PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1988,
edited by ICOT.

WHEN WILL MACHINES LEARN?

Douglas B. Lenat

Principal Scientist
MCC Artificial Intelligence Laboratory
3500 Balcones Center Drive
Austin, Texas 78759 USA

ABSTRACT

AI programs (such as AM, Bacon, Eurisko, and LEX) already exhibit rudimentary types of learning capabilities, so why don't they just "keep on learning" and become generally intelligent? We argue that a learning program should start off already knowing the tens of millions of facts, heuristics, representations, etc., that comprise our late twentieth century consensus reality knowledge. That is the ten-year goal of the CYC project, which we've been working on at MCC since late 1984. If we succeed, Knowledge Acquisition in the post-CYC era will be not unlike the human teacher-pupil paradigm.

1 INTRODUCTION

Computer programs can induce specific discriminations and learn narrow skills today, so why don't they just "keep on going", learning more and more knowledge? Why can't we just "send them to school"?

Most of our learning occurs at the *fringe* of what we already know. As Minsky has said, "we learn *which*, not *what*." I.e., we learn that some new thing X' is similar to something we already know about, X, but here are the differences...*

This observation is sometimes phrased as the maxim:
The more you know,
the more (and faster) you can learn.

Well, how much *do* our learning programs know, to begin with? The answer, even for the largest machine learning programs (e.g., Eurisko), is that they know only a tiny, tiny fraction of what even a six year old child knows. The programs know a few paltry hundreds of things, but humans know almost a million times that many things!

So the inverse of the above maxim, then, is the real culprit:
If you know next to nothing to begin with,
you won't (quickly) learn much.

There are really two ways out of this dilemma:

(1) The "100% Natural" Approach: Figure out all the instincts, skills, needs, drives, and predispositions to learning that Nature (Evolution, God,...) has hard-wired into human brains and spinal cords and sense organs, and figure out how neonates' raw perception refines into usable knowledge. Then build such a system incorporating all of those things, plus of course the right sort of "body" (which includes sensors and effectors and a lot more) and allow it to "live" for years in the real world: nurture it, let it play, let it bump into walls, teach it to talk, let it go to kindergarten, etc.

(2) The "Prime the Pump" Approach: Codify, in one immense knowledge base, a large fraction of the tens of millions of facts, heuristics, representations, etc., that "everybody knows". This body of knowledge might be said to comprise *late twentieth century consensus reality knowledge*. They're the things that we each assume the other knows, when we open a conversation with a stranger. They're the things that the writer of a newspaper article, or an encyclopedia article, or a dictionary entry, can safely assume that the reader already knows.

Once the large consensus reality knowledge base exists, either via methodology (1) or (2), then the everyday sort of "fringe learning" takes over, and the system should be educable in the usual ways: by giving it carefully graded series of readings to do, asking it thought-provoking questions, and helping it over novel or difficult parts by posing a good metaphor drawn from its existing knowledge base.

* By the way, each of *them* -- each difference -- is either something we're already very familiar with, such as "but the size is much larger" or, in rare cases, recursively explained by relating it to some sort of difference metric we know well, and explaining the differences from *it*.

There are many researchers who are working on limited forms of approach (1) -- e.g., the CMU World Modelling Project -- and approach (2) -- e.g., the Stanford KSL Engineering Design Project.

The CYC project, which we've been working on at MCC since late 1984, is aiming at the fully scaled up approach (2).

We knew when we started that there would be many "representation thorns" (e.g., how to deal with time, space, belief, stuffs (mass nouns), etc.) and "methodological thorns" (e.g., how to keep the knowledge enterers' semantics from diverging) that we would have to overcome. At the time we began, we estimated that there was less than a 20% chance of our project succeeding. Undertaking such high-risk high-payoff long-term projects is precisely what MCC was created for. We've clipped enough of those thorns for us to raise that estimate now to 60%.

