utterances, we would need some notion of context in ascribing meaning to our utterances.

That much we agree with. But then comes a huge jump in the argument: from such natural language utterances, Smith concludes that the same will apply to utterances (propositions) in logic. I.e., he jumps to the conclusion that one cannot ascribe meaning to a logical assertion independent of context.

The main argument for this is that utterances in logic are, after all, in some sense linguistic statements. The flaw in this argument is that there is a crucial difference between sentences in logic and sentences in natural language. Natural language sentences presume common sense, user modeling, etc., on the part of the listener, and utilize this to become relatively (compared to our corresponding logical encoding) terse—at the price of introducing ambiguities in word sense, ambiguities in pronominal referents, ambiguities in metaphorical and analogical references, and ambiguities in the interpretation of ellipses. By contrast, our sentences in logic have been "universalized" [16] to the extent humanly possible (but see the next paragraph): including explicit clauses that refer to the sorts of contextual information that would be omitted in natural language utterances. So Smith's argument for why non-situated representations are meaningless is just sophism, i.e. relies on a confusion between a natural language utterance and a logic utterance. If exactly the same notion of meaning etc. held in both, we wouldn't need to invent the formal languages of logic, would we?

There was an important clause in the previous paragraph: ". . . to the extent humanly possible. . .". Can we truly "universalize" a sentence? Is it really possible for us to unearth all the contextual information and make it explicit (without possibly introducing new implicit contextual assumptions?). The answer (both from the work of Carnap et al. and from results such as Gödel's theorem) is that this is indeed very very hard.[4]

Not one place did Smith make it clear what exactly this beast "context" is. The moment Smith lapses into what the EC position is, we get lots of fancy words used in a very fuzzy sense. E.g., "the egocentricity obtains in virtue of the machine's existence, not in virtue of any self-reference." Being that "fuzzy" is not bad science—it's simply not science at all. It might be foolhardy

[4] Basically, we have eliminated contextual information when only a single model (up to isomorphism) satisfies our axioms.

for Lenat and Feigenbaum to be ambitious, but we are at least precise in what we say. Mystifying contexts is of no use. Much better is to try to come to grips with it, and we know of no better tool for this purpose than logic. Just because we can't 100% universalize a statement in logic does not mean it is inadequate and should be abandoned.

So, alright, we need to deal with this contextual effect. What this means is that we should go and make this concept explicit in our representations and this is exactly the stance being pursued today by John McCarthy (what he calls "contexts"), Guha (what he calls "microtheories"), and others.

In a way, Smith's "computational examples" of context-assuming programs argue against his position, not for it. When someone at MCC sends a mail message to DOUG, it reaches me. Why? Because the operating system accesses a file which quite clearly defines how to disambiguate such partial addresses. I.e., it contains a simple yet adequate explicit model of context.

A similar response applies to his remark that CYC "wouldn't know what time it was." Natural language interfaces, or other programs written on top of CYC, would have the job of answering such questions. They in turn would call on the CYC KB to do various sorts of disambiguation. The KB in turn has explicit models of its being used (in this case, the particular conversation going on at a certain date and time, on a certain terminal, what has been said so far, etc.) and from that it is straightforward to disambiguate references to "today" and "next year". This is not much different from the way the e-mail program disambiguates what DOUG means.

If you broaden Smith's first two objections (about "explicit representation" and "not being situated") to the levels that he and his references typically intend—namely that a program cannot possibly be intelligent unless it "lives" in the real world and has direct sensory experiences—then we patently disagree with such mysticism. To dip our toe into Smith's metaphysical swamp, we might say "Yes, our KBs are indeed 'somewhere': they are where they are being used." That in turn would suggest that they should contain explicit models of situations where we expect them to be used, not used, etc., i.e. a rich explicit meta-theory about the scope and limitations of use of our systems. Just such a scheme was discussed in detail in [11].

Question 3: Content depending on use

We definitely hold that we can and are constructing knowledge bases whose content means something; i.e., KBs which have meaningful content independent of any particular use of that knowledge. We are unhappy with the somewhat vicious tone that Smith uses in his review about this issue; and we find it surprising that he believes in the negation of such a possibility. E.g., if you consider some of the facts ("George Washington was the first President of the USA") and heuristics ("If it's raining, then the ground is probably getting wet") in our large KB, it seems to us that we have precise, commonly agreed

upon meanings for each of them, and for each of the terms they mention. We may use the second piece of knowledge to answer a variety of questions, such as deciding if we should watch where we're walking, or to guess whether or not the concrete mason will bother showing up to try to work on our driveway today, or to guess at why there is a large puddle of ammonia today in front of our spaceship (landed last week on some moon of Jupiter).

