AI Magazine Volume 6 Number 4 (1985) (© AAAI)

CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks

Doug Lenat, Mayank Prakash, & Mary Shepherd

Microelectronics & Computer Technology Corporation, 9430 Research Boulevard, Austin, Texas 78759

The major limitations in building large software have always been (a) its brittleness when confronted by problems that were not foreseen by its builders, and (b) the amount of manpower required. The recent history of expert systems, for example, highlights how constricting the brittleness and knowledge acquisition bottlenecks are. Moreover, standard software methodology (*e.g.*, working from a detailed "spec") has proven of little use in AI, a field which by definition tackles ill-structured problems.

How can these bottlenecks be widened? Attractive, elegant answers have included machine learning, automatic programming, and natural language understanding. But decades of work on such systems (Green *et al.*, 1974; Lenat *et al.*, 1983; Lenat & Brown, 1984; Schank & Abelson, 1977) have convinced us that each of these approaches has difficulty "scaling up" for want of a substantial base of real world knowledge.

Making AI Programs More Flexible

> [Expert systems'] performance in their specialized domains are often very impressive Nevertheless, hardly any of them have certain commonsense knowledge and ability possessed by any non-feeble-minded human. This lack makes them "brittle." By this is meant that they are difficult to expand beyond the scope originally contemplated by their designers, and they usually do not recognize their own limitations. Many important applications will require commonsense abilities... Common-sense facts and methods are only very partially understood today, and extending this understanding is the key problem facing artificial intelligence. —John McCarthy, 1983, p. 129.

How do people flexibly cope with unexpected situations? As our specific "expert" knowledge fails to apply, we draw on increasingly more general knowledge. This general knowledge is less powerful, so we only fall back on it reluctantly.

"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 knew without being told (*e.g.*, an object can't be in two places at once).

Abstract

MCC's CYC project is the building, over the coming decade, of a large knowledge base (or KB) of real world facts and heuristics and—as a part of the KB itself—methods for efficiently reasoning over the KB. As the title of this article suggests, our hypothesis is that the two major limitations to building large intelligent programs might be overcome by using such a system. We briefly illustrate how common sense reasoning and analogy can widen the knowledge acquisition bottleneck The next section ("How CYC Works") illustrates how those same two abilities can solve problems of the type that stymie current expert systems. We then report how the project is being conducted currently: its strategic philosophy, its tactical methodology, and a case study of how we are currently putting that into practice. We conclude with a discussion of the project's feasibility and timetable.

We would like to thank MCC and our colleagues there and elsewhere for their support and useful comments on this work. Special thanks are due to Woody Bledsoe, David Bridgeland, John Seely Brown, Al Clarkson, Kim Fairchild, Ed Feigenbaum, Mike Genesereth, Ken Haase, Alan Kay, Ben Kuipers, John McCarthy, John McDermott, Tom Mitchell, Nils Nilsson, Elaine Rich, and David Wallace

A third, important, immense, yet nonobvious source of general knowledge is one we rely on frequently: all of the *specific* knowledge we have, no matter how far-flung its "field" may be from our present problem. For example, if a doctor is stymied, one approach to deciding what to do next might be to view the situation as a kind of combat against the disease, and perhaps some suggestion that is the analogue of advice from that domain might be of use ("contain the infection," "give the infection some minor chances, as the risk is worth the information learned about its behavior," and so on). Unlike the first two kinds of general knowledge, which simply get found and used directly, this type of knowledge is found and used by analogy. In other words, the totality of our knowledge can—through analogy—be brought to bear on any particular situation we face; and that, after all, is what we mean by knowledge being "general." To perform this in "real time," we employ heuristics for prioritizing which analogies to consider first, and we may also use our brain's parallelism to good effect here.

Presenting an example of this sort of synergy that doesn't appear contrived is difficult. We refer the reader to Skemp (1971), Hadamard (1945), and Poincare (1929), to name a few, who document cases of the use of detailed analogies to aid in solving difficult problems. In notes only recently analyzed and reported (Broad, 1985) Edison describes how most of his inventions started out as analogues of earlier ones; *e.g.*, the motion picture camera started out looking like a phonograph and gradually evolved from there. Some evidence for the reliance on metaphor may come from the reader's own introspections: try to be aware in the next few days of how pervasively—and efficaciously—you employ metaphors to solve problems, to cope with situations, and in general to make decisions. Usually these metaphors are not very "deep;" rather, we get power from them because of the immense breadth of knowledge from which we can choose them. Lakoff and Johnson (1980) go even further, arguing persuasively that almost all of our thinking is metaphorical.

The test of this idea—solving problems by analogizing to far-flung specific knowledge—will be in the performance of the CYC system, once it has a large enough accumulation of specific knowledge.

The CYC project is an attempt to tap into the same sources of power by providing a comprehensive skeleton of general knowledge (to use directly) plus a growing body of specific knowledge (from which to draw analogies).

Making It Easier to Add New Knowledge

Interestingly, the large real world knowledge base necessary to open the brittleness bottleneck also provides an answer to the other bottleneck problem, knowledge acquisition (KA). Let's see why this is so. How do people learn and understand new information? In a recent essay, Marvin Minsky (1984) points out that humans rarely learn "what"—we usually learn "which." In other words, we assimilate new information by finding similar things we already know about and recording the exceptions to that "analogy." This leads to amusing mistakes by children ("Will that Volkswagen grow up to be a big car?") and by adults (*e.g.*, cargo cults), but these are just extreme cases of the mechanism that we use all the time to assimilate new information. In other words, we deal with novelty the best we can in terms of what we already know (or, more accurately, what we *believe)* about the world.

We have to date seen no AI system which tries to do knowledge acquisition "from strength"...

Another way of saying this is that the more we know, the more we can learn. That is, without starting from a large initial foundation, it's difficult to learn. We have to date seen no AI system which tries to do knowledge acquisition "from strength," from an initially given large, broad body of knowledge. If the knowledge base is large (and representative) enough, then adding a new piece of knowledge ought to be doable just by pointing to—and thereby connecting—a few existing pieces.

A very weak form of this process is in use today in building expert systems, namely the simple expedient of "copy&edit": As you build up your expert system's knowledge base, and you're about to enter the next chunk (rule, frame, script,...), it is often faster to find a similar chunk, copy it, and then edit that copy, than it is to enter (formulate and type in) that new chunk into the system from scratch.

One limitation to current uses of copy&edit is that if the growing system only knows a couple hundred things (rules, frames, etc.), then those form the total universe of potential objects from which to do the copy. As the size of the knowledge base grows, it becomes increasingly likely that one can find a match that's close enough to result in a large savings of time and energy and consistency. (The new piece of knowledge is more likely to preserve the existing semantics if it's been copied, rather than typed in from scratch.)

CYC is aiming at tapping this source of power, gladly swapping the problem of "telling the system about x" for the problem of "finding an already known x' that's similar to x." But there are two more powerful ways in which CYC may help the copy&edit process along. The first is to employ analogy to suggest or help the user choose particular existing frames from which to copy. The second is to use analogy to do some of the editing itself, automatically. The following section shows how these two new aids to knowledge acquisition would work.

Analogical Reasoning

Suppose one has begun to explicate the "medical treat-

Representing Knowledge in CYC

CYC's representation language is frame-based and is similar to RLL (Greiner and Lenat, 1980) and KRL (Bobrow and Winograd, 1977). One of the central principles in its design is that it be a *part* of the CYC KB itself. That should facilitate translating the system to other "underlying languages," and should allow CYC to apply its own knowledge and skills (question-answering and analogizing) to itself. We believe this is important if it is to monitor its own runtime behavior and enforce a consistent semantics on its builders and users.

To implement such a self-describing language, each kind of slot is given a full-fledged frame describing its semantics: What is its inverse? What kind of frames can legally have this slot? What kind of values can fill it? How can its value be found if none is currently there? When should such values get cached? What relations of various sorts does it participate in with other slots?

This last category of information (relations to other slots) is very important. It means that the set of known slot names forms a part of the CYC hierarchy of concepts. So a rule or script or slot definition that asks if Grebes is an element of Supergenuses can get an affirmative response even though there is no elementOf slot explicitly recorded on the Grebes frame, because there is a bioTaxonomicLevel slot sitting there, and because the frame for bioTaxonomicLevel says that it is a specialization of elementOf

Almost every step of our methodology loop entails adding new, specialized kinds of slots to the system, as needed, along with the other kinds of frames that get added. Let's consider a case where new kinds of slots were added to the system.

"Grebes have small wings." That sounds easy to represent. Given a Grebes frame, we could add any of the following slot/value pairs to it:

wingsize: Small
wings: (# - - - size Small)
partsSize: (Wings Small)
partsDescription: (Wings (size Small))

Unfortunately, there are actually many different things the sentence could mean. What is it, exactly, that is small? Is it:

- The length of a Grebe's wing, compared to the length of standard meterstick? (actualValue)
- What we'd expect their wings' length to be (compared to a meterstick), knowing that Grebes are aquatic birds, and the actualValue of the length of most aquatic birds' wings?*(expectedValue)
- What we'd expect their wings' length to be (compared to a meterstick), knowing the length of most other parts of a Grebe?* (expectedPartValue)
- The ratio of the length of a Grebe's wing to the length of most aquatic birds' wings? (relativeMagnitude)
- The ratio of the length of a Grebe's wing to the mean length of its body parts? (relPartMagnitude)
- The ratio of Grebes' wing-to-body-size to the wing-to-body-size for typical aquatic birds? (*I.e.*, the quotient of relPartMagnitude for Grebes' wings and relPartMagnitude for AquaticBirds' wings.) (relProportionPartsDescription)

In the case of our Grebes article, the sixth meaning was intended. Our language should - and does - permit each of these meanings to be represented in a separate but equally efficient manner. Essentially, there are six different relations being talked about, and we translate them into six separate kinds of slots. Each of these can exist for Grebes and can have many entries; in each case, one of the entries could be (Wings (size Small)). Each of the six slots can be considered a relation over four arguments (u i m v), where u is the name of a frame (in this case, "Grebes"), i is the name of a part (in this case, "Wings"), m is the name of a measuring or comparing function (in this case "size"), and v is the purported value (in this case "Small"). Each of the six slots has its own definition, a function that takes the first three arguments and returns the fourth one (the value v):

```
actualValue.    (m (i u))
expectedValue.    (m (i (basicKindOf u)))
expectedPartValue     (m (apply* (basicKindOf i) u))
relativeMagnitude:    (Quotient (actualValue u i m) (expectedValue u i m))
relPartMagnitude:    (Quotient (actualValue u i m) (expectedPartValue u i m))
relProportionPartsDescription:    (Quotient (relPartMagnitude u i m)
                                            (relPartMagnitude (basicKindOf u) i m))
```

One of our language design principles is to view "huge values" as a sign that we haven't divided up the world enough, or properly. Hence the proliferation of related slots, above. Another design principle is to view any piece of Lisp code in our KB as a transient, an intermediate value which has been computed and then "cached" from some declarative, descriptive knowledge.