There are other recent articles (see References) which describe the CYC project in great detail, and explain how each of those problems has been "solved". "Solved" means that we've found an adequate way to handle the 99% of the common cases that crop up in everyday life. E.g., CYC can't easily represent and reason about "the Cantor set of moments from three to four p.m." -- but then again, neither can most people!

In this article, we will limit ourselves to some brief remarks about knowledge acquisition in and by CYC.

2 ANALOGY

There are two ways that people -- and, one day, CYC -- can cope effectively with novelty:
(1) fall back on increasingly general knowledge, and
(2) analogize to specific but superficially disparate knowledge.

One important special case of (1) is to fall back on investigative and learning methods, such as asking someone for help, or trying to discover the answer oneself. The answer one gets, from other people or from Nature, is then, recursively, a new bit of "novelty" that can and must be dealt with via (1) or (2).

So the mattress in the road to AI is "Lack of Knowledge", and the anti-mattress is Knowledge. But how much does a program need to know, to begin with? The annoying, inelegant, but apparently true answer is: a non-trivial fraction of consensus reality. If I liken the Stock Market to a roller-coaster, and you don't know what I mean, I might liken it to a seesaw, or to a stormy sea. If you still don't know what I mean, I probably won't want to deal with you any more.

2.1. Using Analogy to Acquire New Knowledge

It's pretty clear how to cope with novelty by falling back on increasingly general knowledge. What of falling back on analogy? Isn't analogy just an isolated curiosity, or a literary device which has dramatic power but no real heuristic power? See for yourself. Here is a partial analogy between treating a disease and waging a war:

Treating a bacterial infection

```
enemyType:  Disease
enemyLocal: Bacteria
protagonistType: Physician
enemyProcess: Infecting
protagProcess: ClinTreating
usefulPreprocess: Diagnosing
usefulTactics:  Vaccination
locale: BodyPart
```

Fighting a war

```
enemyType: MilitaryForce
enemyLocal: EnemyTroops
protagonistType: Soldier
enemyProcess: Invading
protagProcess; MilRepulsing
usefulPreprocess: J2ing
usefulTactics: MilContainment
locale: GeographicRegion
```

Notice that some of the concepts on one "side" are analogues of those on the other "side". But Vaccination and MilContainment aren't analogues. In fact, they each have no analogue (yet) on the other side. This is bound to happen, and is an opportunity to use the analogy. Namely, we can let the analogy guide the search for new, useful concepts on each side.

For instance, maybe we should define a medical analogue of "military containment", and a military analogue of "vaccinating". The former might be "medical containment" -- e.g., the use of a tourniquet on a venomous snakebite, or the use of quarantine on a virulent plague. The military analogue of vaccination might be "fortifying" or "propagandizing". To take the analogy even more seriously, the military vaccination might entail letting a small group of enemy soldiers overrun some territory, before our friendly forces secure it, as a way of driving home to the local populace just how bad the enemy is.

That illustrated the utility of analogy as a *guide for defining* new concepts. Analogy can also help *flesh out* the new concepts. For instance, what is "medical containment" containing? what does it locally constrain? how should one do it? To answer those questions, go to the "military containment" concept, look up the answers there, and map them back using the existing analogy:

```
MilContainment
  usefulTacticIn: Fighting-a-war
  containedType: MilitaryForce
  containedLocal: EnemyTroops
  attributeLimited: Mobility
  howTo: Bound&Isolate
  counterTactic: (Threaten containedArea)
  containedArea: GeographicRegion
```

That suggests that medical containment, in the case of treating a bacterial infection, is containing a disease, and, more locally, bacteria. It might be done by surrounding and isolating the infected part of the body.

We could carry through a similar procedure to guess at the values of various slots on the military analogue of Vaccinating.

