AFRL-IF-RS-TR-2002-209
Final Technical Report
August 2002

LEVERAGING CYC FOR THE HIGH PERFORMANCE KNOWLEDGE BASE (HPKB) PROGRAM

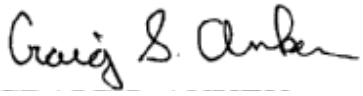
Cycorp

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. F105

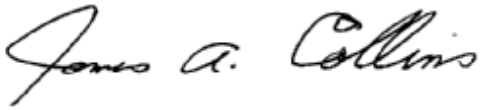
AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2002-209 has been reviewed and is approved for publication

APPROVED: 

CRAIG S. ANKEN
Project Engineer

FOR THE DIRECTOR: 

JAMES A. COLLINS, Acting Technical Advisor
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	AUGUST 2002	Final Jun 97 – Oct 01

4. TITLE AND SUBTITLE
LEVERAGING CYC FOR THE HIGH PERFORMANCE KNOWLEDGE BASE (HPKB) PROGRAM

5. FUNDING NUMBERS
C - F30602-97-C-0182
PE - 62301E
PR - IIST
TA - 00
WU - 05

6. AUTHOR(S)
Douglas B. Lenat and Mary A. Shepherd

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Cycorp
3721 Executive Center Drive, Suite 100
Austin Texas 78731-1615

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

AFRL/IFTD
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2002-209

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Craig S. Anken/IFTD/(315) 330-2074/ Craig.Anken@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

This work was part of the DARPA High Performance Knowledge Base (HPKB) program. The work described in this final report has focused on providing to the HPKB program the robustness and effectiveness of common sense knowledge as embodied in the Cyc knowledge base. Its objective was to provide intermediate level knowledge necessary to tie together high level, abstract knowledge and low level application specific knowledge to ease integration of knowledge bases and provide more efficient and more powerful inferencing mechanisms. The pre-existing Cyc KB had tens of thousands of useful rules for HPKB Integrated Knowledge Base (IKB) to inherit, and the Cyc team had already analyzed the "perennial conceptual issues" for thirteen years prior to HPKB. Early adoption of Cyc's Public Upper Ontology as the "HPKB Jumpstart Ontology" gave both the Cycorp and SAIC teams a uniform, convenient, and reliable environment to add knowledge, ask questions and gather measurements.

14. SUBJECT TERMS
Knowledge Base Technology, Artificial Intelligence, Information Technology

15. NUMBER OF PAGES
31

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

LIST OF FIGURES

Introduction

The work described in this final report has focused on providing to the High Performance Knowledge Base (HPKB) program the robustness and effectiveness of common sense knowledge as embodied in the Cyc knowledge base.

The Cyc project has spent the past seventeen years developing appropriate data structures and algorithms to represent and efficiently handle common sense knowledge. The pre-existing Cyc KB had tens of thousands of useful rules for HPKB Integrated Knowledge Base (IKB) to inherit, and the Cyc team had already analyzed the "perennial conceptual issues" for thirteen years prior to HPKB. Early adoption of Cyc's Public Upper Ontology as the "HPKB Jumpstart Ontology" gave both the Cycorp and SAIC teams a uniform, convenient, and reliable environment to add knowledge, ask questions and gather measurements.

There were two main goals of the Cycorp High Performance Knowledge Base (HPKB) effort.

1. The first goal was to build libraries of *intermediate-level* ontologies. Given the HPKB goals of breadth and flexibility (rapid accommodation to changes), we felt there was much value in codifying the intermediate levels of knowledge: the background conceptual structure needed for battlefield- and crisis- modeling and analysis. For example, concepts such as "pinning a force that's guarding a critical resource" lies far below a concept like "Tactic" and far above the level of tactics specific to particular weapons systems, particular opponents, particular terrains, etc. See Figure 1 below.