For most slots, this means that their defn slots are to be thought of as virtual slots, which are computed from other slots. In the simplest case, these other slots are slotCombiner and builtFrom, and CYC applies the value stored in the slotCombiner slot to the value stored in the builtFrom slot. For instance, relativeMagnitude has a slotCombiner slot filled with the value CombineByRatioing and a builtFrom slot filled with the list (actualValue expectedValue).

---

*Average together the length of its head, neck, breast, back, legs, etc. The concept of "body parts of a bird" is very crisp, thanks to C. Sanders.

ment is warfare" analogy; *i.e.*, CYC has been told about this correspondence (or discovered it), has seen "bacteria" map to "enemy soldiers," "infection" to "invasion," "containment" to "containment," "drugs" to "weapons," has seen some medicine heuristics map to warfare heuristics and vice versa. CYC could then notice that certain medical concepts had no warfare analogues—at least, not yet—and suggest that they ought to get created.

For instance, suppose the knowledge base already contains a heuristic rule about the use of infiltrators to subvert and undermine the enemy's effort from within. If no medical analogue were known, then CYC could suggest creating one. Then, instead of requiring a person to create and edit the new rule, CYC could use its knowledge of the overall "medicine as warfare" analogy (of what mapped to what) to do most of the needed substitutions itself.

The two analogous items (in this case, heuristic rules) might be kept separate or merged into a common generalization, or all three items might be worth keeping in the KB. This last case is most likely, since each has some reason for continuing to exist. The generalization may be easier to apply to a new situation and may capture some deeper knowledge about the world than either special case. The special cases may have some idiosyncratic information attached only to them such as the relative success rate of the rule in each specific domain; the frequency of ignorance of the rule among practitioners in the field; the cost of applying the rule, etc. Another way of thinking of this is that each analogy is a useful but partial generalization (Lakoff and Johnson, 1980). The seven items (a-g) in Dimension 1, below, suggest even more clearly a spectrum between generalization and analogy.

Most analogies "work" because there is some common causality that led the analogues to be similar. Analogies are *useful* because often that common causality has resulted in many more shared attributes that haven't yet been noticed. We understand so little about the world that those common causes are often unknown or even undetectable by us directly; hence analogical reasoning is important and ubiquitous. This suggests that causal metaknowledge (justifications for values of slots) will play a key role in deciding how to find and extend analogies.

One or the other of the two analogous items is likely to be much better understood at the time the analogy is formed; often, in English, this leads to the practice of using *its* terminology for the common generalization as well. Nevertheless there is a full-fledged symmetry here, and every slot whose value exists for one frame but not for the other is a candidate for analogizing a new piece of information to the other.

Each medical heuristic (and object and operator) could lead to a warfare analogue and vice versa. Each *kind* of slot that exists in only one domain suggests an analogous kind of slot that might be worth defining in the other domain. For instance, the susceptibleToDrugs slot, and its inverse, effectiveAgainstBacteria, would map into brand new concepts, new kinds of slots, ones we might call, respectively, susceptibleToWeapons and effectiveAgainstEnemySoldiers

Then, as each of these new frames are created, CYC should use its growing model of the analogy to decide which substitutions to make in the frame body, instead of forcing the human (knowledge base builder) to make them all manually. For instance,[1]

```
makesSenseFor(susceptibleToWeapons) =
analogue(makesSenseFor
        (analogue(susceptibleToWeapons))) =
analogue(makesSenseFor(susceptibleToDrugs)) =
analogue(Bacteria) =
EnemySoldiers
```

In a similar fashion, the makesSenseFor slot of effectiveAgainstEnemySoldiers would be filled automatically with Weapons (*i.e.*, each kind of weapon has its own characteristic effectiveness against each type of enemy troop).

We do not believe the analogy process needs to be much more complicated than the structure-matching process that we (Lenat, 1984b) and others (Greiner, 1985; Mitchell, 1985; Bobrow and Winograd, 1977; Winston, *et al.*, 1983) have hypothesized and described. In its simplest form, analogy is the matching between values of corresponding slots of frames.

On the other hand, analogy is certainly not limited to finding identical values in two identically named slots of two frames. For instance, the two analogous frames might have very different levels of detail in the slots describing them. Consider a particular VLSI circuit, whose complete layout is known, and a particular city, whose layout is only vaguely known. Because the slots in CYC are part of its knowledge base, they are arranged hierarchically in a generalization/specialization graph. In particular, fullCircuitLayout is a specialization of vagueLayout Whenever the former slot exists for a frame, the latter slot exists virtually (it can be computed from the former, more detailed value; see Greiner & Lenat, 1980). So the analogy could be detected at the vagueLayout level, because that slot exists (actually or virtually) for both frames. This process relied on the slots existing in a hierarchy; one of our tasks is therefore to taxonomize slots and build that hierarchy into CYC's KB.

We agree with Amarel (1983) that the term "analogy" is a vague catchall obscuring dozens of more specific kinds

[1]Frequently, throughout this article, a call to get the value of the f slot of frame x is written in functional notation: f(x); this dual notation carries through to the current CYC program as well. Typically, the names of frames will be capitalized, whereas the names of slots will not be. A conflict arises when one wants to discuss the frame that talks about the elementOf kind of slot; in those cases, we also choose to leave the name uncapitalized.

of phenomena, each of which is worth distinguishing[2] and studying. Even our initial inspection turned up several different dimensions along which one could profitably classify analogies. "Profitably" here means that different, specialized heuristics apply at each point of each dimension.

The four dimensions below define a four-dimensional matrix, and for each point in this "analogy-space" a unique set of heuristics holds, heuristics for how to find analogies in that region, when it is and isn't worth looking for them, when and how to extend an existing analogy, when and how to use an analogy, etc. Think of this matrix as a start in meta-analogical reasoning: reasoning about when and how two analogies are similar to each other.

Dimension 1: Variations in the level and degree of matching.

(a) Two frames have several slots with identical names and values. For instance, Irrigating and Raining share:

basicKindOf Transporting
transportee Water
destination Ground
recipients Plants
aidsRecipientIn Growing

(b) Two frames can have identically named slots whose values are not quite identical; the resultant mapping becomes part of the analogy. Irrigating and Teaching are analogous in this fashion, since Teaching has a recipients slot filled with Students, and an aidsRecipientIn slot filled with Learning. This is the analogy that teachers distribute knowledge to aid their students in learning, just as irrigation systems distribute water to aid plants in growing. This analogy will, in turn, either suggest the "Learning as Growing" analogy, or, if it already exists, be strengthened by it.

(c) Two frames can have matching slots whose names are not identical, but whose values are. For instance, "Juries" has NumberofMembers 12 and "Doughnuts" has soldBy 12. Not all analogies appear useful, unless one has been sequestered for a long time.

(d) Both the names of the slots and the values they contain may match but be nonidentical. An example is the bioVector slot for bacteria and the troopTransport slot for troops. This is handled naturally by CYC because the kinds of slots are part of our (hierarchical) KB. In this case, both of those slots are specializations of transportVehicles, which makes sense for Transporting in general.

(e) The matching can also be at a higher level: two aggregates of frames match each other. The "Medicine as Warfare" analogy involves dozens of frames from each domain, including objects, operators, heuristics, and so on. This is more the rule than the exception in practice.

(f) The matching can also be at a lower level: the two entities that match each other may merely be properties about the values of two slots of two frames. For example, two stocks on the NY Stock Exchange may be analogous because their prices fluctuate wildly; in our terms, the values in the price slot of the $Stock_1$ frame and the $Stock_2$ frame have different and uncorrelated values most of the time, but both of those slots have the same "extra" property/value—namely volatility High.

(g) In some analogies, the two analogues coincide. For instance, it may be worth conceptualizing the match between two parts of the same frame, say the physical and mental attributes of a person. At another level, there may be a match between the volatility property of all the (values of the) physical description slots of an object (*i.e.*, for most objects, either all the physical attributes are all stable over time or else many of them are volatile).

Dimension 2: Nature of the boundary, i.e., *the kinds of places where the analogy almost breaks down, or just barely does break down.* For instance, some analogies share the property that using them can be a powerful heuristic method in one situation and yet be a fatal trap in nearly identical situations (*e.g.*, "nuclear weapons are like conventional weapons"). Other analogies are similar to each other in that they fail less abruptly at their boundaries. We expect that analogies that differ along this dimension will have some specialized heuristics about when and how to use them—and not use them.

Dimension 3: The particular nature of the analogues (domain-specific processes, objects, heuristics, etc.), the kinds of domains they come from, and the particular identities of some of the analogy pieces. What we're saying in the first case is that an analogy between two processes probably has some special heuristics for finding and using it, compared with analogies between two static objects. The second case says that there may be some special heuristics that apply whenever you analogize to medical domains, to physical processes, or to computer science. The third case goes even farther: it says we distinguish some very special cases. An example of one is when the

[2]Given Brachman's article in the last issue of this magazine, it is worth mentioning when it is (and isn't) worth distinguishing two concepts, *i e*, making two separate frames out of them. Such distinguishing should be done for those, and only those, concepts for which some specialized, powerful heuristics exist and can be brought to bear Thus, "one-legged-sushi-loving-programmers" is not worth distinguishing unless there's more one can say about such individuals than one would inherit individually from one-legged people, sushi-lovers, or programmers

difference between the two analogues is mainly one of *time* having gone by; in such a case we rarely think of the match as analogy at all, calling it instead something like continuity, aging, or stability. When the difference is one of the nontemporal situational variables, we again think of this as something other than analogy, namely flexibility or generalization. There are no doubt many more such special cases to be found and organized; even the two just mentioned need to be much further refined.

Dimension 4: The context for the analogy. This is a way of saying that the current goals, resources, audience, recent performance, *etc.*, all contribute to the way in which we find and exploit analogies. So two analogies lie close to each other along this dimension iff they were found in, or are being used in, similar extended contexts. This distinction was useful in getting programs to do goal-directed learning (Mitchell, 1983). As with all the other dimensions, when this dimension is further refined, we expect different heuristics to apply at various places along it.

