Enabling Agents to **Work Together**

There are two quite different "software agents" paradigms today:

- Paradigm 1: Competence emerges from a large number of relatively simple agents integrated by some cleverly engineered architecture. The choice of architecture is the make-or-break theoretical part of this; the detailed characteristics of the implementation of the architecture (and the algorithms that crawl around it) are the make-or-break pragmatic parts. The archetype of this paradigm is SOAR [6]; its forerunners were the early "pure production systems."
- Paradigm 2: Competence emerges from the aggregate system possessing a large amount of useful knowledge; for most real-world tasks, this includes a dauntingly large fraction of what might be termed "general common sense." In this paradigm, the architecture is relatively unimportant, and building the system as a set of "agents" is little more than a form of scaffolding, reducing the cognitive load on the human builders. The archetype of this paradigm is Cyc; its forerunners were the early expert systems.

For 20 years, we have witnessed tens of thousands of successes in which knowledge-based systems and other useful intelligent software agents have been constructed and deployed. But amidst all this success, there is constant failure as well: these systems cannot share their knowledge, hence cannot pool their expertise and cannot work together synergistically.

What is required for agents to do this? They need to share "enough" of the foundational knowledge in terms of which their specialized knowledge—or even more commonly the results of applying that specialized knowledge—can be communicated. This is, for example, what happens when you have to see a doctor or a lawyer, when you deal with a taxi driver or a salesperson or a reservations clerk. It happens constantly in every classroom with a competent teacher.

This notion, sharing "enough" of the meaning, in turn leads to three new questions: What is this foundational knowledge? How exactly do intelligent agents share it? How much is "enough"?

We began seriously examining these questions—and seriously testing the "second paradigm of software agents"—a decade ago, when (in September 1984) we began the Cyc project. Namely, we were testing the hypothesis that the primary impediment to achieving interesting agent behavior was lack of knowledge. To wit, that we wouldn't have to work nearly so hard to come up with clever algorithms and data structures and architectures, if we had a large corpus of knowledge to fall back on. Halfway through the project, we wrote a *Communications* article [5] about our progress. Today, more than four years later, we report how things have been progressing since then.

First, let's see how we would currently answer the three questions mentioned previously:

1. **What is this foundational knowledge?** It is the knowledge that, for example, a high school teacher assumes students already have, before they walk into class for the first time. This includes such things as common sense notions of time, space, causality, and events; human capabilities, limitations, goals, decision-making strategies, and emotions; enough familiarity with art, literature, history, and current affairs that the teacher can freely employ common metaphors and allusions; and so forth. Notice that the instructor assumes that two things are shared between student and teacher: (1) most of the vocabulary, and (2) most of the knowledge involving those terms. Our project—Cyc—has been codifying just such knowledge for almost a decade now.

2. **How exactly do intelligent agents share it?** Essentially, this is the question of how multiple agents can cooperate with one another. It is the backbone question of intelligent agents (IAs), the skeletal framework on which the entire enterprise depends, and, as with animal backbones, the answer for IAs comes in two forms, exo- and endo-:

- *exo-:* Impose rigid standards "from the outside." This in effect means that the collection of agents is really one large engineered (albeit modular) program. Unfortunately, dictating standards rarely works in the real world even across divisions of one large company, let alone across companies, across industries, across countries, and across decades of real time. As with animals, exoskeletal "standards" are a better idea at the insect level of size and complexity than at the level of humans (or for that matter organizations) with backbones. If carried to extremes, this method of "knowledge sharing" merges with the first IA paradigm, the one that focuses on having a clever system architecture.
- *endo-:* Inject at least partial understanding of the meaning of the information which each agent is working on, and emitting. This is what people do, after all. It is what allows us to accept one another's communications despite their being terse, ambiguous, and idiosyncratic; it bridges cultural differences and skill differences, for example.

The focus of this article will be on the second sort of cooperation. We will argue that a good maxim for an intelligent agent population is: "share most of the meaning of most of the terms, most of the time." This brings up the third question:

3. **How much is "enough"?** Almost certainly, this question will be answered empirically, and the answer will vary from task to task. Our research so far suggests that the answer is:

> *a surprisingly large fraction of "common sense" knowledge needs to be shared, even for relatively narrow tasks.*

On the other hand, this same large body of knowledge is almost all that is needed for a very wide range of tasks. So we argue that, for IAs to cooperate by sharing partial understanding of the information they are processing, they require most of the knowledge we're accreting in Cyc's knowledge base.

To address these questions, we must therefore at least briefly discuss Cyc's current state, including such issues as: how the knowledge base is constructed, what sort of knowledge it does and does not contain, how we maintain efficient **inference** in a system containing millions of axioms, and how it copes with the inevitable inconsistencies it contains (inevitable because of its sheer size, because of changes in the world, and because of cultural differences.) We will also briefly mention some of the recent Cyc applications work.

Anatomically, Cyc has many parts: the Cyc knowledge base (KB), the CycL representation language and inference engine, the knowledge server utility that allows multiple people to work together simultaneously building up the KB, the knowledge-entering (KE) methodology and the various user interfaces (UIs) employed during that process.

The knowledge server, KE methodology, and UIs are merely tools helping us realize the KB, which is ultimately the valuable product of this enterprise. For that reason, we focus here on the KB, and to a lesser extent on the CycL representation language, and not at all on Cyc's other anatomical parts.

The State of Cyc circa 1990

CycL circa 1990

CycL is the language in which the Cyc KB is encoded. It includes both the specification of the syntax and vocabulary of the statements that constitute the KB, and the programs for performing **inference.**

Until 1987, CycL was essentially a set of procedures for performing inheritance, automatic classification, etc., that operated on a set of **frame**-like data structures. Uncertainty was expressed by associating certainty factors (numbers between 0.000 and 1.000) with every statement in the KB.

The frame-based representation was awkward for expressing various assertions we wanted to make: disjunctions ("apartment residents may keep a cat or a small dog"), inequalities ("retirees have more grandchildren than children"), negations ("homeless people don't watch many movies"), existentially quantified statements ("one member of the Cabinet is likely to be a minority woman"), meta-level statements ("the previous axiom was entered in 1988 by a cynic") etc. We also came to realize that having an implementation-independent semantics for the KB (which frame-based systems usually lack) was vital so that we wouldn't have to redo the KB every time our implementation of one of Cyc's **inference mechanisms** changed. These factors slowly moved CycL away from frames, and toward a predicate logic orientation. This gave it more expressiveness and also provided an implementation independent semantics for the KB.

We also did away with numeric certainty factors for each assertion. They are downright dangerous in a huge KB, as any two numbers can be compared with each other, and the three-digit numbers were often little more than fabricated ways of expressing the fact that one assertion was a bit "more likely than" another assertion. In their place, we allowed explicit statements that "P is more likely than Q," and we developed a more principled way of handling defaults: Cyc constructs and compares *arguments* for and against each proposition, using explicit rules to decide when an argument is *invalid* or when one argument is to be *preferred* to another. We will discuss this procedure—which we called *argumentation*—in a later section of this article.

"Users" of Cyc (both humans and application programs) interact with the system via an *epistemological-level* (EL) language similar to first-order predicate calculus (FOPC) with augmentations for **reification, defaults, modals, reflection,** etc. The entire

Glossary

API: Application Program Interface. An agreed-upon input/output format for a particular APP (application program). Humans, or application programs, may rely on that format and in particular use it to "call" the APP and to interpret the output returned by the APP. For example, SQL.

completeness and correctness: If a statement does indeed follow from the KB—that is, it should theoretically be inferred—and you ask the system whether that statement is true or false, are you sure the system will say yes, it is true? If so, the system is said to be complete. If it will at least not say no, it's false, then the system is said to be correct. We can and do all live with extreme and ubiquitous incompleteness in our lives, but incorrectness should be relatively rare.

defaults: statements which may not be absolutely true. Often they are "usually true," and exceptions might be known or expected. Sometimes a default statement is just an assumption, and later statements might entirely refute it.

frame: a set of attribute/value pairs, about a single concept. For example, the USA frame might have pairs such as leaderType/President, yearOfFormation/1776, etc. One can think of an s/v pair on the x frame as equivalent to stating s(x,v).

indexical: for purposes of this article, assume it to mean any pronoun or other word that requires context to disambiguate. For example, "now," "yesterday," "here," "we," and so on.