The example above, leading to military vaccination and to medical containment, illustrates that analogy can be useful in suggesting new concepts and in fleshing them out. That shows that analogy does have quite significant potential heuristic power. Moreover, as Lakoff & Johnson argue quite convincingly, in *Metaphors We Live By*, it is far more pervasive than one might at first imagine, appearing in almost every sentence we utter.

Our sentences and our thoughts are riddled with analogies (such as "riddled with") which in turn are built on analogies, and so on, like the skins of an onion. At the core of the onion is a small tangle of very primitive somatic metaphors such as forward, backward, up, down, hungry, tired, sleep, pain, push, breathe, and so on.

It's easy to imagine how this arrangement might have come about: We understand (and communicate) "which", not "what"; that is, we perceive (or transmit) something that's already well known plus a small set of changes. Learning and understanding and speaking occur at, and can modify, only the fringe of what we already know. So, as we live our life, new skins of the onion are added on.

2.2. Why Analogy Works Frequently

The preceding paragraphs have argued that analogy is frequently useful, and discussed specifically *how* it might be useful as a knowledge acquisition method. But *why* is it frequently useful?

The answer to that lies in the nature of the world we happen to inhabit, and the state of our understanding (and perhaps, our capacity to understand.) We see three aspects of our world, and ourselves, that make analogy frequently useful to us as human beings:
The moderate amount of novelty we're confronted with;
The moderate number of distinct causes in the world;
The mediocre ontology and "knowledge metric" we have.

Novelty: If the world were very volatile and chaotic and wildly unpredictable, analogizing would avail us little; it would almost always fail. If it were totally staid and static and unchanging, we'd have little need for analogy; memory would suffice.

Causality: Analogies which have no common root cause are superficial and weak and, more often than not, no good for more than literary divertissement. So if there were a zillion disparate causes in the world, analogy would usually be superficial and powerless in this fashion. On the other hand, if there were no variety of causal mechanisms in the world (e.g., only one emotion, only one method of physical propulsion, only one kind of chemical reaction, etc.), there wouldn't be much power in classifying which of those causes was behind something. (In fact, there wouldn't be much point in having those terms, either.)

Knowledge metric: If we had a terribly wrong view of the world, analogy would lead us even further into error (e.g., thinking that the cosmic objects and meteorological phenomena are sentient, one might try to bribe them with offerings.) If we had a terrific grasp of the world, we'd always *know* what knowledge was relevant to our present dilemma, and exactly how it should be applied. We wouldn't need analogical leaps into the possibly-relevant. In many situations, we have *some* knowledge as to what aspects may be relevant to the decision-making problem, but we don't know how to 'compute the function' from those aspects to the correct decision. Analogy is useful because it allows us to find and borrow from other situations in which the 'value of the function' is known.

In other words, analogy is a useful heuristic method to employ because we and the world happen (?) to fall near the midpoint of those three axes. A skewing along any one of them would reduce the power of analogizing.

2.3. How to Do It

Ah, here is the problem. Analogy can be great, frequently used, and perhaps even indispensible. Now, how do we get a program to do it? Uemov (1964) drily notes:

> *When analogy is successful it is called 'deep', 'strict', 'scientific' and so on. When it fails, it is called 'superficial', 'loose', 'unscientific' and so on. Naturally, there is a question as to how these two types of analogy can be distinguished before the practical realization takes place.*

The simplest model is that of structurally mapping slots of one frame to slots of another frame. This must be generalized to include mapping between one network of frames and another network of frames; and must include knowledge-guided reformulation to reveal commonalities between imperfectly-matching entries, and likewise between imperfectly-matching slots.

Example 1: Consider the statement "Fred is ursine". Presumably that means to map Bear (and the cluster of units associated with Bear) to Fred (and the cluster of frames associated with Fred). We might want to map the Bear's qualitative size (Large, compared to the typical WoodlandCreature) to Fred's qualitative size (Large, compared to the typical HumanMale.) That was easy, since both the slot (qualSize) and the value (Large) are the same. We might map the Bear's claw-length and claw-sharpness to various attributes of Fred's fingernails, which would be a cross-slot mapping. We might map some absolute numeric values (such as 10') to different numbers, and so on.