We expect to need a taxonomy not only of the types of analogies, but of the types of analogical reasoning as well. These include finding analogies, extending them, importing them to new areas, suspecting misleading analogies, restraining overzealous ones, assimilating new ones, and repairing old ones.

The final wrinkle we'll mention here is that the precise way two concepts are represented can radically effect how easy it is to find the analogy between them; this is discussed more fully in Lenat and Brown (1984). What this means for CYC is that our representation language must expand in ways to facilitate analogical reasoning, not merely in ways to shorten the expression of facts (in other words, elegance is its own reward).

Even copy&edit in its present, 100 percent manual form, is highly cost-effective, saving expert system KB builders a large amount of time. CYC's two analogy-driven extensions (automatically suggesting new concepts to copy, and helping to automate the new concepts' editing) should be an even larger step in that same direction.

Functionality of the CYC System

A very important part of the CYC knowledge base is a large, organized body of reasoning methods. These are being described declaratively in CYC in a network of frames spanning both problem-solving architectures (*e.g.*, exhaustive backward chaining) and specific heuristics. The methods also include ways of operationalizing methods (*i.e.*, actually executing them), and CYC has "cached" efficient procedures for many of the common methods. Included in this part of the KB are analogical and common sense reasoning methods, as well as more traditional problem-solving techniques.

The foregoing sections have illustrated only a few of the functions that a large real-world KB could be used for, given the "methods base" just described. CYC holds the promise of a qualitatively new kind of reference work (Lenat *et al.*, 1983), the improvements coming from the dynamic nature of the interaction, the multiply connected nature of the knowledge (rather than linear text), the three-dimensionality and aural "soundspace" that aid in navigating and apprehending, and the custom-tailored nature of the experience: filtering through individual user models; organizing material based on the individual user and his/her current purpose, rather than based on traditional article boundaries and sequencing. This leads to a kind of exploring of knowledge space (by humans). A weaker and probably sooner-realized form of this is question-answering. If the system is given more of the initiative, this also merges into ICAI, although to be an effective teacher, the KB would have to include a great amount of additional material (models of the human learning process, how to diagnose and repair students' models of a subject, useful metaphors for teaching specific lessons, etc.).

In a standard expert system, the rules refer to technical terms that are related to each other in various ways but are not "tied in" to a global model of the real world. What we propose is precisely that tying in.

CYC may also be able to boost AI performance programs' ability to solve problems: semantic disambiguation for natural language programs, guessing at possible new "Unless conditions" for expert system rules, and so on.

Earlier, we discussed how analogy-finding could be done in the context of a specific problem that must be solved. But it can also be pursued "offline" by the system, more or less for its own sake. One might run CYC overnight, each night, as was done for Eurisko (Lenat & Brown, 1984), having it discover analogies of intrinsic interest to the user, meanwhile building up an ever larger bank of analogies for possible future use.

How CYC Works

This section illustrates how general real-world knowledge might aid expert systems. In this example, a rule is entered with its "unless" conditions absent, and CYC then fills them in automatically. This could be done either to improve knowledge acquisition (do it when the rule is first entered) or to improve performance (don't do it until the expert system gets "stuck" and needs common sense knowledge to guess what sort of exceptional case it's in the midst of).

Before we see how CYC might help, let's see what the problem really is. In a standard expert system, the rules refer to technical terms that are related to each other in various ways but are not "tied in" to a global model of the

real world. What we propose is precisely that tying in. Consider the following dialogue between a human (called "Expert") trying to build an expert system and a conventional AI expert-system-building shell (called "TOOL"). The $ signs indicate variables.

```
TOOL:   Please type in the next rule.  IF:
Expert: AskedInInterview
        ($doctor,$patient,''Do you have $x?'')
        =Yes
TOOL:   THEN:
Expert: Has($patient,$x)
TOOL:   Thank you.
        Here is my rephrasing of the rule in English:
        If doctor AskedInInterview patient
        ''Do you have x ?''
            and result was Yes,
        Then conclude patient Has x.
        Is this correct?
Expert: Yes.
TOOL:   Good.  Please type in the next rule.  IF:
.
.
.
```

Now nothing is actually "wrong" with the rule, and the expert may continue to enter hundreds of additional rules that help the system diagnose various kinds of medical problems. The system appears to work fine on many cases, even though it does not really understand much about symptoms, disease, doctors, patients, or interviewing people.

As various cases are run, some mistaken diagnoses may be tracked to this rule, and each one may result in a new clause being added to its If part; *e.g.,* what if $x were the kind of thing someone might lie about, such as "a history of insanity in your family?" Indeed, many expert system languages (Petrie, 1985; Winston *et al.,* 1983) provide for explicit "Unless" conditions to be added to each rule. Unless conditions are *logically* equivalent to (negated versions of) clauses one could conjoin to the If parts of rules, but it is worth keeping them separate for several *pragmatic* reasons: Unless conditions are learned more cheaply, they can be added and modified with less effect on the rest of the system and they're less important to teach neophyte performers early on.

When actually coding the new Unless condition referred to above ("unless the patient might lie about $x"), the kinds of things someone might lie about would have to be listed explicitly (insanity in the family, use of illegal drugs, country of birth, etc.). Here is an opportunity for the knowledge base builder to accidentally overlook some items. As new embarrassments (*e.g.,* herpes, AIDS) and societal mores (*e.g.,* abortion, homosexuality) come and go, the rule would have to be updated.

We've now begun to get at the real problem. It comes down to a *knowledge illusion:* The rule appeared to make sense to us, but only because we as human beings already knew what the various terms it mentioned all meant: doctors, patients, intake interviews, and symptoms. All the expert system knows about AskedInInterview is that it is an operator, a Lisp function it can call, which returns one of the values Yes, No, and Unknown To the expert system, it is not much different from the various lab test operators, for instance, such as GramStainTest and AerobicityCheck. Consider the rule from the point of view of the expert system. It might just as well have been written:

> If the result of F083(A,B,C) was D,
> Then conclude F002(B,C).[3]

When recast in the harsh light of GENSYMs, the limits of expert system "common sense" become all too clear.

How can CYC help? After the session in which the rule was added, CYC would ask the expert to explain, in turn, each of the unknown terms in the rule: AskedInInterview, $doctor, $patient, and $x.

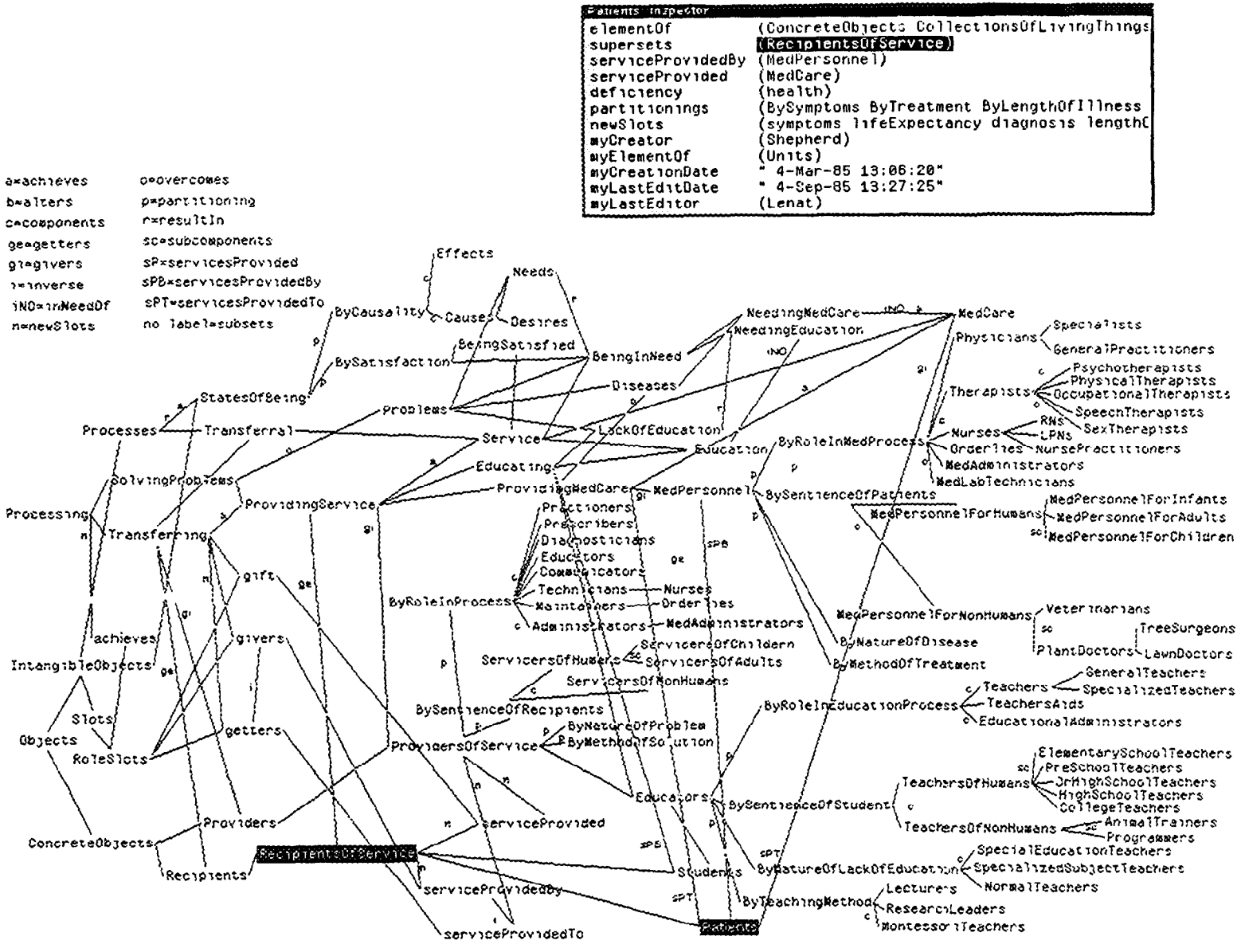
CYC starts by requesting, "Please help explain what the AskedInInterview operator is all about." In response, the user goes through a rapid search through the knowledge base, from Anything, to Processes, to MentalProcesses, on down through Communications, all the way to Querying. This is implemented graphically at present and will take only a few seconds to perform. The user needs only to glance at the log of the number of frames in the system.

CYC now asks, "What set is the variable $doctor intended to range over?" The user finds the relevant frame (Doctors) and points to it. Similar interactions occur to find that $patient ranges over Patients, and $x over Symptoms.