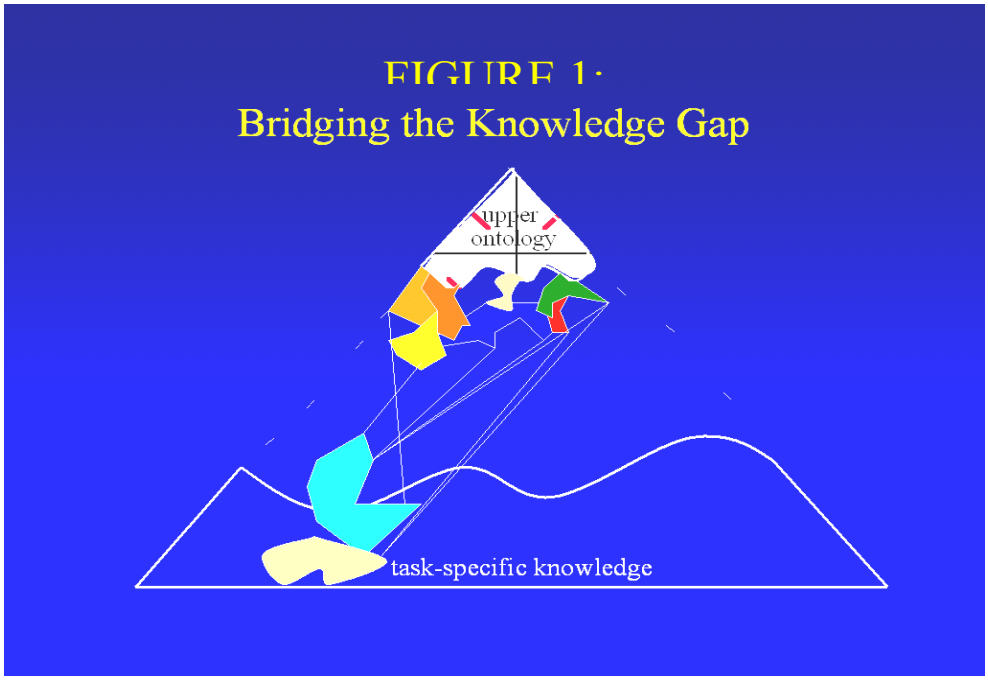


Figure 1

Result: The HPKB participants (on the Teknowledge "team") were able to effectively leverage this intermediate layer of ontology and knowledge, to do rapid model creation and revision. See Figure 2, below, for how much power this actually added. In our original proposal, we estimated that we would need to create about 200 new separate micro-theories, covering many aspects of the physical environment, devices, human limitations, etc. During the execution on our HPKB contract, we ended up creating over 800 new micro-theories. Together, those new micro-theories comprised about 300k axioms interrelating 50k concepts. In addition to working bottom-up, Cycorp was able to work top-down by extending the large existing Cyc ontology: approximately 60% of those 200k axioms and 20k concepts were already in the Cyc Knowledge Base.

2. Our second goal was to make it possible for machines to perform efficient reasoning on large Knowledge Bases (KBs). In the past, the techniques for efficiently building, editing, browsing, and reasoning on a 5,000-rule knowledge base had not been able to scale up to 100,000+ rule KBs. We extended our existing Cyc inference engine code, developing new tools and techniques that enabled practical, resource-limited operation on huge KBs, at least for the classes of reasoning most commonly needed for HPKB problems. I.e., we developed new low-level search guidance heuristics and new high-level special-case reasoning modules that were relevant to the HPKB target applications.

   As in Task 1, we took an engineering approach to this task, the development proceeding incrementally and driven by the target applications, not by aesthetic concerns or ideology. To enable others to use just as much of the results as was appropriate, we created three quite different deliverables:

   (a) a formally specified API (Application Program Interface) to maximize independent development, which we posted (and still have posted) on our website;
   (b) a text repository documenting useful ideas, techniques, algorithms, data structures, etc., written up in prose, PowerPoint slides, periodic project status reports, and journal articles; and
   (c) an executable piece of software embodying those ideas and adhering to that API. This code, referred to as the integrated knowledge base (IKB), was distributed free of charge to all the HPKB participants to use as a sort of interlingua or reference ontology, inference engine, and interface suite.