How can a program automatically find this (or other good) mappings? I.e., how does it notice to even try to do this, and how does it manage to carry it out?

To answer these important questions, let's ask: *why* might a person make that analogy and utter the statement "Fred is ursine"? If you think about it for a minute, the surprising answer is that this is not a powerful analogy after all, it's just a nice, powerful, and compact way of communicating a bundle of information that just happens to be true. There is no causal connection here, just coincidence. All the speaker was doing was (a) compacting their message, and (b) injecting a little humor and hyberbole and variety into their speech. So it's not hard to imagine that a program could notice that a good match exists between Fred and Bear, and that Bear is well known to the typical person, and, if the current problem is to describe Fred, it could decide to refer to Fred as ursine in order to communicate a lot about Fred all at once. A large fraction of humans' use of "analogy and metaphor" is not analogy or metaphor at all, then, but rather falls into this category of merely compacting a message by finding superficially similar "analogues". Let's turn to an example of a "real" analogy.

Example 2: "Mowgli is wolflike". This sure seems similar to Example 1, at least on the surface. What's different about it? Here, in contrast to Example 1, there is a *causal connection* underlying the analogy, since Mowgli (the character from Rudyard Kipling's *Jungle Book*) was raised by wolves. One's upbringing is a strong determiner of one's attitudes toward food, shelter, possessions, ethics, life, death, music, physical conditioning, and so on. If we ask you whether Fred (from example 1) likes to catch fish, or eat them, or eats raw fish..., it's unlikely that the answers from Bear will prove to be reliable guesses. But if I ask you whether Mowgli likes to catch fish, or eat them, or eats raw fish, then you would expect to be able to guess the answers by looking at Wolf. To answer Uemov's skepticism, we remark that there's not too much bias in this example, since we don't know the values of these attributes on Wolf, or on Mowgli, but we'd be surprised to learn that there's any episode in *The Jungle Book* that shows them to differ.

There are several routes by which a program might first suspect this analogy:

(1) by noticing that Mowgli was raisedBy Wolves, and that raisedBy strongly *determines* many other important properties. ("Properties x and y *determine* z" is used in the sense of [Russell 88]; namely, objects sharing the same value for their x and y properties will likely have the same z values as each other.)

(2) by noticing that there are some unusual (uncommon) similarities between various attributes of Wolf-003 and Mowgli -- such as their tableManners, howlingFrequency, goals, dreads, and gait; and then
(2a) trying to "explain" these by finding some deeper, independent attribute(s) that determined them, and/or
(2b) trying to extend the analogy to other (not necessarily "deep") attributes to probe its boundary and simply because extending it is a cost-effective way of getting more information into the KB.

(3) by some external clue, hint, reference,... ["External" means provided explicitly by a human user, the novel, or some very far-flung piece of the system itself.]

(4) by random search for analogies. This is a special case of all three of the preceding methods, and is probably too inefficient to make it worth trying.

(5) by noticing "X is like Y, and analogies of type Z helped us with Y, so maybe analogies similar to Z will help with X". This may be viewed as a variant of method 2b, above, where X and Y are noticed as similar and then the usefulAnalogies slot of Y is mapped back to X.

In order to get these various schemes to work (excepting #3, *deus ex machina*), the system must have access to a large knowledge base, a rich space in which to prospect for matches. Most of the 5 schemes also require deep understanding of the knowledge in the KB, to permit obscure or imperfect matches to be made, to sort out the superficial from the significant matches, to posit and judge the plausibility of a common causal explanation for the similarities, and to decide along which other axes (slots) to search for further similarities that would extend the analogy.