If this is the tenth medical diagnosis rule being entered, probably all of those sets exist already. But what if, say, Patients wasn't known to CYC yet? The expert would discover this by arriving at the place where Patients should be (*e.g.,* kinds of RecipientsOfService) and not finding it there. At that moment, s/he enters a screen editor, to record several facts about a brand new frame called Patients. S/he chooses to copy it from Students, which eliminates the need to type in the facts that Patients are People and also are RecipientsOfService; it incidentally defines a weak but intriguing new analogy, namely "medicine as education."

---

[3] We're willing to admit that the program might understand "the result was," but will not grant that it understands the meaning of "Has." There are dozens of things that the average five year old child knows about "having a symptom" and "having a disease" that no current medical expert system knows. See McCarthy (1983)



Entry of Patients frame into the knowledge base.

Figure 1.

Now the expert manually adds two more slots to the Patients frame, representing the facts that the service being received is medical care and that, by definition, something is wrong with the health of the typical patient. All these related concepts—People, RecipientsOfService, typicalmember, MedCare, health, NecessaryDefiningSlots—are concepts CYC knows. That is, they are the names of frames it already has in its KB. They in turn participate in relations with many other frames representing sets (Doctors, Symptoms), attributes (mentalHealth, physicalHealth), and scripts (ProvidingService).

The expert makes most of these connections by pointing to related frames, as s/he navigates through "knowledge space." This navigation can be implemented simply by a Zoglike network of menus or less simply by using three-dimensional graphics, joysticks, helmets, and even less simply by employing artificial personae as guides. At present, we are using an Inspectorlike (Interlisp Reference Manual, 1983) frame editor to display each frame in detail, plus a Grapherlike network of "mouse-buttonable" nodes showing only a few kinds of links among them at any one time. Buttoning a node causes it to be Inspected (and thereby made editable). Figure 1 is a snapshot of two of the windows on the screen while the Patients frame is being entered. Near the top is the Inspector's display of what's known about Patients; below that is the Grapher's display of some "nearby" nodes that the enterer can browse through and copy.

The difference in the two dialogues—without CYC and with it—is that CYC doesn't let the expert get away with ambiguity and jargon, or with new terms it doesn't understand. It continues to ask about all the various unknown terms and variable names, forcing the expert to describe and explain them, until s/he finally is using only terms that CYC already understands (such as Goal, Querying, BodyPart, LogicalDeduction, Remove, Weapons, Symptoms, HasAsAPossession, HasAsAnAttribute, or Transporting).

This seems to make a fair amount of extra work for the knowledge base builder. We have claimed that it more than compensates for this. To see how this is so, we need to peek at some of what CYC knows (see Figure 2).

Our point is to show that CYC can use the above knowledge to automatically compute many of the Unless conditions for our rule, conditions we would otherwise have to get by introspection or hard experience. In other words, these Unless conditions are implicit in the knowledge base, specifically in the constraints slots of certain frames. How does CYC locate this relevant knowledge and transform it into explicit Unless conditions?

We simply ask CYC for values of the allWaysItFails slot of AskedInInterview. This is typical of the way in which computation is initiated in CYC: calling for the value of a particular slot of a particular frame. Even a complex task, such as proving a difficult theorem, would be initially posed in this way, calling for the *proof* slot of the Theorem87 frame. An important "rule of operation" in our system is not to give up just because a primitive access (such

Communicating
basicKindOf: Informing
actors: (communicator a) (recipient b)(message c)
action: (InSequence (OpeningCommunications a b)
(Telling a b c)
(RecallMatching b c)
(ClosingCommunications a b))
constraints: (CanUnderstand b a c)
(Desires a (Communicating a b c))

2WayCommunicating
basicKindOf: Communicating
action: (Repeatedly (Communicating a b c_i)
(Communicating b a d_i))

Querying
basicKindOf: 2WayCommunicating
actors: (question c_i) (declaration d_i)
constraints: (ResultOf d_i (Remembering b c_i))

AskedInInterview
basicKindOf: Querying

Remembering
basicKindOf: Informing
actors: (rememberer a)(memory b)
constraints: (Accurate (Remembering a b))
accuracy: (# - - Moderate variance High)

allWaysItFails
defn: (Lambda (u) (MapUnion
(fullAction u) 'waysItFails))
elementOf: InheritedSlots UnlessFillingSlots
builtFrom: fullAction waysItFails
combiner: MapUnion
makesSenseFor: Processes

waysItFails
defn: (Lambda (u &rest) (negate (constraints u)))
elementOf: DerivedSlots UnlessFillingSlots
builtFrom: constraints
combiner: negate
usedInBuilding: allWaysItFails
makesSenseFor: Processes

constraints
elementOf: ScriptSlots PrimitiveSlots
usedInBuilding: waysItFails
makesSenseFor: Processes

A few pieces of general CYC frames, plus the one fact known about AskedInInterview. Figure 2.

as (GETPROP 'Theorem87 'proof)) fails; rather, the rule[4] says to examine the definition of the slot involved (proof) to see if there is some way of calculating its current value. If so, that procedure is applied to the frame (Theorem87). So calling for a slot of a frame can start a cascade of additional processing, which might take minutes (or longer!) to terminate. This is one reason why our language enables extra arguments to the Get function: to implicitly or explicitly restrict and guide the computational resources that should be brought to bear before giving up.

In the current case, CYC looks at the AskedInInterview frame and sees that there's no slot there labelled allWaysItFails, at the moment, there's no slot at all on that frame, except a basicKindOf slot that points to Querying—this is the fact that the user told CYC in the above dialogue, in response to the question about what AskedInInterview was an instance of. CYC doesn't give up at this point; it uses the "rule of operation" we discussed above and consults the defn slot of the frame called allWaysItFails. There it finds that it can compute the desired value by combining the (values of the) waysItFails slots of all the specific subactions that come together to form the fullAction, or script, for AskedInInterview. The frame for fullAction specifies how it can be inherited and pieced together from the various generalizations of a frame: one appends their action slots after doing the appropriately composed variable substitutions (as specified in the actors slots along the way). In the present case, the search will assemble together pieces of action from AskedInInterview, Querying, 2WayCommunicating, Communicating, Remembering, Informing, and on up to even more general frames not shown above (MentalProcessing, Processing, Anything).

After assembling this large fullAction script for AskedInInterview, CYC finds that none of these frames has a waysItFails slot! But instead of giving up, our "rule of operation" comes into play again. CYC consults the defn slot of the frame in the system called waysItFails There it finds it can compute such values by negating the value of the constraints slot for each of those frames. Some of them, namely Communicating and Remembering, do have nonempty constraints slots listed. Others don't, and CYC consults the constraints frame to see if it has a defn from which constraints can be computed; since there is none, it quits at this point. The value returned is therefore the union of the negations of the constraints it did find:

```
((NOT (CanUnderstand patient dr query))
 (NOT (Desires dr (Communicating dr patient query)))
 (NOT (CanUnderstand dr patient
   (Remembering patient query)))
 (NOT (Desires patient(Communicating patient dr
   (Remembering patient query))))
 (NOT (Accurate (Remembering patient query)))
 (NOT (Accurate (Remembering dr
   (Remembering patient query)))) )
```

In English, here is the meaning of those expressions, the Unless conditions CYC would propose:

> The patient didn't understand the doctor's question;
>
> The doctor didn't really want to ask the patient the true question;
>
> The doctor misunderstood the patient's reply;
>
> The patient didn't really want to tell the doctor the true answer;
>
> The patient mis-remembered the answer;
>
> The doctor mis-remembered the significance of the patient's answer.

Actually, many more Unless conditions might turn up; we haven't shown enough of the knowledge base in the above example to generate the compleat fullAction of AskedInInterview. For instance, OpeningCommunications might have failed because the patient was deaf and never heard the question, and the doctor mistook silence for an "I don't know" answer. Telling might have failed because of poor pronunciation or background noise. We used the archaic spelling of "compleat" here, because we believe that no objective value can ever claim to be the "full script" for a complicated real-world action. We could always delve one level deeper in each subaction. In our present case, that means that more and more Unless conditions could be generated, though they gradually would become less and less likely events.

To build CYC, we must encode all the world's knowledge down to some level of detail; there is no way to finesse this.

Some general knowledge can be used to prune as well as generate Unless clauses. Since patients are the RecipientsOfService from doctors, the second Unless clause in the above list can be reasoned to be unlikely (the one that said the doctor might have intentionally misled the patient).

This incremental style of reasoning is somewhat different from the normal von Neumann model. CYC returns an answer and begins using it, but meanwhile continues

[4]This is not implemented as an If/Then rule in CYC Rather, it is a function stored "in the proper place," namely, in the toGet slots of frames describing each kind of slot that follows this rule of operation (such as proof). In most cases, this value itself would be inherited from a much more general frame (*e g*, ComputableSlots) Finally, even there the function would just be a superfluous caching of a more declaratively specified piece of information stored in a frame describing a problem-solving method

looking for more accurate answers, which can thus appear at much later times. CYC must explicitly manage its processing resources, deciding when and how to allocate background processes to carry on—and terminate—such "incubation" searches. This use of concurrent processes is analogous to pipelining: The later values are only rarely so much at odds with the quickly-found ones that a backtrack is required. Techniques of nonmonotonic reasoning can be brought to bear on maintaining dependencies, so as to preserve as much as possible when a contradictory later value does arrive.

This is the kind of thinking that makes doctors robust problem solvers compared to current expert systems. The stream of Unless possibilities can be mechanically generated by the program because it has tied previously isolated terms, such as AskedInInterview, into a global knowledge base of general real world information. General knowledge about human communications and human memory were thereby brought to bear on the problem.

Building the CYC System

This project is based on a series of assumptions and hypotheses that we enumerate below. They have led to a methodology, a set of tactics for carrying out the project, which we discuss following the listed assumptions.

Underlying Assumptions

Assumption: To build CYC, we must encode all the world's knowledge down to some level of detail; there is no way to finesse this. Fifteenth century explorers couldn't discover new lands without long voyages. The breadth of our task may sound staggering, but consider that a one-volume desk encyclopedia spans that same magnitude of information.

Assumption: We are not talking about assembling a large textual data base of facts; entries in the knowledge base are so heavily interrelated that they are nothing more than their set of interconnections. Also, much of that knowledge is knowledge of how to do things, ranging from cached efficient algorithms to high-level descriptive scripts. Entering knowledge into CYC means that we must represent the knowledge in such a form that the program can find and use it when appropriate. For this reason, simply having an online version of an encyclopedia would be of little use, as there is practically nothing that current AI technology could draw from the raw text. Rather, we must carefully re-represent the encyclopedia's knowledge—by hand—into some more structured form.