inference: deriving conclusions that follow from the Knowledge Base.

inference method/procedure/scheme: a procedure used for doing inference. Examples include logic, analogy, argumentation, and statistical pattern recognition.

ist: a very special modal predicate, which stands for "is true in." (ist C P) means that proposition P is asserted as being true in context (microtheory) C. Notice that from (ist C P) and (implies P Q), we *cannot* in general infer (ist C Q). If, however, we know (ist C (implies P Q)), then indeed we can infer (ist C Q). (More precisely, we must also know that Modus Ponens "is true in" context C, but this is assumed by default to be the case.)

modals: statements involving special **predicates** such as *beliefs, desires, knows, expects.* One cannot substitute equals for equals inside a modal predicate. For example, Fred's age may be 30, and Joe might believe that Fred's age is 29, but that certainly doesn't imply that Joe believes 30 is 29. This is also called "referential opacity."

predicate: a true/false-valued function. For example, one can think of "age" as a unary function, as in age(Fred) = 30, or as a binary predicate, as in age(Fred,30) = True. In prefix notation, we would write this as (age Fred 30).

prefix notation: writing (F x y) instead of, for example, xFy, to denote function/predicate F applied to arguments x and y.

reflection: the ability to control inference using statements in the KB. For example, one reflective statement might say: "It's better to try 'analogy' only after all the other inference methods have failed."

reification: the ability to explicitly refer to a statement, or even a bundle of statements (for example, by name, by pointing at them) so that one can then make metalevel statements about those other statements.

KB is represented in this EL language. It has a simple semantics and is therefore easily used to communicate. Internally, at an invisible (to the users) *heuristic level*, these EL assertions are converted and handled by special-purpose data structures and algorithms for speedy inference.[1]

CycL's default reasoning relies on the concept of arguments. An "argument" is very much like a proof, but it may contain sentences that have been labeled as assumptions. Arguments themselves may be labeled as "invalid" if, for example, they rely on assumptions that are (due to other arguments) believed to be false. Often Cyc will come up with numerous distinct arguments for both P and its negation, $\neg P$. In such situations, CycL relies on a technique called *argumentation* [3] to determine whether to believe P, $\neg P$, or (for the time being) neither.

The Cyc user (human or application program) communicates with Cyc via an Application Program Interface (API) containing the sort of commands one would expect: Assert, Deny, Ask, Justify, etc. In particular, "Ask" is used to test the truth value of a statement, and—if Ask is given an expression containing some free variables—to find sets of variable bindings which make that expression true. An optional argument (UNWANTED-BINDINGS) turns Ask into a *generator;* that is, each repeated call yields one or more new sets of bindings. Other optional arguments to Ask can provide it with resource bounds (time/space cutoffs), heuristics for ending or even guiding its search, etc. Most of Cyc's fast inference comes about because of the way we implement Ask.

Most representation languages make some trade-off between expressiveness (how easy it is to say many different complicated things) and efficiency of inference. Higher-order logics, and natural languages such as English, are at one end of this tradeoff. Subsets of propositional logic, and low-level programming languages such as C, are near the other end. Most "expert system shells" such as ART and KEE are somewhere in the middle. CycL clearly needs to be both expressive and efficient. In order to get both, we made the difficult decision to sacrifice **completeness.** As long as Cyc draws more or less the same inferences that people

[1] Collectively, the latter are sometimes referred to as the *heuristic level* (HL) language, but this may be somewhat misleading as no user—human or application program—ever writes or reads HL expressions.

do, in various situations, we are satisfied with the degree of completeness of its logic. Furthermore, the user can determine the "level of completeness" by specifying resource bounds (e.g., in terms of time spent, depth of search) on their Ask's.

CycL incorporates a host of heuristics for improving efficiency of inference. Some of these efficiency-boosting heuristics are in effect specialized inference mechanisms, based purely on the syntactic structure of the axioms. Other heuristics exploit some special properties of a set of domain-specific axioms, and/or domain-specific use of a set of general axioms, for example, temporal reasoning. We have found it is possible to obtain significant improvements in problem-solving efficiency by using an analysis of the structure of the axioms in the KB. Here is a simple example: most of the axioms that mention emotions refer to a specific emotion by name, rather than just saying "an emotion x." CycL uses this simple statistical regularity to speed up inference of new axioms pertaining to the hedonic states of actors involved in events described to (or inferred to occur by) Cyc. At the heart of all these mechanisms is a relatively general inference mechanism that performs resource-bounded iterative deepening search.

Cyc's KB circa 1990

The kinds of terms used in Cyc assertions include, primarily, categories (collections) and individual things in the world. To a lesser extent, there are terms that refer to reified assertions, and to internal machine objects such as strings, numbers, lists. Further details on the Cyc ontology (circa 1990) can be found in [8].

Cyc's categories are organized in a generalization/specialization hierarchy (a directed graph, not a tree, since each category may have several direct generalizations.) The Cyc **predicate** *allGenls* relates a category to its supersets; for example, in **prefix notation,** we might assert to Cyc the following statement: (*allGenls Person Animal*). Membership in a category is represented by the predicate *allInstanceOf*; for example, (*allInstanceOf DougLenat Person*). Though we frequently use set-theoretic notions to talk about Cyc collections, they are more akin to "Natural Kinds" [12]. Cyc collections are not merely mathematical sets, they are sets with intensional properties, sets with useful things to be said about them.

Predicates in Cyc are strongly typed. The "type" of each argument must be a single Cyc collection. These "type" constraints are stated as assertions in Cyc, which is no problem as each predicate itself is a first-class object in the language. For example, (*mother x y*) is a Cyc expression meaning that *y* is the mother of *x*; the following axiom says that *y* must be a female animal: (*argument2type mother FemaleAnimal*).

Our 1990 *Communications* article [5] focused exclusively on some of the highest-level (most general) collections—and hence distinctions—in Cyc's ontology; our 1990 book did even more of that: Thing, PartiallyTangible individuals; Substances; SomethingOccurring; and issues such as time, space, causality, and belief. This is somewhat misleading, however, in that the day-to-day entering and exercising of knowledge in the KB has, for the last five years, not focused on the organization of knowledge at those ethereal heights. The knowledge enterer has, instead, typically been working on a domain such as "Furniture" or "Eating," and they need to state their new axioms using (some new terms and) old terms like Sitting, Sleeping, Storing, Flat, Size, Cost, Vegetable, not vague general terms like PartiallyTangible or Thing.

Similarly, at the time of writing of our previous *Communications* article, most of the KB was conceptually just one enormous list of assertions that were supposed to be self-consistent (not counting exceptions to defaults.) That has changed, so that now each of the KB axioms is located in one (or occasionally more than one) specific "context" or "microtheory."

Each microtheory captures one "fairly adequate" solution to some ontological area: either general areas like representing and reasoning about time, substances, agents, and causality, or specific areas like weather, manufacturing, and dining. Some areas can have several different microtheories, representing different points of view, levels of granularity, and distinctions that are and are not made. Note that we are not claiming to have—we are not trying to have—a single complete theory of time, or space, or for that matter human physiology. What we do have, in each case, is a suite of microtheories whose union covers the most common cases.

We had predicted something like this occurring, back in 1990: "*We expect that most of the theories of topics at a finer granule than those mentioned in this section, that we will add to Cyc, will make use of this notion of microtheories and that it will figure prominently in the overall structure and contents of the KB.*"

Cyc's Evolution from 1990 to 1994

(Non-)Changes in the Overall Plan

One of the most significant items we have to report in this article is that the basic mission of the Cyc project—to create a commonsense substrate for the next generation of software—has remained largely unchanged and on track. We also continue to believe that the best way of building such a substrate is to start by hand-encoding, in some form of symbolic logic, a significant fraction of the things that most average people know. As we proceed with building this corpus, applications such as NLU (natural language understanding, based on the knowledge in Cyc) can be built to bootstrap and accelerate the KB-building process. Over the last few years, we have made significant progress both in building the substrate and in starting the boot-strapping process.

One area in which our thinking has significantly changed has to do with the first generation of Cyc applications. In 1984, we envisioned the primary 1996 use of Cyc as being a substrate on which to build the next generation of expert systems, helping to make them less brittle. We now feel that the first uses of Cyc will have to do with more mainstream computing, in the area of information management.