We've glossed over some hard issues -- perhaps *the* hard issues -- in successful analogical reasoning, namely how *exactly* to pick a promising analogy, how to develop it, which slots to extend it to (e.g., usefulTactics) and which ones not to (e.g., inventor), how to tell if the phenomenon going on is really "exploiting a hitherto unrecognized common generalization", how to tell if the analogy is superficial and not worth extending any more, when to rely on it, and so on.

Our basic approach to answering all these questions is divide and conquer. I.e., we posit that "analogy" is a vague English word covering a multitude of types of inference. The types of analogical reasoning can be usefully arranged in a space whose dimensions include things such as the nature of the boundary (where the

analogy breaks down), the nature of the analogues (e.g., the hardness of each analogue's field), the nature of the attributes being matched, the purpose of the analogizer, and so on. Each cell in that n-dimensional matrix represents a type of analogical reasoning, and each of those types deserves to be teased out separately and studied. A bundle of heuristics can then be assembled for each cell, though of course many of the heuristics would apply to whole sections of the matrix.

We hope to tame Analogy through this two-pronged attack -- (1) breaking down the phenomenon into its various subtypes and then handling each one, and (2) having a realistically large pool of (millions of) objects, substances, events, sets, ideas, relationships,... to which to analogize.

3 CYC AS AN ACTIVE AGENT

The most easily forseen mode of failure for the CYC project was -- and is -- that the knowledge enterers might diverge. There are two types of divergence that can occur:

(1) Stepping on each other's toes: One knowledge enterer uses another's already-entered terms, but uses them to mean slightly different things. As a result, increasingly large amounts of wrong information are entered.

(2) Passing each other in the night: One knowledge enterer re-enters some knowledge that is already entered in the system, but under a different name. If the duplication is ever noticed, then it may be fairly easy to fix (either by relating the units to each other, or, in extreme cases, just merging them). Hence, the failure mode is to never realize that this duplication occurred in the KB.

Naturally, we must build up the CYC KB from *some* sort of primitives. We have claimed that it must be built from deeply understood knowledge rather than from complex "impenetrable" predicates (or slots or whatever).

At first, doing this just made life difficult; having a deep but small KB didn't pay off. Yet, fortunately, when we began to build CYC ever larger and larger, we found that the set of primitives began to converge. That is, it requires less and less work to enter each new fact. This phenomenon is not surprising (it was, e.g., predicted in Hayes' *Naive Physics Manifesto*), it is merely very very important. And it was quite comforting to see it really happen!

Since 1984, we've been building and organizing and reorganizing our growing consensus reality KB in CYC. We now have about half a million entries in it, and we expect it to increase by a factor of 4 by mid-1989. Thanks to an array of explicit and implicit methods for stating and enforcing semantics, they appear to be converging, not diverging.

Let's illustrate how convergence can occur. Consider a legal reasoning system which must advise someone whether to sue someone else. Say their car has been scratched, and there are plenty of witnesses. With little common sense, the system might fire a rule like

```
IF your property has been damaged by X,
   and there is little doubt as to the facts,
   and monetary recompense is not forthcoming,
THEN sue X
```

But if the car is scratched by a bag lady this may be a bad idea. So maybe the builder of this expert system would add a clause like "...and the perpetrator is not a bag lady". If s/he has an eye on more generality, s/he might phrase the clause "...and X is not destitute".

The CYC approach would be different. We would describe what the process of suing is (a way of getting money from a defendant), what money is, give very general rules about the process of transferring some X from Y to Z, including the precondition that some X must exist at Y in order to be transferred from there, the fact that there is some overhead cost to running a transfer process, and perhaps some special knowledge about who does what to whom and who pays what to whom during and after the suing takes place in America today.

From this, CYC could generate the appropriate behavior in this case. But more importantly, the system would now be able to exhibit robust behavior in an unimaginably large number of other cases also. For instance, it will know that if one has 50 cents in one's pocket one cannot pay for a movie ticket; and that the world's consumption of resources will eventually have to cease; and that one usually doesn't borrow money from bag ladies.