Assumption: AI has for many years understood enough about representation and inference to tackle this project, but no one has sat down and done it. It is at once an extremely humble and an extremely brash hypothesis. To take the edge of humility off it, we remark that only by pulling together the latest human interface tools, Lisp machines, ideas of enforced semantics, and funding for a decade-long effort could we attempt a project of this scale. To take the edge of brashness off the hypothesis, we remark that we may fail in some ways (*e.g.,* our current ideas about how to reason by analogy may not "scale up"), but we expect that in the process we will learn some powerful new lessons about building large intelligent programs.

Assumption: Common sense knowledge is just the knowledge (facts, procedures, representations, heuristics) we employ most frequently. Therefore it includes specialized knowledge about human-sized and human-velocitied objects moving in space and time, human artifacts moving in space and time and doing their job, and humans interacting—and developing—socially, culturally, politically, militarily, economically, scientifically, etc.

Assumption: As the first few hundred most mutually distinct types of articles are encoded, we will encounter a typical representative of each fact that will ultimately be a part of the finished CYC knowledge base. After those articles are entered, simple Copy&Edit will be an adequate, cost-efficient method to add the final 99 percent.

Assumption: Most of the common sense will also emerge during our coding of the first few hundred mutually distinct articles. When a knowledge enterer gets to "Panthers," there will be little or no common sense required to understand the article that wasn't already required for "Lions" (or some other representative article, such as "Guerillas").

Assumption: The sentences in encyclopedia articles provide enough structure and context to enable the extraction of their underlying common sense. Introspection currently serves us as an adequate tool for extracting the common sense knowledge, so long as we work sentence by sentence. As the KB grows, the existing framework of common sense knowledge should facilitate this process. The system's hierarchy of problem-solving methods should eventually help, as well.

Assumption: Similarly, there needs to be little time spent agonizing over the ambiguities and contradictions in the sentences. Introspection (plus the growing KB itself, especially its methods) will work instantaneously most of the time, and whatever uncertainties and disagreements remain can be explicitly represented in CYC.

Our plan is to carefully represent approximately 400 articles from a one-volume desk encyclopedia.

Assumption: Parallelism is not a panacea. It is an alternative, usually not a good one, to knowledge directed (best first) search. A small amount of parallelism is probably usable, and even a highly parallel machine architecture

is not antithetical to our system: allotting each frame, or even each slot of each frame, its own processor; allowing small sets of frames to "caucus" in private areas. Nevertheless, we are not actively pursuing any of this research at the present time. When time becomes a factor, when the system runs too slowly to be useful, and knowledge-based guidance (heuristic rules and metarules) fails to correct the problem, we will turn to parallelism. Hopefully, by that time, AI will have learned enough about its use that we'll be glad we waited.

*Assumption: One potential problem is coordinating a large team of knowledge enterers, so that they employ a shared semantics (*e.g., *for the meaning of each kind of slot); we believe this problem can be managed by enforced semantics.* Implicitly, this occurs as people copy existing structures rather than try to come up with ways to organize things on their own. Explicitly, we have our language state the semantic constraints. For example, every rule's ifMaybeRelevant slot is supposed to be faster than and logically implied by its ifTrulyRelevant slot. This can be checked dynamically, and if someone is misusing those slots then the "proper" use can be explained to them. Another semantic constraint is that each *example* of a frame must satisfy the frame's *definition.* A third such constraint is that a frame's *definition* must be implied by the *definition* of every entry listed on the frame's generalizations slot. Hundreds, eventually thousands, of such explicit semantic constraints work together, much as rules in an expert system, to advise the knowledge enterer when and how he or she is misusing the commonly accepted meaning of the various slots. Knowledge about the slots is a full-fledged part of CYC's knowledge base, hence they are organized and interrelated in dozens of ways. This not only enables explicitly enforced semantics but also enables the knowledge enterers to quickly locate the "right" slot to use (or to see that it doesn't exist and needs to be defined).

Assumption: Another potential problem is the increasing difficulty of finding the right existing frame to point to (or type in the name of) as one is entering a new piece of knowledge; we believe that this will require some human interface innovation, but not turn out to be insolvable. We expect that most of the knowledge entry will be done by helicoptering around "knowledge space," pointing at objects and substructures to be tied together. Although this is just a graphical extention of current-day text-based copy&edit, we have already found text editing and conventional graph-editing tools, insufficient for easily extending the growing CYC knowledge base. We are adding raw graphics power (an IRIS interface, a helmet-mounted display, and orientation-sensitive Palhemus sensors for fingers), but that alone may not suffice to keep humans oriented as the knowledge space keeps growing. We expect that some human engineering of the interface, however, will keep this problem tractable. We are considering: always displaying certain parts of the neighborhood of the "current" frame; maintaining menus of recently visited or frequently useful landmark views; using heuristics to choose what to display, and where; shifting to other metaphors besides piloting, such as mapping frames as rooms, slots as objects in the room, editing as redecorating, frame creation as building; and having fictional personae act as guides, advisers, and agents.

Assumption: Once built, the CYC system could form a common language or foundation upon (within) which future AI programs could be written. Even if we don't get the representation to suit everyone (a Pareto triviality), each researcher and engineer may still judge that it's cost effective to build within the CYC framework because of the already large (and thus, in a sense, large enough) knowledge base that will exist there. This is a positively-reinforcing critical mass assumption. CYC is not being built to demonstrate new AI theory; it is conceived by us as a beneficial juggernaut, but a juggernaut just the same: its size and breadth and usefulness serving as irresistible lures to its use, and those uses serving in turn to enhance CYC's size and breadth and usefulness.

Methodology

Our plan is to carefully represent approximately 400 articles (about 1000 paragraphs worth of material) from a one-volume desk encyclopedia. These are chosen to span the encyclopedia and to be as mutually distinct types of articles as possible. We will then bring on board a large number of knowledge enterers to add the final part (the last 99 percent—about 30,000 articles) by using copy&edit.

Phase I: For each of the 400 mutually distinct articles, we go through the following loop.

1. Represent the stated information in our representation language. During this step we must disambiguate what the writer actually meant. Also, we may need to expand our language to handle the new information to be represented, though this should be less and less frequent each time we repeat the loop.
2. Move each frame/slot/value "fact," up to the most general frame for which it's valid. If that frame isn't already in the system, create it now.
3. For each fact F, write down the additional facts about the world that are needed to understand F These are common sense facts that the writer of the article presumed the reader already knew. Now repeat steps 2 and 3 on this new set of facts. This step may entail creating new frames. The careful reader will notice that it might also lead to an infinite regress, as each common sense fact gets "explained" by a set of new common sense facts. In practice, this regress ends rather fast, "bottoming out" in primitive facts about human motivation and economics and everyday physics.

4. For each adjacent pair of sentences in the article, write down—and encode into our language—the additional facts about the world that are needed to understand what transpired "between" the two sentences. Now repeat steps 2 and 3 on this new set of facts, which may entail creating new frames.
5. Throughout the above four steps, collect a set of questions that ought to be answerable and problems that ought to be solvable, based on having read the article (plus "common sense"). If CYC errs on one of them, that points out a lack of either some common sense fact or some general reasoning method. In either case, add it. Occasionally, go back to articles encoded earlier, and ensure that their set of common sense problems and questions still get answered correctly.

Phase II: After the initial 400 articles are encoded and the representation language settles down, we converge on the KB.

1. Employ a large cadre of (10-50) knowledge enterers to encode the final 99 percent of the knowledge base. Each enterer will take an article, locate the already-represented similar article(s), and perform a machine-assisted "copy & edit" procedure to produce a machine-understandable version of the new article. For example, to enter the knowledge about Britannia-Metal, the enterer would copy the closest existing article already represented, say Pewter, and then edit that structure to reflect the differences between the two alloys This is the analogue of the source of power tapped by the TI NLMenu (Ross, 1981) system: giving users choices rather than having them type in things "from scratch " It is one way of implicitly enforcing a common semantics.

2 As this long phase proceeds (1988-93), test out the system by hooking it to—or, more likely, building within it—various particular AI programs: expert systems, natural language understanders, and problem solvers During this period we will examine other sources of knowledge (such as childrens' stories, Tell Me Why books, newspapers, fictional short stories), and will hopefully enter a sample of each type.

3. *Systematic* entry of that level of knowledge is beyond the ten-year scope of this project and could occur during the mid- and late- 1990s, if the project is a success. That would also be the time when many expert systems could "hook in," and during which the CYC knowledge base would be adapted into products: knowledge utilities akin to electric utilities, advanced entertainment and art tools, the next generation of CAI systems, and so on. As more and more expert knowledge is accessable to (processable by) CYC, it should be able to exploit analogy more and more effectively for acquisition and for problem solving

An Example of the CYC Methodology

We have looked at scores of articles in several encyclopedias by now, but here is the first article we looked at and to which we applied our methodology.

> **coke,** hard, gray, porous fuel substance with a high carbon content. It is the residue left when bituminous coal is heated in the absence of air. Coke is used in extracting metals from ores in the blast furnace
>
> —*Concise Columbia Encyclopedia,* 1983, p. 180

Almost any article could have been used as the initial article, as it was bound to be a representative of some class of articles. Analyses of articles, begun in Lenat *et al* (1983), provided the approximate figure of 400 for the number of types of articles. The intent is that two articles are of the same type iff we could reasonably expect a knowledge base enterer to add the second article relatively easily using copy&edit from the first one. Thus, Lions and Tigers are of the same type, but no two of these are of the same type: Lions, ForwardChaining, AnimateObjects, Cheese, Cheesemaking. Our notion of "type" is a vague measure, to be sure, but we have not needed to revise it much even after several months of further work.

Step 1: Encode the Explicitly Stated Factual Information.

This resulted in the following frame being created and filled in. (See Figure 3.)

Coke
basicKindOf: SolidFuelSubstances
color: Gray
ingredients: (# - - (Carbon relAbundance High))
hardness: High
porosity: High
usedAsInputTo: Smelting
createdAsResidueIn: Coking
myCreator: Shepherd
myCreationDate: "2-Feb-85 9:57:10"

Figure 3.