The explosion in computer connectivity over the last few years has led to vastly greater amounts of infor-

mation being available on-line. However, most of the task of knowing where relevant information resides, and accessing it, has been left to users. Despite a growing number of tools such as WAIS (Wide Area Information System), WWW (World-Wide Web), Gopher, this still remains a very hard problem, and one which is likely to become dramatically more serious quite soon, as the National Information Infrastructure/Highway and global information accessibility continues to mushroom. This is one area in which we believe Cyc has important and economically significant contributions to make. Later in this article, we describe prototypes of some Cyc-based information management applications that have been built by us over the last few years.

Progress on the Knowledge-Entering Front

The primary Cyc project activity over the last few years has been improving the Cyc knowledge base. The content of the KB has grown significantly. Unlike during the first half of the project, the focus over the last few years has not been on general issues such as time and substances, but on much more concrete topics such as business processes, social events, and tangible products.

As much as we would like to provide quantitative numbers indicating our progress, our experience has taught us that such numbers (e.g., the number of constants, or the number of axioms in the KB) are very misleading. This is because there are two different ongoing processes that affect these numbers: On the one hand, the KB is increasing in size because of knowledge entry. On the other hand, there is a constant effort to generalize the statements in the KB, which often results in a large number of axioms being replaced by a relatively smaller number of better axioms, without decreasing the set of conclusions that can be drawn.

So "improving the Cyc KB" includes of course increased content of the KB, but no less significantly also includes increased quality and utility of the KB (and generally *de*creased size of the KB.) While it is difficult to quantify progress on this front, (except through the performance of Cyc applications), the qualitative metrics are all positive.

The most significant qualitative criterion for determining our progress is that of "convergence." One of the primary assumptions behind Cyc is that there will be convergence as the KB building progresses. To make this more specific, here is our list of convergence-related problems that could have caused us to stumble. We were afraid that as new topics were tackled . . .

a. . . . we would be forced to redo much of our treatment of basic topics such as time, substances, and intentions. Fortunately, this has not been the case. We have found our treatments of these topics adequate and have been able to focus our efforts on the new topics themselves and have not been distracted by having to redo the basics.

b. . . . each topic would require (and only use) its own idiosyncratic vocabulary and basic axioms. If this had turned out to be the case, the KB would be nothing more than an agglomeration of many small, separable KBs; and Cyc would degenerate to nothing more than a large KBS-(knowledge-based system) building tool with a number of built-in libraries. Contrary to our fears, we have required a *decreasing* amount of new predicates, functions and important categories for each new topic we deal with. Most of the vocabulary for axiomatizing a new topic typically now comes from one or more existing related topics. This makes it relatively easy for the new knowledge that is being entered to be integrated with existing knowledge, to be used productively with it during inferencing.

c. . . . it would become increasingly difficult for humans to manually incorporate each additional piece of new knowledge into the right place in Cyc. The difficulty in doing this has remained approximately constant, due to two factors offsetting each other: On the one hand, the KB has grown more complex and larger. On the other hand, the topics that are being entered are more specific and more constrained; also, Cyc itself provides more and more help with the assimilation. Together, these factors seem to have cancelled each other and the average productivity of our knowledge enterers has remained about the same.

d. . . . we would have to keep adding new features to CycL to express pieces of knowledge in new topics. The only significant addition to CycL over the last four years has been more tightly integrating and refining the concept of "contexts." We will have more to say about this later in the article.

Two New Cyc Activities

Cyc is now at a stage where it is time to start planning for the inclusion of Cyc in real applications. As mentioned earlier, we believe the first generation of Cyc applications will have to do with information management. In anticipation of this, over the last few years we have used Cyc to power a number of small proof-of-concept prototypes, for various types of information management applications.

The other major new activity has been in the realm of Cyc-based Natural Language (CNL) processing. For a little over a year now we have been building CNL, and the initial results look quite promising.

Later sections provide more details on these two new activities.

The Cyc KB circa 1994

Figure 1 is a sort of macroview of the current (March 1994) state of the Cyc KB.

A microview of the KB is described in the following paragraphs through excerpts form one of the many theories recently entered into the KB, namely one on physiology. Included are examples of the vocabulary that was introduced, plus some of the axioms written (both in English and, in a few cases, in CycL.) The goal here is not to exhaustively list Cyc's knowledge about this topic, but to provide the flavor of the KB's contents.

Cyc's knowledge of physiology can be broadly broken down into three interrelated topics: anatomy, ailments, and medical treatments.

To tell Cyc about anatomy, we de-

```
1.  The 30k constant terms

Categories              ---40%
     Categories of categories  -    .5%
     Categories of individuals - 39.5%
          Categories of Intangible objects -- 3%
               C. of Information-Bearing Objects
               C. of Numbers, C. of Physical attributes, etc.
          Categories of Tangible objects --- 18%
               C. of Living Things
               C. of Artifacts
                    C. of Things around the house
                    C. of >Human-sized objects
     Categories of Script types       --- 15%
          Actions by 1 "person"
               Physiological Actions
               Problem Solving/Planning
               Work/Hobby/... Actions
          Actions by >1 "person"
               Rites of Passage
               Communication
          Natural Phenomena (e.g., Weather)

Predicates & Functions          --- 15%
          Unary:  see Categories and Attributes
          Binary -- 12%
          Ternary -- 2%
          Quaternary+ -- 1%

Attributes (a type of unary predicate) --- 10%
Lexical objects --- 15%
     Words -- 14+% (and growing)
     Parts of Speech, Tense, Number, Gender,... -- <1%

Proper Nouns (Specific people, places, languages, events, etc.) --- 15%

Microtheories (long-lived contexts) -- 1%

Misc. and Sundry --- 8%


2.  The distribution of formulae

Taxonomic information --- 25%
     (including type constraints on predicates, etc.)

Partonomic relations --- 35%
     what kind of parts physical/anatomical/subEvents...
          might various types of objects have? -- 15%
     what kind of actors are involved in
          various script-types? ---15%

Lexical information --- 10%
     Linguistic properties of different word senses -- 8%
     Denotations of word senses -- 2%

More complex information interrelating script
     types, people and tangible objects --- 10%

Generic topics (time, space, intentions, etc., stuff about numbers,...) --- 10%
Information about specific people, places, etc. --- 5%

Misc. and sundry formulas --- 5%
```

Figure 1. Homunculus of the Cyc knowledge base circa March 1994

scribed to it all the "well-known" organs (heart, liver, but not pituitary gland) and external body parts (leg, toe, but not philtrum). For each of these, Cyc was given axioms about its typical size range, the number possessed by each animal, how many of these parts are critical for survival, and so forth. Also included, of course, was the function of each anatomical part. However, the functional descriptions included very little mechanistic description, but rather consisted mostly of commonsense dependencies: heart beating is required for life; the stomach is required in digestion; the mouth is required for eating and breathing; the stomach connects to the intestines, and so forth.

Here is some of the vocabulary specific to Physiology, which was created during this axiomatization activity:

- BodyPartOF—(Function)—takes an Animal and a UniquePartType (which is also limited to body parts) as arguments and returns the specific part described
- BodyPartTypeOf—(Function)—takes an Animal and a BodyPartType as arguments and returns the collection of that animal's body parts of that type
- medicallyRecommended—(Predicate)—To describe the relationship between drugs and the ailments they are designed to treat, we use the predicate medicallyRecommended. That same predicate can also be used with nondrug treatments, such as exercise or sleep. In general it takes a PhysiologicalCondition PC, a SituationType ST, and an ActorSlot AS as its three arguments. Then (medically-Recommended PC ST AS) means that if an animal has the condition PC, it is recommended that they play the role AS in some instance of ST, in order to ameliorate the condition.
- medicallyProscribed—(Predicate)—analogous to medically Recommended, but this predicate describes actions that an animal with a certain condition should not do (more precisely, it describes certain roles in events that this animal should not play).
- DrugTherapy—(ScriptType)—To talk about events when someone is under the influence of medicine. The duration of a DrugTherapy event is the entire period of the drug's effects on the animal.
- DrugTherapyUseOf—(Function)—takes the type of drug as an argument and returns a ScriptType which is an instance of DrugTherapy and which involves the use of that drug. For example, an instance of (DrugTherapyUseOf Aspirin) immediately follows the event in which the patient swallowed the pill, and lasts four hours.
- ModernWesternMedicineMt—(MicroTheory)—This new context was created for assertions that are true primarily in the 1990s, in the developed Western countries. It is not intended to be accurate if applied, for example, to Ethiopia or China today or Pennsylvania in 1800. One of its main uses, ironically, has turned out to be for assertions about current medical costs.