To solve the specific "car-scratching" problem, it's tempting to put in special case knowledge. But as you widen the definition of the problem domain (e.g., "acting intelligently in interpersonal situations involving money or property"), it becomes more economical to opt for the deeper, more spread out approach. In the limit ("acting intelligently"), such a policy is wildly cost-effective.

So we must build a good global ontology of human knowledge (i.e., spanning current human consensus reality) if we are to avoid the representation trap. Choosing a set of representation primitives (predicates, objects, functions) has been called 'ontological engineering', i.e. defining the categories and relationships of the domain.

This is empirical, experimental engineering, as constrasted with 'ontological theorizing' that philosophers have done for millenia. This project is Man's first foray into large-scale ontological engineering.

Well, what about encyclopedias and thesauruses? Encyclopedia writers, e.g., have been able to finesse 90% of this issue because
(i) an encyclopedia is largely a linearly ordered sequence of articles, so the main decision to make is "grain size", not organization, and
(ii) people learn early in life what sorts of topics will and won't have articles dedicated to them, and
(iii) to the extent that i and ii are insufficient, a peppering of cross references will help a person jump from an almost-correct place to the correct place.

As for thesauruses, go take a look at the table of contents of one, and you'll see an interesting phenomenon: it's terrible! For instance, one thesaurus was developed a couple centuries ago, and so one of the top-level divisions of knowlede is Theology; Physics is a sub-sub-part of Chemistry; and so on. And yet, the thesaurus works fine. Its job is not to be a good global ontology, but rather to be good locally, to clump together words with very similar meanings.

So both CYC and its ontology must be built almost from scratch. The question, then, is: How should such a gargantuan knowledge base be constructed? What methodology, what tactics, will suffice?

Developing a good set of primitives will be one of our "keys" to achieving convergence: defining knowledge in each area in terms of knowledge in other (often more general) areas. E.g., when explaining Baseball to CYC, it is in terms of generic actions like running, hitting, catching, competing, cooperating, taking turns, and so on. [This is also the source of power behind modularity in programs, primitives in programming languages, and grammatical structure rather than monolithic messages in natural languages.]

The other "keys" to semantic convergence are
(a) to have a sufficiently large KB that one can tell it something new by plucking a similar piece of knowledge and making some small editing changes to it, and
(b) to have a good enough global ontology to make that easy, and
(c) to have CYC function as an intelligent agent whose first important task is to help with its own continuing enlargement.

The simplest example of (c) is the assistance that CYC gives during the copy&edit process. Let's say we're copying France to create Italy. CYC examines the variation in the value of each slot on France, and decides whether it should have the same value on the Italy unit (e.g., continent: Europe; and instanceOf: Country), or one modified in a particular way (e.g., railSystem: FrenchRailSystem becomes railSystem: ItalianRailSystem), or some totally new value (e.g., chiefJustice; or capitalCity). In the case of ItalianRailSystem, CYC automatically creates the new unit and recursively enters the copy process on that, copying from FrenchRailSystem, but it carries along all the various substitutions, namings, and choices made by the user so far in the copy process. When this process ends, dozens of new units have been created in a matter of minutes.

We can briefly illustrate some more examples of (c) -- having CYC function as an intelligent agent helping in its own knowledge acquisition process.

As a new unit is being entered, the knowledge enterer sees a dynmaically updated display of which other units in the system subsume this one (contain all this information, at least.) That way, the user knows to keep adding more knowledge about the unit, at least until it is disambiguated from all the others in the system.

A more complex case of (c) is the error-checking that CYC does. Some of this occurs instantly, on each knowledge enterer's machine, as they try to enter some knowledge that violates constraints. Some occurs remotely, at the knowledge server machine, which reconciles nearly-simultaneous user operations. And some occurs off-line, when the server partitions out the checking process to multiple machines, and they report back on subtle "stepping on toes" errors, and on suspected "passing in the night" duplications.