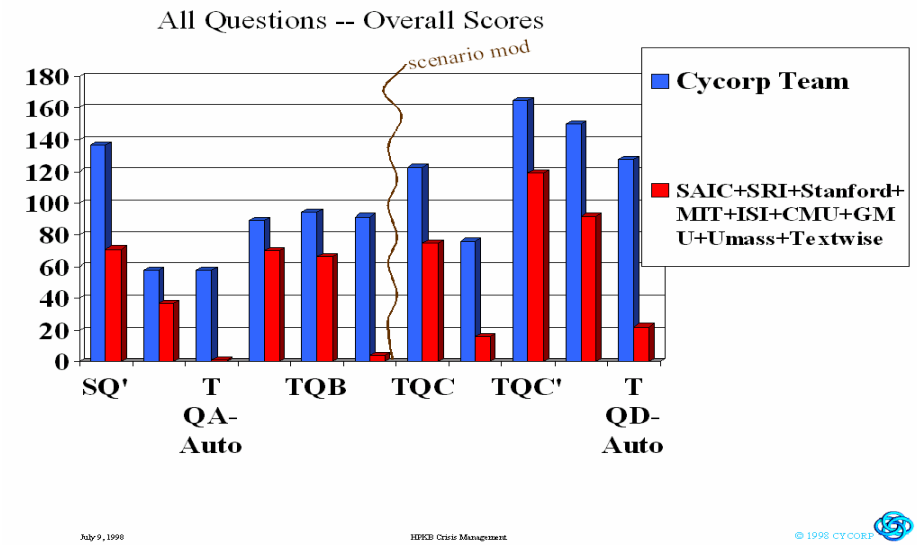


Figure 2

The full leveraging of Cyc was only utilized by the Teknowledge team, shown in blue on the graph that comprises Figure 2, above, prepared by the HPKB Evaluation team led by IET. The SAIC team, shown in red, used only a small portion of this and level (b), above. The results speak for themselves; each column represents a batch of 100 parameterized test questions that intelligence analysts posed (to each team's system) during the HPKB Evaluation exercise.

Why did Cyc turn out to be so needed for HPKB? When an application is developed, its very narrowness and specificity enables the developer to make all sorts of simplifying assumptions about time, space, the participants, etc. This "ontological corner-cutting" makes that application work sooner and more efficiently. Unfortunately, it makes that application rather brittle – it is quite difficult to change as the task changes. But more than this, it makes it virtually impossible to integrate its knowledge with a different application that is, conceptually, quite relevant. That is because an application constructed this way is developed under its own set of simplifying assumptions.

Even though two applications may use the same representation language, and the same "upper ontology" of very general concepts, they would not be able share their knowledge unless they shared those same simplifying assumptions. There would be just too many hidden assumptions underlying each of the two rule-sets. E.g., one might ignore time (such as MYCIN) and one might track entities over extended periods of time, and mixing their rules would lead to chaos and error. This phenomenon was even stronger when the size of the KB was increased to HPKB-magnitude levels, and Cyc thereby became not just a luxury but a necessity.

Innovation was required at each step of the process: to identify, articulate, codify, formalize, and organize that intermediate-level HPKB knowledge. We leveraged both

the Cyc content that had been accumulated and the ontological engineering methodology that had evolved in the process of acquiring and organizing that content. We provided HPKB modelers with painstakingly analyzed and axiomatized terms (including relations) relevant to battlefield models and crisis analysis, and dozens of assertions involving each of those terms. The terms were arranged in concept hierarchies, and — independently — organized into a structure of many hundreds of "micro-theories". The new classes and relations inherited all the relevant constraints and knowledge from Cyc's upper and middle levels.

To summarize what was unusual here: There was a focus on the intermediate levels of knowledge, not the upper ontology nor the domain-specific terms. There was also focus on the content — the axioms about the terms, not just the terms themselves, and we were able to leverage the existing content and methodology, fully declaratively, so that even the "escapes" to procedurally attached code were redundant with declarative axioms.