In the area of common ailments and symptoms, the KB includes the types of anatomical parts affected, the sensations caused (e.g., pain, nausea), and the capabilities affected (generally impaired.) There is also information about duration and typical causes.

Ailments are represented as event types. Though many physiological states are mostly characterized by a sensation, there are usually expectations such as duration and progression. These concepts are best represented using events. For example, Headache is not just the state where pain is felt in the head. Headaches have an expected duration, an intensity, and a certain "inertia" (removing the cause doesn't instantly relieve all the pain). The pain experienced when hit in the head wouldn't usually be called "headache," though the impact is likely to cause one.

The coverage of common drugs includes types of products sold "over the counter" in grocery stores, types of drugs commonly prescribed by doctors for various symptoms and ailments, common recreational drugs and addictive/abused drugs, and otherwise "well-known" drugs. For each of these, the KB includes the drug's effect, typical dosage, and common forms (e.g., ointment, pill, liquid.)

Also entered into the KB was information about the "schema" of standard medical documents such as drug prescriptions, medicine labels, patient records, test result records and insurance claim forms.

The following are just a few sample

Figure 2. A CycL axiom that says "If someone has a sore throat, their throat hurts when they swallow."

```
(LogImplication
  (LogAnd
 (allInstanceOf $AIL SoreThroat)
 (bodilyInvolved $AIL $AGT)
 (temporallySubsumes $AIL $SWA)
 (allInstanceOf $SWA Swallowing)
 (performedBy $SWA $AGT))
  (holdsIn $SWA
 (feelsSensation (BodyPartOf $AGT Throat) LevelOfPain Positive)))
```

Figure 3. A CycL axiom that says "If someone is asphyxiating, they cannot breathe."

```
    (LogImplication
 (LogAnd
    (allInstanceOf $AIL Asphyxiation)
    (bodilyInvolved $AIL $AGT))
 (holdsIn $AIL
    (LogNot (behaviorCapable $AGT Breathing bodilyInvolved))))
```

axioms from this microtheory:

- If someone has a sore throat, their throat hurts when they swallow—see Figure 2.
- If someone is asphyxiating, they cannot breathe—see Figure 3.
- After major surgery, a patient will feel pain when performing strenuous physical activities.
- After being vaccinated for a disease, an animal is no longer susceptible to it.
- Bone repair is medically recommended for people with broken bones.
- An object which is sterile is not an infectionCarrier of anything.
- SurgicalTools are sterile before a surgery
- Patients are unconscious during general anesthesia.
- Patients don't feel sensations during general anesthesia.
- In any surgery that involves StitchingAWound or MakingAnIncision, some anesthesia is given.
- A medical consultation with a doctor typically costs between 20 and 200 U.S. dollars.

Table 1 presents some numbers summarizing the new terms and new axioms, which were added as part of this theory.

The CycL Language circa 1994

CycL's implementation has changed completely since 1990. The system is now available both in Lisp and C, thus enabling us to run Cyc on relatively inexpensive platforms such as Macintoshes and PCs. However, thanks to the implementation-independent semantics of CycL, this did not require a reworking of the entire KB. Over the last four years, there has been only one significant change to CycL: the addition of contexts.

Motivation and Summary of CycL's Context Mechanism

Expressions (such as English utterances), used for communication draw heavily on the context in which they occur to convey their intended meaning. However, expressions used for declarative representation in AI have assumed that there is no such dependence on context [2, 9].

A typical example of context dependence is the influence of the intended use of a KB on the representation [7]. Some of the aspects of the representation that are affected include:

- the vocabulary (i.e, the predicates, categories, individuals) used for the representation. For example, Mycin and Oncocin overlap significantly in their domains. However, unlike Oncocin, Mycin has little concept of extended periods of time. This is because these two programs are used for very different tasks.
- the granularity and accuracy of the theory.
- the assumptions. The assumptions that the task allows often lead to a simplification of the vocabulary to a point where the assumptions are no longer even *statable* within that theory. For example, it is not possible to state Mycin's assumptions about time being unimportant, about the patient being alive, about the availability of modern medical facilities, using mycin's limited vocabulary.
- Other factors that affect the representation include the spatio-temporal location of use (e.g., our advisor for choosing among 1990 cars was meant to be used then, not in 1994), the vocabulary used by the designers of the KB, for example.

The textbook solution to these problems is to design, in the first place, an extremely expressive vocabulary, use a very accurate theory, make very few assumptions, to make the representation as context-independent as possible. This, however, produces large complex theories that use extremely cumbersome vocabularies. These are very difficult to design, understand, and maintain [1], and therefore are generally undesirable. Even after these heroic efforts, the knowledge in the KBS will generally only be *partially* decontextualized, especially if it is about people doing things in the day-to-day commonsense world. Attempts at producing completely decontextualized representations, in such domains, are destined to be futile.

In sum, then, each particular theory we treat will inevitably incorporate much of this sort of "context dependence." The problem is that as we try to accrete ever larger sets of theories, as we do and so forth in Cyc, the various discrepancies in assumptions, vocabulary, across theories will cause the global accreted KB to become increasingly inconsistent.

In the past, this has served as a sort of barrier to how large knowledge-based systems could grow. The limit seemed to be around 10,000 rules (e.g., axioms, assertions). To break this barrier, each encoded theory in Cyc explicitly cites its assumptions. Indeed, the boundary between theories—between contexts—is generally due to a change in the assumptions being made.

How is this done concretely, in Cyc? Using reification, we say that a set of CycL sentences constitute a theory. Assertions (axioms, statements) in Cyc are not universally true; they are only true in certain contexts. For example, the sentence:

$$\mathbf{ist}(NTP,\ \forall x\ \neg supported(x) \supset falls(x))$$

says that in NPT (a particularly naive theory of physics), an object generally falls if it is unsupported. Cyc has more sophisticated physical theories, but often NTP suffices for everyday situations.

NTP is also termed a "context" or "microtheory" (which in turn is abbreviated "mt," so we might talk about the NTP mt.) The term "NTP" participates in assertions like "(ist NTP P)", meaning P is true in the NTP microtheory. NTP also participates in assertions that state various meta-level things about the theory: e.g., its scope, when it should or should not be used, assumptions it makes. For example, a few of NTP's assumptions are that the objects involved are not too many orders of magnitude larger or smaller than a breadbasket, are not moving too fast, are all on or near the surface of the Earth.

As another example, suppose P is the statement "an apple costs about 30 cents." Instead of adding P to the Cyc KB, we add (ist C_1 P) where C_1 is the context associated with the making of this statement P. Note that we are *not* attempting to completely decontextualize P. Doing that would

involve making explicit the time period over which the statement (that apples cost 30 cents) is true, where it is true, what exactly we mean by cost, to whom it will cost that much and so forth. Rather, we are simply recording the fact that P has certain context dependencies, a few of which are currently explicitly written down. This provides us with a hook we can use later for further (but still just partial) decontextualization. So C_1 is a "rich object" [10] in the sense that it cannot ever be completely described.

A syntax, semantics (i.e., a model theory) and proof theory for *ist* is presented in [4]. Due to space constraints, we will only briefly list here some of the logical properties of contexts and of *ist*:

- Within any context, we assume we have all of first-order predicate calculus (FOPC) available. This makes it possible to use conventional FOPC tools (including its proof theory) with contexts.
- Contexts are first-class objects and *ist* formulae are first-class formulae. That is, (ist c F), $(\forall c_i$ (ist c_i F)), $\neg$(ist c F), (ist C_1 F) $\wedge$ (ist C_2 F), (ist C_1 (ist C_2 F)) are all formulas (where c_i is a variable, C_1 and C_2 are constants and F is any formula.)
- Each context has a vocabulary associated with it. So, there might be some contexts in which P might not be statable (in the vocabulary of that context) and there might be yet other contexts in which P is stated differently. For example, C_1 might state P as (cost Apple (Cents 30)). C_2 might state P as $(\forall \times$ (apple $\times) \Rightarrow$ (cost $\times$ (Cents 30) USA). C_3 might not have the predicate cost at all, hence P would not be statable in C_3.[2]
- A statement might be true in one context and false in another. We would have another context C_4 where P is false. For example, C_4 might be about Depression-era America, when apples cost a lot less than 30 cents.
- The domains associated with different contexts (and therefore the scope of the $\forall$ and $\exists$ symbols) could be different. This is a direct consequence of different contexts making different assumptions. If C_1 makes the assumption that the spatio-temporal location of things was late twentieth century America, this implies that objects not satisfying this constraint will be excluded from the domain of C_1.[3]
- The same symbol might denote different things in different contexts. That is, *ist* is referentially opaque in its second argument. For example, we can have both ist(C_1 table (It)) and ist(C_2 person(It)) The constant *It* would obviously denote different objects in C_1 and C_2.