Of course the kind of uses of a proposition may be limited and biased by the way we choose to represent it, and we as reasoners are limited by what knowledge we choose to represent in the first place. The net effect of this is to make there be a pragmatic limit on the multiple uses of the knowledge in a KB. There aren't an infinite variety, there is a bias making some more natural or efficient, and the choice of contents of the KB limits the in-principle macro-level uses (problems tacklable). The net effect is at least multiple-use, if not truly use-neutral, knowledge.

Use-dependent meaning ("Is there water in the refrigerator?") does not imply we have to abandon the computational framework of logic. It might mean not insisting on an absolute account of the world. In fact we (and symbolic AI in general) don't even take a stance on the existence of such an absolute account. On the other hand, Smith seems to be insisting that there indeed exists such an "all independent" notion of meaning. Yes, of course the meaning of the English word "water" depends on the discourse in which it is used. This does not imply that we abandon explicit representation. It simply argues—and we would agree—that we should represent knowledge about discourses (common types, communication conventions, etc.). A large task? Yes. A theoretical impossibility? Hardly. The concept of use-dependent meaning only undermines the concept of soundness if one is reckless in introducing it.

Smith presumes much too direct a translation between the English word "water" and the term Water in the KB. He is assuming that we opt for a close connection, so that the natural language/logic translation is easy. However, we opt for as "deep" a representation as possible, one that often is quite far removed from the accidents and surface phenomena of English or any natural language.

Smith then goes on to point out some of the assumptions made in our research program—such as compositional semantics. We point them out, too [6, 12]. Every research program must have, and does have, numerous assumptions behind it. Not being prepared to make any assumptions leads only to apathy or to sterile argument. His approach implies that every problem in AI is "AI-complete"; perhaps this explains his hesitation to decompose problems.

Smith appears to be confusing the role of logic in mathematics (where it was used not as a real computational tool but as a precise language in which to state a minimal set of axioms from which everything else would follow) and the role of logic in AI. Precision (or rather a lack of it) is not even an issue that

computer scientists can choose sides on: programs are precise, period. Logic is used in AI for its other properties such as having a denotational semantics, modularity, etc.

He frets that "nothing in the KB means anything". Well, there are a lot of expert systems out there built on logic that are very useful—their users would not care that Smith feels that they don't "mean" anything. Then, at the very end of the question 3 tirade, Smith asks us to rely on his "experience". If he has scientific evidence as to why we won't succeed, he needs to be more precise than just saying "it is simply my experience".

Incidentally, a rich and quite readable account of the "bottoming out" of metonomy into somatic metaphors is given by Lakoff and Johnson [8]; we encourage Smith and other interested readers to examine it.

Question 4: Consistency mandated?

We disagree with Smith's comment that logic hates and avoids inconsistency. That is a rather dated point of view. Inconsistency at some point is the hallmark of any nonmonotonic system, and a vast amount of attention has been focused recently on how to deal with this. Perhaps this is again a case of his equating logic in math with logic in AI.

Questions 6 and 8: Only discrete propositions?

We do believe that discrete propositions can arbitrarily closely model continuous phenomena. More importantly, they can do it adequately and efficiently for real-world problem solving. And they can capture whatever is worth capturing about a situation. That is, one need never *in principle* throw up one's hands and say "you just had to be there, I can't describe it!"

There is nothing to prevent one from adequately describing the terror and confusion at a theater fire, or the trials of committee work (here "adequate" means that conclusions could be drawn about something involving, say, a theater fire, conclusions which enable the problem solver to correctly predict victims' reactions and memories, media coverage, pre-catastrophe fire codes, etc.).

Smith's very example disarms him: the implicit assumptions that writers build their text upon. We have looked at thousands of such snippets, and continue to look at them, chosen from such diverse sources as encyclopedias, novels, and newspaper advertisements. Of course there is a tremendous amount of unstated assumptions, presumed shared experiences and knowledge—indeed, that is precisely what we hope to capture and represent in CYC. But as for implicit *nonconceptual* inferences, we have yet to run across one. All such apparent references have so far been successfully reduced to discrete concepts and propositions involving them.