The value stored in the ingredients slot says that one of the ingredients of Coke is Carbon and, moreover, that coke is mostly carbon. We subsequently found out more about coke's composition, and developed a vocabulary of connectives that let us describe most mixtures: InDecreasingAbundance, InIncreasingAbundance, InNoParticularOrder, Absolute%, InApproximatelyEqualAmounts. Using these connectives, we could concisely express the fact that coke is 98 percent carbon, and the remaining 2 percent is mostly metals, with small amounts

Coking
elementOf: ChemicalProcesses IndustrialProcesses
inputs: BituminousCoal
operators: Heating
residues: Coke
constraints: (none-present air)
(greater-than temperature 550)
action: apply operators to inputs under constraints

Smelting
elementOf: ChemicalProcesses IndustrialProcesses
inputs: MetalOre Coke
operators: Heating
residues: Metal
locus: BlastFurnaces
action: apply operators to inputs under constraints
applicationOfProcess: ReductionOfMetalOxide

Figure 4.

Coke
basicKindOf: SolidFuelSubstances
color: Gray
typeOfChemComposition: Mixture
ingredients:
(Absolute% (Carbon 98)
(InDecreasingAbundance
Metals
InNoParticularOrder
OrganicCompounds
TraceElements)))
relativeMagnitude: (hardness High) (porosity High)
usedAsInputTo: Smelting
createdAsResidueIn: Coking
MyCreator: Shepherd
myCreationDate: "2-Feb-85 9:57:10"
lastEditor: Lenat
lastEditedDate: "3-Feb-85 18:22:56"

Figure 5.

of organic compounds and other miscellaneous trace elements. We put off the phrase about "heating bituminous coal in the absence of air" into a separate, named frame, namely the one for the process called Coking. We similarly separated and identified the extraction of metals from ores in a blast furnace as the process called Smelting. We then sketched in those two frames as shown in Figure 4.

By now we had dozens of additional frames to add, namely one for each kind of slot (*e.g.,* "applicationOfProcess," "hardness") and one for each concept named inside a value so far (*e.g.,* "Heating," "BituminousCoal").

While trying to answer questions (step 5) using the hardness and porosity slots, we noted that they are *relative* attributes: Coke is hard compared to coal, say, but not compared to diamonds or steel. Coke is *relatively* hard, compared with (most) other solid fuel substances. At this stage, we replaced hardness and porosity by two new kinds of slots, relHardness and relPorosity, whose values used the # notation to capture the notion that it is SolidFuelSubstances with respect to which Coke is hard and porous. This later got subsumed by the relativeMagnitude slot. The Coke frame now looked like Figure 5.

Step 2: Move Information to More General Frames.

In this subsection, we list three criteria for deciding that a piece of information might merit being moved from its current position to one on a more general frame. Then, we show examples of when and how we did this for Coke (and related frames).

The most obvious signal is when information appears to be duplicated time after time, frame after frame. Even in the few snippets of frames we've listed in the section above, there are duplications (*e.g.,* the action slot of both Coking and Smelting). These are clear signals that something more general is going on here.

A second signal of this kind is the use of deep constructs. In the ingredients slot of Coke, the deepest entry (namely, that traces of miscellaneous elements exist in the mixture) is probably true for a large collection of mixtures used in the real world. Perhaps it ought to be a default ingredient for Mixtures; *i.e.,* if you (or CYC) had to guess, those trace impurities are probably present in most mixtures, or else it would be worth noting that exception explicitly.

The third signal that some information ought to get inherited from somewhere (*i.e.,* ought to now be moved from its present position to a more general frame) is that the English language version of the fact is much terser than its current encoding. For instance, saying that "Coke is hard and porous" took up a large fraction of the symbols used in encoding the Coke frame at one time, namely the two huge slots relHardness and relPorosity which explicitly recorded the fact that coke's hardness and porosity are high relative to the set of solid fuel substances. Almost all of that went away when we switched to the relativeMagnitude slot.

The same kind of generalizing is done for each piece of knowledge that is signaled to be moved. For instance, in the Smelting frame, the action slot is fully inherited from ChemicalProcesses; most of what we initially entered for that frame eventually was moved farther on up to Processes. The values of Smelting's operators and residues slots are fully inherited from ReductionOfMetalOxide, as is one of the values of its elementOf slot. Thus the Smelting frame now is shortened to:

Irrigating
energySource: Pumps Gravity CapillaryAction
genlProcesses: CommercialWaterSupplying WaterConduction
transportVehicle: Pipes Ditches

Transporting
specProcessess: Conduction TransportationOfDiscreteObjects
genlProcesses: MechanicalProcesses Transferring
achieves: (Deprive (a Actors with location = departurePoint) Of transportee)
(Supply (a Actors with location = destinationPoint) With transportee)
action: (InSequence
(Inserting transportee Into departureReceptacle)
(InAnyOrder (Transferring transportee Into transportVehicle)
(Enabling energySource))
(InCausalOrder (Applying energySource To transportVehicle)
(Changing (location transportee) To
(location destinationReceptacle)
(Transferring transportee Into destinationReceptacle))
actorConstraints: (elementOf departureTime Times)
(elementOf arrivalTime Times)
(earlierThan departureTime arrivalTime)
actors: transportee transportVehicle departurePoint energySource
departureReceptacle destinationReceptacle
departureTime destinationPoint arrivalTime
constraints:(Nearby destinationReceptacle destinationPoint)
(Nearby departureReceptacle departurePoint)
(Can (Contain transportVehicle transportee))
(Can (Contain departureReceptacle transportee))
(Can (Contain destinationReceptacle transportee))
(Can (Power energySource transportVehicle))
(Can (Survive transportee action))
(Interval (action Transporting) departureTime arrivalTime)

Most of the action of Irrigating was moved to more general frames, notably Transporting.

Figure 6.

Smelting
elementOf: IndustrialProcesses
inputs: MetalOre Coke
locus: BlastFurnaces
applicationOfProcess: ReductionOfMetalOxide

This activity of generalization and abstraction not only reduces the redundancy of the knowledge base, but it also drives the conceptualization—the defining and naming—of new, general concepts such as RelativePropertySlots and SolidFuelSubstances. It is at this *intermediate* level of generality that we expect convergence and power, as CYC develops, more than down at the level of particular specific facts such as those about Smelting. Intermediate level concepts exist when, and only when, we have some useful specific information about them. Useless generalizations like "GrayFuelSubstancesInAustria" will thus never be created.

As another illustration of this process, consider the frame describing irrigation (Figure 6). This began with a large script in its action slot, describing how irrigation was done. We then moved more and more information to increasingly general concepts: WaterConducting, FluidConducting, Conducting, Transporting, and so on.

Below is a snapshot of the Grebes frame while it was being entered. Note that its toes are much less than a meter long (not too surprising!); their absolute length is about average for aquatic birds, but since their bodies are so small their toes are disproportionately large for their bodies (as compared with other aquatic birds toe/body ratios).

Grebes

bioTaxonomicLevel : SuperGenuses
courtshipRituals : (# - - (Including (PairDancing with locus = (Surface BodiesOfWater)))
relComplexity High)
distributedOver : Earth
genuses : (# - - - numberOfEntries (# - - (Range 3 5)
justification (EncyclopædiaBritannica)))
habitat : QuietBodiesOfWater
physicallyResembles : Loons Ducks
relFunctionDescription : Flying (skillLevel Low) Walking (skillLevel Low)
relProportionPartsDescription : Wings (size Small) Tail (size VSmall) Toes (size Large)
relativeMagnitude : Toes (size Medium) Legs(rearPlacement Large)
Body (height Small shape Stubby) Bill (shape Pointed)
actualValue : Tail (size Small) Body (height Small)
Toes (size Small shape Lobed webbingBetweenToes absent
webbingAroundEachToe (OneOf (# - - absent justification EncyclopædiaBritannica)
(# - - present justification ColumbiaEncyclopedia)))
rolesOfParts : (Tail NoFunction)
superGenusOf : Podicipedidae
supersets : AquaticBirds Podicipedidae
myElementOf : Units
myCreator : Mitchell
myCreationdate : " 7-Aug-85 14:17:01"
myLastEditDate : " 7-Aug-85 16:55:41"
myLastEditor : Lenat

The # notation is akin to LOOPS' (Bobrow and Stefik, 1983) active value notation. For example, in the *courtshipRituals* slot of Grebes, there is a property/value pair that says that those rituals—regardless of what they actually are—are highly complex. Other common uses for property/value pairs on slots' values are to indicate the justification for that value, who believes it, at what time it was true, etc. They also hold meta-level performance and other record-keeping information which is useful later to guide the processing of resource-limited Get requests (and, eventually, as data to induct upon).

For our project to succeed, the need to augment our representation language must taper off in time. That is why we're representing one example of each kind of article first, rather than focusing on a particular area in depth and then only later discovering holes in our representation language.

Indeed, almost all the Irrigating information was moved all the way up to Transporting; a little was left behind along the way (*e.g.*, the frame WaterConducting retains the knowledge that the transportee is water). The Irrigating frame now only needs to remark on the restricted set of transport vehicles (pipes and ditches) and the restricted set of energy sources used (pumps gravity capillary action). Once this general knowledge of transportation processes was codified, adding the frames for the concepts of the rain cycle (*i.e.*, EarthWaterEvaporating, RainCloudsForming&Moving, and Raining) went quickly because each of them is an instance of Transporting, having its own particular values for destination and departure points, energy source, and a few other slots.

Step 3: Extract and Encode the Implied (Common Sense) Knowledge.

At this stage, after representing the factual information from the Coke article, we inspected each sentence, extracting from it the common sense knowledge of the world that its writer assumed its reader would know. Examining the Coke article, we noticed these unstated facts:

- Each kind of fuel is a mass noun; *i.e.*, there isn't just one piece of coke in the world; if you split

one in half, each half is a full-fledged piece of coke.
- Hardness and porosity are negatively correlated; both are uncorrelated with color; all three are form (appearance) properties of physical objects.
- Co-occurrences of negatively correlated properties are worth noting; they help define a concept (or at least help distinguish it from others).
- Fuels have widely varying color, solidity, and other physical attributes.
- Real world objects have both *form* (physical attributes) and *function,* and both of these kinds of properties are assumed to be relatively constant over time, place, and from one individual (piece of coke) to another.
- "Fuels" is a class of substances grouped together because of similarities in function, not form. They store energy from one source so it can be released at a different time, place, and/or rate that enables some subsequent process to take place. Typically, the fuel is consumed at that time.
- Most processes, including heating, generally start at a certain time and stop at a later time. The duration is rarely random: it's either fixed or lasts until some condition is met (typically, this condition is: the goal of the process is either achieved or irrevocably failed) or noticed by a monitoring process.
- After a process runs, residues may be left—usually deposited where the original inputs to the process were located. While a process is running, dynamic outputs are generated, usually at that same locus.
- Industrial processes are run either for their dynamic effect, or because their outputs (including residues) are more valuable than their inputs.
- A substance is valuable if it's used in a process that is highly valuable (*e.g.*, producing an even more valuable end product).
- Often, substances are needed in purer states than those in which they naturally occur (or initially get produced).