[2]If a statement α (not involving ist) involves vocabulary not in the context C_i, then we have $\neg$(ist C_i α), which of course is very different from (ist C_i $\neg\alpha$).

[3]The Barcan formula (ist c $(\forall x(p\ x))) \Leftrightarrow (\forall x$ (ist c (p x))) does not hold; in fact, it does not hold in either direction.

means that conventional FOPC problem solvers can be used, and used efficiently.

- . . . as the essential **indexical,** in the sense used by Perry [11]. The idea is the following. Though natural languages support many different indexicals (he, she, . . .), one can attempt to replace these with definite descriptions, thereby reducing the number of indexicals we have to deal with. However, Perry makes the point that it is not possible to completely do away with indexicals. We need at least one and this he calls the essential indexical.

Table 1. New terms and axioms added as part of Ailments microtheory

Topic	new terms	new axioms
Anatomy	8	53
Ailment	20	77
Capability	0	20
Medical care	38	63
Medical costs	8	20
Drugs	50	88
Medical equip	27	90
Facilities	20	100
Insurance	16	42
Medical people	20	90
Reproduction	24	87
Toiletries	0	16
Total	231	726

At any time, Cyc is *in* a context. The concept of the system being *in* a context is defined . . .

- . . . in terms of interactions with the system. Being "in" the context called C_1 and then stating P is equivalent to being in the outer context and stating (ist C_1 P);
- . . . in terms of problem solving. When the system is in the context C_1, the only axioms available to the problem solver are those axioms P such that (ist C_1 P) is true in the outer context. So, if the system is "in" context C_1, most (if not all) of the formulas available do not involve *ist*, which

Kinds of Contexts

Though there is a single logical machinery associated with contexts, there are many different kinds of contexts and, correspondingly, many different kinds of uses of contexts. In general, contexts provide a means of referring to a group of related assertions (closed under entailment) about which something can be said. Such a group of assertions might form one or more of the following:

- A general theory of some topic. For example, a theory of mechanics, a theory of the weather in winter, a theory of what to look for when buying cars. Contexts used in this sense are called "Microtheories." Microtheories are usually rather large (hundreds or thousands of axioms) and are long-lived.

Different microtheories make different assumptions and simplifications about the world. Contexts provide a mechanism for recording and reasoning with these assumptions. For any given topic, such as "the weather," there may be different microtheories of that topic, at varying levels of detail and generality.

Also, by keeping different theories distinct, the problem of maintaining consistency is transformed from that of maintaining global consistency to maintaining local consistency, which in practice is vastly simpler and faster. This becomes especially important as the total size of the KB increases.

- A representation (of some situation) that is tailored for the problem it was set up to solve. For example, a model of a Christmas tree as a perfect cone, used for determining whether it will fit in a given space in a store window or car trunk; a model of an object as a point mass for determining its trajectory. Contexts specific to a particular problem-solving task are called Problem Solving Contexts (PSC). A problem-solving task might involve answering a single question or a number of related questions. These contexts are usually created dynamically by the system and are ephemeral.

- By collecting a small relevant subset of the KB into a bundle, they provide a mechanism for focusing on relevant information during problem solving. Often, a PSC is created dynamically, has many axioms from several Microtheories "imported" into it, some problem solving goes on "in" that PSC, an answer is produced and "exported," and the PSC is soon forgotten and discarded.

- A "very slightly decontextualized" representation of the utterances made in a discourse. For example, a representation that retains anaphoric and indefinite references in a conversation between two people. In such a context, a phrase such as "the person" might not be translated to represent the actual individual referred to, but may be represented by using a term like "(The Person)." Within any one given context, "(The Person)" denotes a unique individual, though it will of course denote different individuals in different contexts. Such contexts are called Utterance Contexts. As with PSCs, these are also often short-lived and relatively small.

The microtheory mechanism has proved extremely useful in the past several years. It allows Cyc to have and use theories at different levels of granularity, fictional contexts, discourse contexts, problem-solving contexts for example. It also speeds up knowledge entry into Cyc, as one can be quite terse and seemingly ambiguous in asserting new axioms to Cyc about the context which it is "in" at the moment. It also speeds up inference, as much everyday problem solving occurs "in" a context—and the typical Cyc context excludes 99.9% of Cyc's terms (and hence assertions involving those terms) from consideration. For example, when trying to find your car keys, you shouldn't have to (even briefly) consider how many legs an arachnid has.

From the point of view of a single axiom, a context is a sort of encapsulation for that axiom, capturing the assumptions, indexicality, and anything else that might be implicit in the axiom for it to carry its intended meaning. Fortunately, large bodies of axioms turn out to share the same context.

Relative Decontextualization, or "Lifting"

This is only half the story on contexts, of course. Keeping otherwise-contradictory axioms insulated from one another is all well and good, but the other half of the story is finding ways to import or translate or "lift" axioms from one context into another, so they can participate in inferences together. To do this "lifting," Cyc must perform a "relative decontextualization" of the imported axioms, as necessary.

When lifting an axiom from one context to another, the ontologies (vocabulary and languages) and assumptions associated with the origin context and target context are likely to be somewhat different. These differences need to be factored in during the lifting so that the lifted axiom means *more or less* the same thing in the target context as it did in the origin context. "Lifting" needs to be as *meaning-preserving* as possible. For a model-theoretic definition of "meaning-preserving relative decontextualization," see [4]. The following short example illustrates this concept.

One of Cyc's microtheories, *WorkplaceCodeOfConductMt*, describes the behavior of individuals while at work. This microtheory assumes that any "people" mentioned in its axioms are sane adult human beings. One of the axioms of the theory says "in office settings, people generally don't make loud noises." Now suppose that Cyc is told about a situation in which a baby is brought into an office by a parent; that is, Cyc is asked to reason about this situation in a new problem-solving context (PSC_1.) Since the infant does not satisfy the "sane adult" assumption made by the context WorkplaceCodeOfConductMt, Cyc will *not* predict that the infant will remain quiet. On the other hand, Cyc will predict that the parent—who is still assumed to be sane and adult—is not likely to loudly scold or shush the child, even if the child is loud.

Notice that this illustrates how the denotation and scope of $\forall$ and $\exists$ in different contexts is typically different, and axioms containing variables have to be appropriately modified during lifting.

A different sort of transformation that Cyc must often make, as part of "lifting," is one of change of vocabulary. For example, in one context, "cost" might be a binary predicate, but in another context it might be ternary. One context might implicitly temporally scope the information contained in it (e.g., a 1990 automobile selection advisor; a medical diagnosis program that assumed all tests were performed at more or less the same time, not days apart.) A context (especially an utterance context) might use indexical terms such as "Now" and "He."

The Default Coreference Rule (DCR) is one of Cyc's heuristics for lifting. The intuition behind the DCR can be explained by the following analogy. Consider a physicist and a

geologist talking to each other about some problem. There will be differences in the way each looks at the problem. These differences will be reflected in the vocabulary and language they use. When physicists say that some process is fast, they mean something different than when the geologist says that a process is fast. However, despite the differences in their languages, there is more in common between the two languages than there are differences between them. They refer to the same concept when they say "year" or "Earth" or "molten" or "mass." Since they refer to the same concepts, they share most of the axioms associated with these concepts.

We are saying that most people share not just most of the terms in their vocabulary, but also most of the axioms associated with these terms. If they shared too little, communication would be impossible; if they shared too much, communication would be unnecessary.