That latter process also reports on potential useful analogies; passing in the night is really just a very strong -- and trivial -- sort of analogy, after all. CYC also helps with its own knowledge acquisition by noticing gaps and asymmetries in the KB.

4 CONCLUSION

The CYC project, which we've been working on at MCC since late 1984, is a full scale effort to codify the tens of millions of pieces of knowledge that comprise our late twentieth century "consensus reality."

A few years ago, shortly after we began, we published our initial plans [Lenat, Shepherd, et al. 1986]. Our schedule was to have enough of the KB built to transition to natural language understanding as the dominant knowledge entry mode in 1994. By now, we've gotten pretty far along. Not surprisingly, there have been unexpected problems and unexpected discoveries. Perhaps the biggest surprise is that we're still on schedule. The thorns we had to deal with -- time, change, the overall ontology, and so on -- have been faced up to and trimmed, rather than avoided.

We are taking the "Engineering Approach" to getting a large initial KB, from which open-ended knowledge acquisition may proceed automatically. Although we expect CYC to become an ever-more active intelligent agent, helping in its construction, at the moment most of the activity is manual, making "surgical" additions and changes to the KB.

Manual KB-building activity is not considered part of Machine Learning, so it may appear that I've disowned the Learning field. Nothing could be farther from the truth! The absence of such a consensus reality KB is the major bottleneck to Automated Knowledge Acquisition, so in that sense we *are* still working on Machine Learning. I also believe that the same absence is holding back progress in other areas, such as the Semantics part of natural language understanding, or getting expert systems to be less brittle and to cooperate with each other.

I was asked to write this paper on the topic of "Future Knowledge Acquisition" because my 1975-84 work on AM and Eurisko helped to spark the rebirth of the field of Machine Learning. I fully expect that CYC will spark an even greater renaissance in that field, and that I will be considered a Learning researcher again during the latter half of the 1990's. Since CYC started in late 1984, and has been scheduled as a ten year effort, a tongue-in-cheek answer to the paper's title question (*When Will Machines Learn?*) might be: Exactly six years from today!

ACKNOWLEDGEMENTS

We would like to thank all the members of the CYC project, who have contributed their ideas and code and units. Of special help were R.V. Guha, Mary Shepherd, and David "Gumby" Wallace. We also would like to thank our colleagues Phil Agre, Michael Brent, David Chapman, Al Clarkson, Bill Gooch, Adolfo Guzman, John Huffman, Mike Huhns, Alan Kay, Dave Loeffler, Chris Maeda, Margaret Minsky, Marvin Minsky, John McCarthy, Colleen Oresky, Stuart Russell, and Larry Stephens. We would also like to thank Elaine Rich, Jim Hollan, and Dave Wroblewski, for the great job their HI team will do in the coming year, building an interface that surpasses our wildest dreams.

REFERENCES

Lakoff, G., and M. Johnson, *Metaphors We Live By*, University of Chicago Press, Chicago, 1980.

Lenat, Douglas, and J. S. Brown, "Why AM and Eurisko Appear to Work", *Artificial Intelligence* 23, pp. 269-294, 1984.

Lenat, Douglas, M. Prakash, and M. Shepherd, "Cyc: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks," *The AI Magazine*, 6, no. 4, pp.65-85, 1986.

Lenat, Douglas, and R. V. Guha, "The World According To CYC," MCC Technical Report Number ACA-AI-300-88, September, 1988.

Lenat, Douglas, and E. A. Feigenbaum, "On the Thresholds of Knowledge," *Proc. IJCAI-87*, Milan, 1987.

Russell, Stuart, "Analogy by Similarity", in (David Helman, ed.) *Analogical Reasoning*, Boston, Reidel, 1988.

Uemov A.I., *Problems of the Logic of Scientific Knowledge*, Dordrecht, 1970.