Smith seems to be confusing the underlying computational formalism (a digital one) and a representation built on top of that. That the former is digital

does not much impact on the latter. Does he want us all to go and build analog computers?

He then complains about the limitations of bivariance (having just True and False and perhaps a few other symbolic truth values). Our defense is twofold: First, the observation that the world is not vague (though language is); and second, we symbolic-AI'ers can get far enough with just what we have (far enough to, say, one day pass the Turing test).

As for his abjuration that we must provide more details on what our notion of inference is . . . we agree. Besides [2–5, 12], we are preparing an article dealing specifically with that issue.

Question 7: Do representations capture all that matters?

Earlier, in Section 3.a, we discussed our confusion over Smith's use of several terms and examples. There are some disagreements here as well, but we shall only call attention to the final line of this section: "there is no way in which L&F's system would ever be able to understand the difference between right and left."

We're rather puzzled by this. CYC can know about right and left propositionally, the same way it knows about hunger and democracy and computers and ownership. The assertional right/left knowledge can be related to digitized images, room floor plans, asymmetric particle physics phenomena, etc., but this is more of an affectation, a luxury, than a necessity. If a program uses "right" and "left" properly in sentences, answers queries involving it (e.g., "Which particular muscles does Connors use in his backhand?"), and acts appropriately (e.g., "Please open the rightmost pod door, HAL"), what more could be required in order to warrant our admitting that it understands the right versus left distinction?

Question 9: Participation and action crucial?

Of course participation often helps one understand a situation—especially in a field which is pre- or non-theoretical. But even in moderately well-understood fields it is "optional" (e.g., men can be gynecologists; non-criminals can be lawyers; and so on). And what does it mean for a college student to "participate" in Einstein's equations or other areas of math and theoretical physics?

Smith seems to believe there must be some fundamental reason we could never handle "See you tomorrow", or knowing that "tomorrow, today will be yesterday". Our response is essentially to repeat the above cry that such reasoning can easily be formalized and automated. We shall treat this example in a bit more detail, to convey the flavor of how we actually handle this sort of reasoning in CYC.

We handle the uttering of "See you tomorrow" by creating a 'frame' E_1 in CYC to represent that uttering event (E_1 is an instance of the set of all events).

Each event is grounded in time (whether or not the absolute time is known), and the meaning of "tomorrow" is clearly the day after this event E_1 takes place. A CYC "frame" E_2 would be created, representing a second event. E_2's temporal grounding would be "the day following the day E_1 takes place", and E_2 would be a seeing or meeting type of event.

This technique has been known to logicians since 1924. They (as does CYC) use an abstract (non-situated) notion of time. Smith claims you can't really compare 5 minutes of CPU time and 5 minutes of waiting at a train station. They (because of different situations) are simply different, incommensurable things. It is difficult for us to take his point of view seriously.

His more complicated example—and many much more complicated ones—are also straightforwardly handled in CYC. To see how, we must first discuss in a bit more detail how CYC handles time.

There is a pragmatically adequate language for describing pieces of time (unifying both set- and point-based abstractions of time), and fifty temporal relations—predicates (slots) which relate one piece of time to another. Although we originally kept separated (a) events and (b) the time intervals over which they occur, several years of experience at knowledge entering convinced us to combine those, so that each "frame" representing an event may have those fifty temporal relation "slots". One important class of events are *objects* (i.e., each object has a starting and ending time, can end-at-the-same-time-as another object, and so on). Another important class—a superset of the previous one—is *temporal sub-abstractions of objects*. Only those objects which are useful or required are created and represented explicitly; even more importantly for finiteness, only required sub-abstractions of objects are represented explicitly (typically, new sub-abstractions are created dynamically as required during the solving of a problem).

Now we can explain how CYC handles the effects of actions occurring and time passing: the basic idea is that each (frame representing an) event has actors (before, during, and after the event) which are temporal sub-abstractions of objects. Rules can thus be stated as to, e.g., the effect of taking a time-delay poison; they effect a particular sub-abstraction of the victim which is related to the present event's actor (the present event is the taking of the poison) in a clearly expressed fashion (expressed using the vocabulary of temporal relations).

4. The logical point of view

Here again we see Smith confusing the Nilsson kind of logic approach to AI (where all that's done with a KB is to prove sentences) and the McCarthy kind of logic approach to AI (where a declarative KB is used by all kinds of programs).