Besides this, the article presumed that the reader was already acquainted with (or would now look up an article on) various items: carbon, coal, hardness, porosity, grayness, residues, heating, absence, air, extracting, metals, ores, and blast furnaces.

As we began to collect common sense knowledge, two things surprised us.

First, common sense knowledge often involves an understanding of economics and, more generally, human motivation and values. For instance, the article doesn't say why we'd bother to produce coke out of coal, and metal out of ore and coke. It assumes the reader understands that such processes must be economically viable, hence the end products must be much more valuable than the inputs. Whenever such processes are or were used, there was probably no known alternative method that produces those same end products more cost effectively (where, perhaps, other "costs" than ergs are involved: threat to life and limb, pollution, need to centralize rather than distribute the loci, and uncertainty of success despite a high expected return on investment).

The second surprising thing is that most of the unstated knowledge is declarative—not procedural—information about other concepts. In other words, the common sense you need to understand $X is often just the explicitly stated form and function knowledge about concepts mentioned in the various values of slots of $X (notably in $X's basicKindOf and other types of generalization slots). For instance, much *common sense* knowledge about Coke turned out to be *factual* knowledge about Fuels; much of the the *common sense* knowledge required to understand Fuels turned out to be the *factual* knowledge about even more general concepts: physical objects, energy, and economics.

Step 4: Extract and Encode the Intersentential Knowledge.

Another surprise occurred when we entered Step 4 of our loop, looking for the unstated knowledge that let the reader cross the gap from one sentence to the next. In most short articles, there was almost no such knowledge required! The articles were simply a stream of facts, loosely clumped into sentences. For instance, in the Coke article, the three sentences could come in any order, with no degradation in the article's content or comprehensibility. Only in the long articles (several paragraphs in length) and in historic accounts of a person or region did we find such "missing link" common sense knowledge. In the latter case, typically what's going on "between" sentences is that various actors in the scenario are reacting to what just happened (matching their perceptions to their own goals and then deciding what to do next). For instance: "Atomic bombs were dropped on Hiroshima and Nagasaki. Japan surrendered."

We have found that compilations of goals into hierarchies of scripts similar to MOPS (Kolodner, 1980) handles both the shallow, fast inferences, and—when needed—the reasoning from deeper and deeper knowledge. Coke does not provide a good example of this; instead, consider these sentences (brought to our attention by John McCarthy): "Napoleon died on St. Helena. Wellington was saddened." The intersentential knowledge includes: Wellington outlived Napoleon. Wellington heard of Napoleon's death. These two facts follow from more general knowledge:
- People react to an event iff they're conscious of it and have an interest in some features of it.
- To be conscious of an event, a person must be alive and witness or hear of the event.

- People can't be in two places at once.
- People live for a single continuous interval of time.

Since the writer of the two sentences didn't explicitly state that Wellington heard of Napoleon's death, he must have considered it to be obvious that such would have been the case. What is the knowledge used to make it unsurprising to a human reader? Below are some of the necessary facts:

- The "doings" of famous people are widely publicized.
- "Doings" include births, deaths, weddings, divorces, victories, defeats,...
- Such doings are even more likely to be known by other important contemporaries (in intercommunicating cultures).
- Such doings are even more likely to be known by "interested parties," those who have some close relationship to the famous person.
- "Close relationships" of this kind include family members, allies (coworkers, team members), enemies (rivals, victims), and so on.
- Wellington and Napoleon were not only important contemporaries, but also bitter enemies.
- European countries have had regular communications passing among them for centuries.

A slightly deeper analysis rests on the fact that there was an initial *witnessing* event (in which one or more people saw Napoleon die or saw direct evidence of it, such as his corpse); following this, there was a chain of *communicating* events, wherein each person told the next one; finally, that chain stretched to Wellington. It is likely that one or more links in the chain were not strictly verbal, but involved the written reporting of the event in newspapers or letters and the subsequent reading of that story by the next person in the chain. This explains the *possibility,* but not the *likelihood* of Wellington hearing the news; If it had been Napoleon's barber who died on St. Helena that day, we wouldn't expect Wellington to have heard of that event. In other words, why is it true that "the doings of famous people are widely publicized, especially to other important, involved contemporaries?"

The witnessing event needs no further justification—perception of other people is more or less "turned on" for human beings all the time. But why would the chain of communicating events take place? Why is it likely that that spreading activation network touched Wellington? To answer that, we had to dig deeper into human motivation.

Clearly the people in the chain were satisfying some goals or subgoals of their own by passing the news along. Mostly, this appears to be curiosity (on the part of the listeners) and self-aggrandizement (on the part of the speakers). Wellington's close relationship with Napoleon would have made him even more eager to hear of the news, and Wellington's high position in England promised yet higher rewards for whoever should bring it to him, so someone may even have gone out of their way to have been the first to tell Wellington.

We went one step deeper, analyzing why people are curious, why they try to make themselves seem important, etc. This entailed organizing (and entering into our system) a top-level tree of human goals. Incomplete and naive as it is, it has proven useful for explaining several intersentential gaps so far. The top (general) level of this tree ended up to be species preservation and self preservation and emotional well-being. Below that are self improvement, self aggrandizement, group (esp. family) preservation and improvement, sating desires, and avoiding unhappiness ("dreads"). Beneath that (as subgoals) come curiosity, ecstasy (including physical pleasure, and vengeance), love (including friendship, family ties, and romantic love), and planning (to decrease the future costs of satisfying goals). Mental dreads include unavenged wrongs, fear itself (!), mental inadequacy (inability to meet a goal), and increasing dependency (such as drug addiction). Physical dreads include death (and—less extreme—wasting of time), pain, and physical inadequacy. Pain included the absence of a needed resource, hence included thirst, suffocation, starvation, and addiction withdrawal. There were some independent partitionings of motivation-space as well, such as conscious vs. instinctual, physical vs. mental, short vs. long term, individual versus communal, and so on. These became the names of slots that each kind of motivation frame possesses.

The point here is that several levels of knowledge are present in and can be used by CYC. Most of the time, it's not necessary to delve below the superficial knowledge we use every day ("famous people know the doings of other famous people"). Sometimes, when the superficial knowledge fails to explain the current situation, CYC must draw upon deeper knowledge, ultimately reaching down to physics and human motivation. See Bledsoe (1985). Whenever one level of depth of knowledge fails, the program can—albeitly more slowly—reason at the next deeper level of knowledge. This is very much in the spirit of the whole CYC effort.

Feasibility and Milestones of the CYC Project

The previous section illustrated qualitatively, by example, how it is that our effort will be conducted. This section provides a set of milestones and asymptotic goals for the project and, in the process, partially quantifies the effort involved.

1. Asymptotic Goal: A documented, debugged representation language, adequate for representing the spectrum of material contained in—and assumed by—the encyclopedia. The way to measure performance for this will be to monitor how often the language needs to be augmented as new articles are encoded. In other words, how often is

the representation language the bottleneck in adding new knowledge into the system? If we fail to asymptotically approach this sort of closure, that would signal the need for a fundamental redesign of the language, perhaps abandoning the goal of enforced simplicity and naturalness in representation in favor of a simpler, fixed language within which we'd put up with somewhat awkward statements of some kinds of knowledge.

2. Milestone: An interface suitable for browsing and editing CYC's knowledge base. The measure of performance is to be able to locate and enter knowledge faster using this interface than with current text-based screen editors, such as the Grapher and Inspector packages of Interlisp-D (Interlisp Reference Manual, 1983). Our initial design involves viewing the KB as a semantic net, employing three dimensional graphics to display it, and developing navigational aids of various sorts. The point of (1) and (2) is that the bottleneck to building CYC should not be orienting oneself in knowledge space nor using this editor to find and change the knowledge base. We want (so to speak) the bottleneck to be in deciding what the knowledge is that ought to be entered, not in the entry process itself. That's why the language and the interface to it are being stressed early on; they are "power tools" for knowledge entry. If we fail, the existing graphics metaphors may have to be expanded, extended, or replaced; additional types of displays and interfaces may be experimented with; the text-based editors may be improved enough to eliminate the need for the graphics interface; automated personae may be simulated to act as agents and guides.

3. Asymptotic Goal: Have the CYC system find and use analogies: (a) analogy useful in entering new knowledge, (b) analogy interesting in its own right to human beings; (c) analogy capable of coping with some situation that would "break" an expert system. The measure of progress toward this goal is how often and how usefully our knowledge enterers exploit analogies while they add new knowledge; to hook performance programs up to our system and track how often analogy (rather than just general knowledge) gets them out of binds; and so on. If we fail to have CYC exhibit any of these three uses of analogy, that would force us to rethink our ideas of how analogizing gets done. We would look more carefully at real cases of problem solving and knowledge acquisition by analogy and try to reformulate algorithms for those processes. There are several alternative diagnoses we may consider: Analogy may not be as useful as we currently believe (this would actually reduce the need to have a broad base of specific knowledge); analogy *is* important and ubiquitous, but requires better interfacing tools to exploit; or more detailed cross-indexing of knowledge (more kinds of slots for our frames) must be added.

4. Milestone: One important milestone is to have represented some pieces of each *type* of knowledge we intend to represent in the coming decade. This excludes perceptual knowledge (*e.g.,* visual scenes), but includes almost all encyclopedic knowledge and the common sense underlying it. The measure of performance is that one example of each type of article is represented within CYC. As discussed earlier, there are about 400 distinct kinds of articles. We expect our system to contain about 10,000 frames at that time, about half of which will be very general (common sense) concepts. Unlike the other milestones, *there is no good alternative for failure here.* We could restrict the domain of knowledge over which CYC will have some grasp (*e.g.,* only knowledge about static, concrete, real-world objects), but it is hard to envision any such restriction that would not defeat the whole point of—and expected sources of power of—the CYC program. If this milestone is not within reach after a few years of work, it will be time to rethink the entire project.