Cyc's situation is similar. Though there are differences between contexts, those are the exceptions, not the norm. Most of the time, most of the meaning of most of the terms are shared by most of the contexts.

The DCR consists of two parts. The first part states that most terms retain their denotation across contexts. The second part states that, as a default, there is a maximal overlap between the denotations of a predicate (or function) symbol across contexts. If Cyc knows (P a) (where *a* is a constant) is true in context c_1, then as a default (P a) is believed to be true in every other context c_2.[4] Correspondingly, if a nonatomic term (F g) has a value *h* in c_1, then as a default, it has the same value in c_2.[5]

Contexts and Interagent Communication

The context mechanism described forms an important part of any substrate upon which interagent communication takes place. It is inevitable that any given pair of agents will have significant differences in their vocabulary, and—even where their vocabularies overlap—they will associate slightly different meanings with each term.

It is important to be able to allow for a partial sharing of the meanings of most terms, that is, to not be forced to either assume they mean exactly the same thing or assume they are entirely different. The context-based approach of "sharing most of the meaning most of the time" provides Cyc with exactly this functionality.

Contexts and Inference

The context mechanism (most specifically, the notion of Problem Solving Contexts) also plays a very important role in speeding up inference. The primary observation on which this is based is the following: the world is almost completely hierarchically decomposable. For solving any one problem, most of the things (and hence knowledge about those things) in the world are irrelevant. The only relevant objects are those that are "proximate" (and typically, this means spatio-temporally proximate) to the situation at hand. So, if one were trying to determine whether to go to a certain restaurant, the number of U.S. Supreme Court Justices is completely irrelevant. In fact, the inference engine would be better off not even being aware of (having to consider even momentarily) that fact.

There are two factors that make inference harder as the size of the KB increases: (1) the total number of axioms and; (2) the number of possible bindings for each of the variables in these axioms. Though the total number of *axioms* relevant to the task may be extremely large, it is unlikely that the total number of *objects* that should be considered as bindings for the variables in these axioms is equally large. So, what we need is some form of focusing mechanism to encapsulate the situation about which questions are being answered in order to consider only the objects in the situation (or at least give them priority) as potential bindings. This is exactly what CycL's context mechanism—more precisely, the Problem-Solving mechanism—does.

Cyc-Based Natural Language Processing

Being able to understand texts is an important part of the Cyc long-range plan. In the long run, the only way that Cyc is going to be able to keep abreast of any reasonable fraction of the happenings in the world is by reading texts. Even in the short run, the ability to understand on-line texts can be used as the basis for a tool that will amplify human KEs (knowledge enterers) as they continue building and testing Cyc. For example, when a KE is entering knowledge about medicine, this tool should be able to consult an on-line encyclopedia or dictionary and make a list of medical instruments (along with their uses) and semiautomatically enter them into the KB (i.e., after some inspection and perhaps disambiguation and elaboration by the KE). Preparing such a list could even be done with only a very partial understanding of the text.

We now describe the Cyc-based natural language understanding program that we have been building over the last year. Our methodology has been empirical and corpus-oriented; we take up text corpora and systematically go through them, modifying the system to handle them. As with other parts of Cyc, we have borrowed heavily from many different approaches—in this case, many different linguistic theories. We should point out that our focus is on translating utterances into CycL, not on cataloging or explaining linguistic phenomena. It is also not our goal to show that any one tool (including Cyc) is adequate, or even required, to do 100% unrestricted NLU (natural language understanding).

Anatomy of the Natural Language System

The Cyc-based natural language system, CNL, consists of four major modules, each of which we describe in turn: lexicon, syntax, semantics, and postsemantics.

- Lexicon. The lexicon not surprisingly contains information about

[4] Assuming of course that (P a) is a legal term in c_2; in particular, both "P" and "a" must exist in the ontology of c_2, P must still be a unary predicate there, etc.

[5] Again assuming that the term (F g), and also *h*, are syntactically legal in context c_2.

words. For each word, it contains the different morphological variants of the word and their lexical properties (such as their part of speech). It also specifies the mapping of different word structures into Cyc expressions. The lexicon is currently part of the main KB; it is encoded in the form of Cyc assertions, just like any other assertions.

• Syntax. The syntax module currently consists of a set of rules written in a context-sensitive grammar. Its function is somewhat less than that of the syntax modules in more traditional NL (natural language) systems; its goal is *not* to come up with complete parse trees. Rather than trying to resolve issues like attachment (which are better dealt with in semantics), the Syntax Module's goal is merely to identify parse structures with sentence fragments and present the results in a neutral—hence ambiguity-preserving—fashion.

For example, consider the sentence: "He killed the girl with a branch." The output of the syntax module is the following set of "parselets":

(subjectOfVerb [killed] [He])
(objectOfVerb [killed] [girl])
(article [girl] [the])
(article [branch] [a])
(withEP ref [branch])

"[branch]" is a complex structure that records the set of word-senses of the word "branch" that are syntactically consistent with the preceding sentence. Similarly for the other [...]'s.

"withEP" is a predicate that points to the various meanings of the word "with" in English (instrument, possession,...) along with linguistic knowledge about constraints on these meanings.

"ref," in the last parselet, is a dangling referent. In the Semantics (and, if necessary, Post-Semantics) stage of processing, the Cyc KB is called on to infer the most plausible meaning of "with," and thus to determine what the dangling referent "ref" most likely refers to.

Techniques from several different approaches (e.g., feature-based grammars, unification grammars, phrase-structure grammars) have been used in the syntax module. We plan to soon incorporate large-corpus-based statistical techniques to help in guessing the parts of speech of a word.

• Semantics. The goal of the semantics module is to transform the output of the syntax into a CycL expression that preserves most of the discourse context dependencies of the utterance. The output of syntax is converted into a set of word-structures of word-types such as

(<Qualifier > <NP >),
(<Ditransitive-Verb > <NP >
<NP > <NP >).

For each word and word-type, the lexicon contains CycL expressions (possibly as a function of the terms that might appear in the word-structure in which that word-type occurs). These CycL expressions are combined according to a set of rules (that are reminiscent of Montague's rules), to provide the overall CycL translation. While doing this, the semantics module also has to resolve the different ambiguities that might have been left by syntax. For doing so, semantics might consult not just the output of syntax but also the "raw" input utterance itself.

The output of the Semantics module is ready for inclusion into the utterance context corresponding to the utterance. We heavily exploit the context abilities of Cyc here. The translation at this point still contains indexical terms such as "He," "It," (The Man).

Consider our example again: "He killed the girl with a branch." After attachment guesses, this module has produced:

(([killed])(normal-verb [He] [killed]
[girl]) (withEP [girl] [branch]))
(([killed])(normal-verb [He] [killed]
[girl]) (withEP [killed] [branch]))

Figure 4 illustrates what is handed to the Post-Semantics module after this stage, now in order of decreasing preference:

Note that some parses have already been eliminated for semantic inconsistency. For example, ...(accompaniedBy $x (AEF Branch-Tree)))

• Post-Semantics. In this phase, CNL disambiguates the references of indexicals, such as "yesterday" and "He," when possible, and also makes task-dependent modifications to the translation. Suppose the user is captioning images, and inputs "A girl sitting on a sofa"; then this is the stage at which Cyc would explicitly record the fact that this is a description of a photograph (i.e., that a certain photo depicts a girl sitting on a sofa).

In the case of "He killed the girl with a branch," suppose that prior utterances have set up a context in which there are three girls walking side by side, and one of them is holding a delicate, valuable golden branch. Then the Post-Semantics module would reverse the Semantics module's choice about "...with a branch" meaning instrument (of the man), in favor of it meaning something being held (by the girl who is holding it.)

To give a rough idea of where the system stands as of March 1994, the lexicon has about 12,000 root words (where all the different variants of a word such as "bear" would come under one root). The Syntax module can properly handle about 75% of the sentences found in the news stories of a typical issue of the newspaper *USA Today*. And in cases in which Cyc knows all the proper nouns in the sentence, the Semantics module can properly handle most of the sentences parsable by the syntax module. By "properly" here we mean that the output is as good as what our knowledge enterers independently come up with, when asked to manually translate the material into CycL.

Cyc Applications

Here we examine criteria for what makes for good Cyc applications, then take a brief look at several recent such applications.