Yes, we are in many ways just a variation on that second "logicist" theme. The main difference is that we think it's high time to start trying for "a competent axiomatization", that additional work on reasoning is either unnecessary or, more likely, should be guided by the difficulties encountered in such an attempt. The "dig" about the expressiveness of our language is of course unwarranted, as this was a position paper and not an account of our current research projects. The dig is also, as so much of his review, simply false. See, for example, [12] for a several hundred page account of our representation language, inference engines, ontology, and yes, even some remarks on our paradigm.

5. Conclusion

Smith seems to be saying that AI can't move forward until we solve all the problems that have been haunting philosophers for centuries. We have tried to clarify why we disagree.

Just because "wheel-barrows are inadequate to try crossing Europe" does not mean that our existing representation technology is inadequate to try representing world knowledge: Analogies can of course be false. As we discussed, coincidentally, in our paper, this is especially likely if (as in the wheel-barrow case) there is no causal basis whatsoever for it, if it was chosen merely for dramatic impact on the reader.

We are hopefully more at the "age of discovery" stage in AI, where attempted ocean crossings will at least point out the inadequacies in our current vehicles and lead to improvements, and may lead to some surprising discoveries as well, even if we fail to reach our ultimate goal. Of course "work lies ahead of us", but let necessity and utility guide that work, not aesthetism and faith. Let's grope towards being Newton, not Aristotle.

Smith has us pegged correctly at the end, as disagreeing with Yeats' romantic but pessimistic mumbo-jumbo. We would say that inanimate objects and lower animals can embody truth, but Man (and one day AI) are distinguished because they *can* know it.

Acknowledgement

Discussions of the Smith piece which we have had with R.V. Guha and John McCarthy have provided many useful insights and examples, which we have included in this reply.

References

[1] M.R. Genesereth and N.P. Singh, Knowledge interchange format, Logic Group Rept. 89-13, Stanford University, Stanford, CA (1989).
[2] R.V. Guha and D.B. Lenat, The CycL representation language—Part 2, Tech. Rept. ACT-AI-452-89, MCC, Austin, TX (1989).
[3] R.V. Guha and D.B. Lenat, The world according to Cyc—Part 2, Tech. Rept. ACT-AI-453-89, MCC, Austin, TX (1989).
[4] R.V. Guha and D.B. Lenat, The world according to Cyc—Part 3, Tech. Rept. ACT-AI-455-89, MCC, Austin, TX (1989).
[5] R.V. Guha and D.B. Lenat, The CycL representation language—Part 3, Tech. Rept. ACT-AI-454-89, MCC, Austin, TX (1989).
[6] R.V. Guha and D.B. Lenat, Cyc: a midterm report, *AI Mag.* **11** (3) (1990) 30–59.
[7] P.J. Hayes, The second naive physics manifesto, in: R.J. Brachman and H.J. Levesque, eds., *Readings in Knowledge Representation* (Morgan Kaufmann, Los Altos, CA, 1985) 467–485.
[8] G. Lakoff and M. Johnson, *Metaphors We Live By* (University of Chicago Press, Chicago, IL, 1980).
[9] D.B. Lenat, Software for intelligent systems, *Sci. Am.* **251** (1984) 204–213.
[10] D.B. Lenat, A. Borning, D. McDonald, C. Taylor and S. Weyer, Knoesphere, in: *Procceedings IJCAI-83*, Karlsruhe, FRG (1983) 167–169.
[11] D.B. Lenat, R. Davis, J. Doyle, M.R. Genesereth, I. Goldstein and H. Schrobe, Reasoning about reasoning, in: F. Hayes-Roth, D. Waterman and D.B. Lenat, eds., *Building Expert Systems* (Addison-Wesley, Reading, MA, 1983) 219–240.
[12] D. Lenat and R.V. Guha, *Building Large Knowledge Based Systems* (Addison-Wesley, Reading, MA, 1990).
[13] D.B. Lenat, M. Prakash and M. Shepherd, CYC; Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks, *AI Mag.* **7** (1986) 65–85.
[14] J. McCarthy, Some expert systems need common sense, *Ann. New York Acad. Sci.* **426** (1983) 129–137.
[15] J. McCarthy and P.J. Hayes, Some philosophical problems from the standpoint of AI, in: H.J. Levesque and R.J. Brachman, eds., *Readings in Knowledge Representation* (Morgan Kaufmann, San Mateo, CA, 1987) 335–343.
[16] W.V. Quine, Natural kinds, in: *Ontological Relativity* (Columbia University Press, New York, 1969) Chapter 5.