5. Asymptotic Goal: Represent most of the common sense knowledge underlying the encyclopedia. This will of course not end until the entire encyclopedia is finished (milestone 7), and even then each piece of common sense knowledge might be further broken down (rationalized) in terms of others, and so on. Each deeper level explains the outlying cases, the exceptions not handled by the previous, more shallow layer. Hence, the codification of common sense knowledge should be asymptotic: the first few thousand facts being the most important and useful ones, and so on. The measure of performance is the number of such concepts represented, and the rate at which new ones are needed. The average article we add now (Fall 1985) requires dozens of new common sense frames to be added; this rate should drop asymptotically over time; our current guess is that the rate will stabilize at about 0.1 new entry per article, after the first few thousand articles are entered. If we fail to converge, working by ourselves, on this entry process, that is an interesting finding about common sense. It would mean that the knowledge enterers would have to have and use tools for adding common sense knowledge themselves, as they went along adding factual knowledge from articles. If the rate of spawning new common sense concepts does not come down, not only would those tools be needed, but careful monitoring of the semantics of the common sense knowledge base would then be in order. It might turn out that new ways of organizing it (*e.g.,* hooking it to a set of simulations of the world) would be needed to keep it from continuing to explode in size.

6. Milestone: Utilize a large cadre of lightly trained knowledge enterers, working together on a common, consistent version of the system. This follows a decision point to hire these workers, reached after milestones 1-4 have been met. The measure of performance for this task (cost-effective

utilization of the knowledge enterers) is the rate at which they add new concepts and the degree to which they maintain a consistent semantics for that knowledge. Our first "knowledge entry personnel" will help us test out the various tools we'll be building to speed up their task. We also must discover what kinds of knowledge enterers we need. If we fail to fully utilize this group of people, we will vary our techniques for selecting them, training them, keeping them oriented, managing them, having them work in teams, etc. Continued failure at this could be fatal to the project, but two possible alternatives are: (a) devise an interface that lets us (a small group of AI researchers) enter knowledge ten or twenty times faster than we can at present; (b) mount a coordinated effort, distributing the task of building the KB to other AI research groups.

7. Asymptotic goal: Enter "the world's most general knowledge," down to ever more detailed levels. A preliminary milestone would be to finish encoding a one-volume desk encyclopedia. Other sources of knowledge (such as childrens' stories) will be examined, and some samples will be entered. *Systematic* entry of that level of knowledge might be called for eventually, if the project is a success. The measure of performance here is probably as simple as the amount already encoded, plus a check on the rate at which each knowledge enterer is adding knowledge. There are approximately 30,000 articles in a typical one-volume desk encyclopedia, and most of them are one paragraph long. For comparison, the *Encyclopædia Brittanica* has nine times as many articles, and their mean length is about five times as long. A conservative estimate for the data enterers' rate is one paragraph per day; this would make their total effort about 150 man-years. The initial 1 percent of the encyclopedia articles and the initial 50 percent of the common sense knowledge—the parts that we are doing ourselves—will be coded at a much slower rate than the final parts, not only because the system will eventually help the knowledge enterers, but also because as we progress we are constantly exercising our representation language and must occasionally add new capabilities to it. If we fail to encode the entire encyclopedia, but come close, that will be fine; if we are not going to even get close by the end of our time horizon (roughly one decade), that will call for serious replanning. Perhaps we will simply need more knowledge enterers, if milestone 6 has been reached successfully. Or, if milestone 8 is within sight, perhaps accelerating work on tying our system to a natural language understanding program will be useful, so that it can "read" some of the remaining articles on its own. If milestone 3 is a success, accelerating the reliance upon analogy may rapidly get us at least a "first pass" over the breadth of the encyclopedia.

8. Asymptotic Goal: As this project proceeds, test out the system by hooking it to various particular AI programs: expert systems, natural language understanders, and problem solvers. The measure of performance for this would be the gain in performance in the hooked-up programs and, to a lesser extent, CYC itself. If we fail to boost their performance, we may have to wait until the CYC program approaches its final size, *i.e.,* we may not have passed the "knee of the curve" yet. Another possibility is that these performance programs would have to be written *within* CYC, rather than merely being loosely hooked up to it.

To close on a more positive note, we consider finally the case that we succeed in reaching these milestones and approaching these asymptotic goals. At that time, many more expert systems (and other performance programs) would "hook in"; meanwhile CYC would be adapted into products: knowledge utilities akin to electric utilities, advanced entertainment and art tools, autonomous design-discovery engines, and the next generation of ICAI systems.

References

Amarel, S (1983) Panel discussion on machine learning, transcribed in (Michalski *et al*, (Eds.),) Proc. 2nd International Machine Learning Workshop, (publication forthcoming)

Bledsoe, W. (1985) *I had a dream* AAAI Presidential Address, IJCAI 9, (forthcoming, *AI Magazine* 7 (1)).

Bobrow, D , & Winograd, T (1977) An overview of KRL, a knowledge representation language *Cognitive Science* 1 (1).

Bobrow, D., & Stefik, M. (1983) *The Loops Manual.* Palo Alto: Xerox Corp.

Brachman, R (1985) I lied about the trees. *AI Magazine* 6 (3): 80-93.

Broad, W. J. (1985) Thomas Edison papers give insights into creative genius *Austin American Statesman* 14 March 1985, p. C8.

The Concise Columbia Encyclopedia (1983) New York: Columbia University Press.

Carbonell, J., & Minton, S. (1983) Metaphor and common-sense reasoning. Tech. Rep CMU-CS-83-110, Computer Science Department, Carnegie-Mellon University.

Encyclopædia Britannica (1984). Chicago: Encyclopædia Britannica

Green, C , Waldinger, R , Barstow, D , Elschlager, R., Lenat, D., McCune, B., Shaw, D., & Steinberg, L (1974) Progress report on program understanding systems Memo AIM-240, AI Laboratory, Stanford University.

Greiner, R., & Lenat, D. (1980) *RLL: A representation language language.* AAAI-1, 165-9

Greiner, R. (1985) Learning by understanding analogies. PhD thesis Computer Science Department, Stanford University.

Hadamard, J (1945) *The Psychology of Invention in the Mathematical Field* New York: Dover

Interlisp Reference Manual (1983) Palo Alto: Xerox Corp

Kolodner, J (1980) *Organizing memory and keeping it organized.* AAAI 1, 331-3.

Lakoff, G., & Johnson M. (1980) *Metaphors We Live By.* Chicago: U. Chicago Press.

Lenat, D , Borning, A , McDonald, D., Taylor, C., & Weyer, S. (1983) *Knoesphere: Building expert systems with encyclopedic knowledge.* IJCAI 8: 167-169.

Lenat, D , & Brown, J S (1984) Why AM and Eurisko appear to work. *Artificial Intelligence Journal* 23: 269-294.

Lenat, D. (1984) Software for Intelligent Systems. *Scientific American* 251: 204-213.

McCarthy, John (1983) Some expert systems need common sense, *Annals of the New York Academy of Sciences* 426, 129-37

Minsky, Marvin (1984) *Afterword to: Vinge, V, True Names.* New York: Bluejay Books, 130-53.

Mitchell, T (ed) (1985) Proceedings of the 3rd International Machine Learning Workshop Rutgers University, (in press).

The New American Desk Encyclopedia (1984). New York: Signet

Petrie, C. (1985) Using explicit contradictions to provide explanations in a TMS MCC AI Tech. Rep. 0100-05, Microelectronics and Computer Technology Corporation, Austin, Texas.

Poincaré, H. (1929) *The Foundations of Science, Science and Hypothesis, The Value of Science, Science and Methods.* New York: Science Press

Schank, R., & Abelson R. (1977) *Scripts, Plans, Goals, and Understanding* Hillsdale, N.J : Lawrence Erlbaum Associates.

Skemp, R. R (1971) *The Psychology of Learning Mathematics.* Middlesex: Penguin Books.

Ross, K. (1981) Parsing English Phrase Structure. Ph D dissertation, U of Massachusetts.

Winston, P., Binford, T , Katz, B., & Lowry M. (1983) *Learning physical descriptions from functional definitions, examples, and precedents* AAAI-83: 433-439.

Call for Workshop Program Sponsored by The American Association for Artificial Intelligence

The AAAI has supported small workshops for the last several years. This support includes publicity, printing, office help, and subsidies for other expenses. $5,000.00 is a typical subsidy, but up to $10,000.00 may be considered. Any topic in AI science or technology is appropriate. Anyone may volunteer to organize a workshop on any topic. The organizer(s) should determine the topic, the date, the site, and the procedure for selecting papers and attendees. S/he should also decide whether preprints should be distributed.

Proposals for scientific workshops should be made to:

Professor John McCarthy
Computer Science Department
Stanford, CA 94305
(415) 497-4430,
jmc@su-ai.arpa

For workshops on applied topics, applications should be made to:

Dr. Peter Hart
Syntelligence
P.O. Box J620
Sunnyvale, CA 94088
(408) 745-6666
hart@sri-ai

AAAI proposes that program committees give special consideration to papers that have been presented at workshops in choosing invited speakers for national conferences.

REQUEST FOR PROPOSALS

Future AAAI Conference Sites

The AAAI's Conference Committee (Jay M. Tenenbaum, Chair; Ronald Brachman, and Michael Genesereth) requests proposals from the membership for conference sites for 1988, 1990, and 1991.

The proposal should be structured around the new five day format described elsewhere in this issue of the AI Magazine. Based on a predictive attendance of 6,500, the proposals should include the following information:

1. Description of the local AI community and its willingness to support the conference.
2. Description of the variety of available housing ranging from first class hotel rooms to dormitories.
3. Description of the University and/or Convention Center's large meeting rooms (ranging from 300 to 3,500 theater seating) for a minimum of three parallel sessions. Description of another set of three, parallel meeting rooms (used for tutorials) that can accomodate from 200 to 500 schoolroom seating each.
4. Description of available exhibit space (minimum requirement of 80,000 net square feet) and service contractors.
5. Description of local regulations (*e.g.*, labor union laws, liquor licenses, and local tax structure).
6. Description of local housing and convention support services from the city's Convention and Visitors Bureau. Description of procedures for processing university housing reservations.
7. Description of site's accessibility by air and ground transportation and local ground support transportation.

Ideally, the Conference Committee would prefer to hold the science sessions on a university campus and the engineering sessions at the larger convention facility.

For further details about this RFP, please contact:

Ms. Lorraine Cooper
AAAI
445 Burgess Drive
Menlo Park, CA 94025-3496

Submit all proposals to:

Jay M. Tenenbaum, Chair
AAAI Conference Committee
445 Burgess Drive
Menlo Park, CA 94025-3496