Criteria for Good (early) Cyc Applications

Though Cyc is a very generic tool, tasks that make for good Cyc applications (at least for the next few years)

all share the following properties:

- **Low infrastructure cost.** Until Cyc has proved itself in other areas, it would be unwise to commit to a Cyc-based application that required an enormous capitalization or retooling. Using Cyc in every classroom in a school district might be great, but would require each classroom to have a computer running Cyc.
- **Incremental use of Cyc.** The application should be a "fail-soft" one: if Cyc can recommend something, great. Otherwise, the program carries on much as the existing software would, for that task.
- **Off-line.** Choosing an application in which rapid real-time response is required is just creating one more obstacle to having a Cyc-based application succeed.
- **Moderately "stylized" use.** That is, the application program's use of Cyc should be characterizable by some small number of high-frequency query types, which can be optimized, if necessary, by adding one or more new heuristics for rapid inference.
- **Need for common sense.** There may be little need for Cyc, if the task is very technical and/or very narrow and/or very simple, and/or very highly stylized. In such a case, it might suffice to use an expert system, or to write a custom non-AI application program.
- **Centralized running.** Ideally, the application should only have to run on one centralized machine (e.g., one machine running Cyc in each supermarket). If every end-user device must run Cyc, that would eliminate a vast fraction of the market for the next 3 years or so (i.e., until the average low-end PC was at about the mid/high-1994 level in speed, memory, and disk).

Recent Prototype Applications

There are two broad tasks that we have decided to focus on for the first generation of Cyc applications.

- Person modeling: The basic idea here is to put together a model of a person (or organization, etc.) based on several (possibly disparate) pieces of information the system might have about that person such as their family, their job, their interests. Such a model can then be used for various purposes such as identifying new postings that might be of interest to them or product offerings they might be interested in. Here are two examples of suggestions such an application might provide:

Let us suppose that Fred, a math teacher, has recently had surgery and is confined to his bed for a month. The system comes across an advertisement for a used chess computer. Given Fred's situation, it is quite possible that he might be interested in seeing this offering. However, the system should know enough not to bring advertisements for volleyball nets, for example, to his attention, even if he is known to be an occasional volleyball player. Making (or not making, as the case might be) these connections does not involve very sophisticated inference, but requires knowing something about a very significant range of topics—something that only Cyc is good at.

After he gets out of the hospital, Fred decides to take a vacation in Hawaii. The system comes across an announcement for a big conference (on a topic that has nothing to do with Fred) that is going to be held in the same hotel where Fred is planning to stay, at the time Fred is going to be there. This is now something that Fred would be interested in (he might consider changing hotels, for instance). Again, this does not involve any sophisticated inference, but draws significantly on the breadth of knowledge that Cyc has.

- Information Access: Information access has traditionally been very syntactic in the following sense. The user specifies some set of words (as part of a command) and a search of some sort is conducted for the occurence of these words (the actual pattern of words searched for is a function of the command). This approach covers everything from simple string-search, to Boolean combinations of keywords, to spreading activation using generalization/specialization trees and a thesaurus, to retrieving information using SQL from relational databases. This approach works fine if the user is aware of at least the conceptual "schema" (if not the actual terminology or field names) of all the Information Bearing Objects (IBOs) being searched. Often, the user must in fact master the literal schema itself. For example, if one is trying to find out who the emergency contact of a colleague is and that information is stored in a database under the column title "emrg-reln" (and the user does not know this or which database this is in), it is very unlikely that they will actually find the relevant information.

This syntactic approach to information retrieval is based on a superficial imitation of the way communication between humans works: by the exchange of words. The difference with humans is that these words are "understood" in some fashion before processing. Imitating this in our programs, except at the most superficial level, is not as easy. In order to have even a shallow understanding of the

Figure 4. The output of stage 2 of the Cyc-based NL system, ready to be handed to the Post-Semantics module.

```
   (LogAnd
(allInstanceOf $x KillingSomeone)
(performedBy $x He)
(objectActedOn $x (TheEF Girl))
(toolsUsed $x (AEF Branch-Tree)))

   (LogAnd
(allInstanceOf $x KillingSomeone)
(performedBy $x He)
(objectActedOn $x (TheEF Girl))
(holds (TheEF Girl) (AEF Branch-Tree)))
```

plethora of words that might be used to access information, the system needs to know about a lot of different things. Going back to our previous example, in order to understand what "emergency relation" means, the system has to have such concepts as "person," "relation," "emergency." If however, the system *does* know all this, a very different approach can be taken to the problem of retrieval—Semantic Information Retrieval (SIR). The SIR-based system keeps track of what information is available where at a semantic level (i.e., in terms of what the data really means instead of just as a series of words), interprets queries similarly, and provides the access based on this understanding.

In the following subsections we describe two prototypes we have built, in which Cyc is used to help do Semantic Information Retrieval tasks.

Cyccess: Smart DBs and Spreadsheets

Cyccess is a prototype application involving the use of Cyc with structured information sources (SIS) such as databases and spreadsheets. Cyccess uses Cyc to understand the contents of an SIS to provide services such as:

- Consistency Checking: Data in the SIS can be checked for consistency with basic common sense. For example, it is not reasonable for a person to be employed before they were born, or for an automobile to be in two countries at the same time.
- Information Retrieval: This is as described earlier.

The key to understanding how Cyccess works is to understand how we provide Cyc an "understanding" of the contents of an SIS. Consider the SIS shown in Table 2, which we'll refer to as PT1.

For the sake of simplicity, we will only go through this procedure for something akin to a relational database, but this treatment can be extended for more complex structures as well.

- Create a new context for PT1. The implicit temporal (and possibly other) assumptions of the database are represented as assumptions made by this context.
- Communicate to Cyc that the person referred to in the "HUSB/WIFE" column is the spouse of the person referred to in the "name" column. How will we do that? Let "(Cell i j)" refer to the thing denoted by the entry in the cell corresponding to row i and column j of PT1. We now make the following assertion to Cyc:

(meaningFormulaPT1 (spouse (Cell i 1) (Cell i 4))).

That's it. Cyc now knows the meaning of the "spouse" column. Axioms such as these are called "articulation axioms" or, as we have previously seen, "lifting axioms."

Once Cyc has an axiom like this for each column in PT1, along with the network address and format of PT1, the entire contents of PT1 can be thought of as "virtually" in Cyc. The contents of the PT1 database table may be retrieved by CycL queries of the form (spouse x y). Note that writing the articulation axiom does not cause the entire contents of the database table to be immediately brought into Cyc. Rather, during querying, there is a sort of "database fault" (quite analogous to a "page fault"), where Cyc generates an appropriate query (in SQL or some other relevant format) to the SIS and retrieves the information. Once brought in, the information is no different than any other assertions that have been made "by hand" to Cyc; all of Cyc's knowledge can be applied to determine whether the information is consistent, use the information for performing other inferences, etc. If we later Ask for MaryJones' husband, Cyc will know the answer is FredJones; if we Ask whether MaryJones has any children, Cyc will argue (guess) that the answer is yes, since FredJones does; if we Ask whether Mary is less than 22 years old, Cyc will argue that the answer is no, since she (arguably) has a child old enough to serve as an emergency contact; and so on.

The traditional approach to performing this kind of integration between DBs has been to write translation rules that map from one DB to another. This is an expensive task, however, requiring N-squared sets of translation rules for N DBs. The Cyccess approach is much less expensive, requiring only N sets of translation rules for N SIS's (one set of rules that maps each SIS to Cyc).

Note that in order to retrieve information from an SIS, one simply poses a query to Cyc, in CycL, and the information is retrieved quite independent of where it is, what schema the DB has that is storing it, etc. We do (Ask (spouse FredJones $x)), and we need have no idea that Cyc might be accessing PT1, let alone that Cyc will be issuing some SQL query involving the field HUSB/WIFE.

In the current Cyccess prototype, one can, alternatively, construct certain queries just by clicking appropriately on an empty cell in one SIS; this in effect generates a CycL query (corresponding to the meaning formula of that cell), asking Cyc to try to infer (guess) a value for that empty cell. That inference process might involve other cells in the same SIS, cells in other SISs (whose existence the user might be unaware of), and other axioms anywhere in Cyc.

The Cyccess prototype can also be asked to consistency-check the entire SIS. To do that, the information in the entire Structured Information Source is then actually "lifted" into Cyc's KB, checked for consistency

Table 2. Structured information source PT1

NAME	PHONE	BIRTH	HUSB	WIFE	EMRG-CTCT	EMERG-RELN	PHONE
FredJones	x3421	9/1/59	MaryJones	KatyJones	daughter		831-5052
°							
°							
°							

with itself, with common sense, and with other specific knowledge Cyc has, and the cells containing potentially bad information are passed back to Cyccess which then (in its current implementation) gives them a pastel tint to draw the human's attention to them.

Cyccess and Enterprise Integration

Enterprise Integration—the idea that different business processes (both inter- and intra-business) can be better integrated by sharing information—has become quite popular in recent years.

The standard approach to enabling sharing of information has been to impose standards on how this information is to be stored. There are numerous efforts under way for standardizing such things as "patient records" and "product descriptions."

There are many problems with trying to standardize. As we mentioned at the very beginning of this article, given so many different players, it is unlikely that everyone is going to be willing to adopt a single standard. And even if they do, it is unlikely the standard will be able to predict all the information needs of the enterprises involved and incorporate them all, especially over periods of years and decades.

Cyccess exemplifies a very different approach to this problem. Rather than standardizing the schemas themselves, the idea is to use one very expressive metaschema (i.e., Cyc) as an interlingua to translate among different schemas.

Image Retrieval

While the word-based approach is marginally adequate for retrieving textual data, it completely fails for images, videos, and sound. The Cyccess approach however does not depend on the IBO (information-bearing object) being textual, and can therefore be extended to cover such sources of information as well.

Consider the task of finding images relevant to some query. Current word-oriented searching techniques, involving Boolean combinations of keywords, thesaurus lookup, for example, are just barely adequate with today's 100,000-image libraries. They will fall woefully short when the next few years make libraries 10, 100, or even 1,000 times that size available.

In the current Cyc prototype of this application we have on-line a database of digitized still images and videos, each of which has a "caption." The captions are entered in English, and then CNL (or a human knowledge enterer) translates them into CycL axioms.

Similarly, querying is done by issuing CycL queries (through a NL interface) and CycL performs matching between the captions and the images, performing inference as necessary. Let us consider an example of how this works.

Consider an image of a girl lying on the beach, which is unsurprisingly captioned with the English sentence "A girl is lying on the beach." This sentence is converted to CycL and stored in Cyc. Months later, someone poses the query "show me images of people at risk of getting cancer." This is then translated into CycL and issued as a query and the system responds with the earlier captioned image. The inference involved in determining that the girl is sunbathing and thus at risk of getting cancer is quite simple. However, if the system were not able to go from the fact that she is lying on the beach to the likely conclusion that she is sunbathing, to skin cancer, there is no way in which it would have been able to do the retrieval.

Another example is when one asks the system for images evoking cuteness. It first asks whether one means cute as in attractive or cute as in heart-warming. After the latter is chosen, it finds many such images. A typical one is captioned "A dog being carried in a backpack." Note that no element of the picture—dog, carrying, backpack—is inherently cute, but Cyc knows that animals in unusual places are cute.

This semantic image retrieval prototype illustrates how relatively shallow knowledge about a very large spectrum of topics can change the way information retrieval works.

Semantic File Systems

The two previous Cyc-powered prototypes (Cyccess and Image Retrieval) illustrate two pieces of what might be called "a Semantic File System."

The idea here is that instead of organizing all of one's IBOs in some kind of tree structure (as we currently do on our hard disks, for example), one simply captions all of one's files and then just drops them into a "pile." Having done this, retrieval is done by asking for the IBO "by content." The pile could include not just files, but such things as one's email messages.

In fact, consider that everyone may very well elect to give up privacy on 10% of their personal files, in exchange for access to 10% of the files of everyone else in the world. In that situation, the current mnemonic tricks we use for naming our own files, searching for words or phrases we recall putting somewhere in them just won't work at all, and something like the semantic file system will become a necessity.

Furthermore, with a semantic file system, retrieval could be proactive. The system could choose to bring some IBO (e.g., snippet of a file, email message, cell in a spreadsheet, captioned image, notice posted on a bulletin board) to the user's attention based on its model of the user, the current task they are working on, and so on.

Conclusion

After almost a decade, the Cyc project is still on target. The CNL (Cyc-based NL understanding and generation) subsystem is developing synergistically with the Cyc KB, and we expect a sort of crossover to occur in the next two years, by which we mean that most of the knowledge entry will take place by semiautomated NL understanding, with humans able to take the role of tutors rather than brain surgeons.

Hardware has kept pace with our hopes (and Moore's Law), and Cyc is now running in C on standard platforms (e.g., Sparc-10s, Mac-IIs).

The first few Cyc-based applications are emerging, at least in prototype form, and they are extremely

encouraging. Their nature is a bit different than what we expected in 1984—namely a focus on helping with information management rather than helping expert systems be less brittle—but if anything that means their potential impact is going to be even larger than we expected.

The prospect is there, finally, for Cyc to be the vector of intelligent agents in the next few years. We invite collaborators who will help us work toward that exciting goal.

Acknowledgments

We thank all the members of the Cyc group, past and present, for their help in building and testing the system. We would like to specially single out Kate Joly for her work writing much of the CNL grammar. Dexter Pratt and Nick Siegel played leading roles in developing the Image Retrieval application. Karen Pittman and Mark Derthick are responsible for the current state of the Cyccess application. Keith Goolsbey and David Gadbois have played key roles in making Cyc available in C. Cyc is supported in roughly equal parts today by seven organizations, and we would like to take this opportunity to acknowledge and thank them: Apple, Bellcore, DEC, DoD, Interval, Kodak, and Microsoft.

References

1. Falkenhainer, B. and Forbus, K. Setting up large-scale qualitative models. In *Proceedings of the Seventh National Conference on Artificial Intelligence*. Morgan Kaufmann, Palo Alto, Calif., 1988.
2. Feigenbaum, E.A. The art of artificial intelligence. In *Proceedings of IJCAI-77*, Morgan Kaufmann, Palo Alto, Calif., 1977.
3. Guha, R.V. The representation of defaults in Cyc. Tech. Rep. ACT-CYC-083-90, MCC, Feb. 1990.
4. Guha, R.V. Contexts: A formalization and applications. Tech. Rep. ACT-CYC-423-91, MCC, Austin, Texas, 1991.
5. Guha, R.V. and Lenat, D.B. Pittman, K., Pratt, D., Shepherd, M. Cyc: A midterm report. *Commun. ACM 33*, 8 (July 1990), 30–49.
6. Laird, J., Newell, A., and Rosenbloom, P. SOAR: An architecture for general intelligence. *Artif. Intelli. J. 33*, 1 (Sept. 1987), 1–64.
7. Lenat, D.B. and Brown, J.S. Why AM and eurisko appear to work. *Artif. Intelli. J. 23* (1984), 269–294.
8. Lenat, D.B. and Guha, R.V. *Building Large Knowledge Based Systems*. Addison-Wesley, Reading, Mass., 1990.
9. McCarthy, J. Programs with common sense. In H. Levesque and R. Brachman, Eds., *Readings In Knowledge Representation*. Morgan Kaufmann, Los Altos, Calif., 1986.
10. McCarthy, J. and Hayes, P.J. Some philosophical problems from the standpoint of AI. In M. Ginsberg, Ed., *Readings In Nonmonotonic Reasoning*. Morgan Kaufmann, Los Altos, Calif., 1987.
11. Perry, J. The problem of the essential indexical. *Nous 13* (1979), 3–21.
12. Quine, W.V. Natural kinds. In *Ontological Relativity and Other Essays*. Columbia University Press, New York, 1969.

About the Authors:

R.V. GUHA is coleader of the Cyc effort at Microelectronics and Computer Technology Corporation. He has authored several important papers and technical reports, including a book and various other publications on Cyc. email: guha@mcc.com

DOUGLAS B. LENAT is principal scientist at Microelectronics and Computer Technology Corporation. He has led the Cyc development team since 1984. He is a prolific author, and has been a faculty member in the Stanford University computer science department since 1978. email: lenat@mcc.com

Authors' Present Address: Microelectronics and Computer Technology Corporation (MCC), 3500 West Balcones Center Drive, Austin, TX 78759.