In getting our inference engine to work on a huge KB, we had to carefully examine the various tools and techniques developed over the past forty years of work in search, representation and inference; select the most powerful of these; get them to work together; and pioneer new tools and techniques as the need arose. Since we believed that there was power to be gained by further mining and assimilating special-case reasoning, we defined and integrated novel special-case reasoning modules. Often, new modules were conceived because it was noticed that some construct was frequently used in some task so a new module was built to intercept such situations in the future. This is a form of compilation of Task 1's intermediate-level knowledge.

The innovations we drew on, and which were extended are:

(1) having both an epistemological level (a clean, expressive language in which to represent the content) and a heuristic level (special purpose modules which recognize and handle commonly occurring situations);
(2) solving the problem of maintaining consistency in enormous KBs by dividing the assertions into clumps ("micro-theories" or "contexts") which share a set of common assumptions;
(3) using default reasoning based on argumentation, for example, gathering and weighing the pro- and con- arguments for each proposition;
(4) employing heuristics and assertions from the KB itself to guide even the innermost search loops in the inference engine;
(5) maintaining a supersaturated indexing structure which (along with deferral to the heuristic-level modules) enables truth maintenance to always be "on" even when reasoning over enormous KBs; and finally
(6) adhering to the principle of making syntax mirror semantics, even when it leads to redundant or counterintuitive representation/inference choices (e.g., some "ugly" redundancies and cachings often were highly cost-effective in the long run.)

The skewing of the results in our team's favor (Figure 2, above) is understandable given that the Cyc project has spent the past seventeen years developing appropriate data structures and algorithms to represent and efficiently handle common sense knowledge. The pre-existing Cyc KB had tens of thousands of useful rules for the HPKB Integrated Knowledge Base (IKB) to inherit, and the Cyc team had already analyzed the "perennial conceptual issues" for thirteen years prior to HPKB. Early adoption of Cyc's Public Upper Ontology, which was made public prior to the start of HPKB, as the "HPKB Jumpstart Ontology" gave both the Cycorp and SAIC teams a uniform, convenient, and reliable environment to add knowledge, ask questions and gather measurements, but this was most heavily utilized by the Cycorp/Teknowledge team.

Although there was inevitable overlap, the work reported here fell into the two general tasks outlined above and expanded below.

Task 1. Libraries of intermediate-level ontologies

Over the course of the project roughly 300,000 assertions were added to the Cyc KB. Many of these assertions captured source material content, including both facts about particular agents and events (e.g. terrorist groups) and generalizations about the goals and behavior of geopolitical agents. The 800 micro-theories were developed based on need, based on their overall relevance to HPKB target applications in general and to the Challenge Problems in particular. The theories worked on in the first year, for example, included:

- direct causality vs. contributed-to
- geography, and travel along pathways
- geopolitical entities or various sizes and types
- group actions, i.e. deliberate actions
- industry/economy and common goals of countries, terrorist groups, etc.
- civil infrastructure of various types
- mass media, and information bearing objects
- pathways in 2- and 3-space
- physics involved in travel, weapon use, etc.
- repeated events
- rules of engagement
- sensors
- transport
- utilities
- weather effects

For each of these theories we produced a 1-5 page English document summarizing the basic concepts that needed to be added to the existing ontology, the basic relationships and rules that needed to be formalized and added to the existing knowledge base, and a set of typical questions related to the HPKB Challenge Problems, HPKB target applications in general, etc., that should have (and in most cases did) depended on that set of rules and be answerable using them.

After examining the HPKB Challenge Problem statements, and associated questions, we made suggestions about gaps in coverage, in the level of the questions, and in supplied background materials. Further, we analyzed a couple of the questions in moderate detail to clarify the type and amount of formalization required to handle them. The validity of this work was confirmed with queries during the HPKB evaluation phase.

Driven by the Crisis Management Challenge Problem specification, and to a lesser extent the Battlespace Management Challenge Problem specification, we codified hundreds of concepts that were necessary for:

(a) representing the questions,
(b) representing the answers, and
(c) representing intermediate knowledge used in answering those questions.

We also codified thousands of additional axioms involving those new terms (and often, involving preexisting Cyc terms as well), and added them to the system. This then formed the basis for two sorts of demonstrations at the HPKB PI meeting December 3-5, 1998, in San Diego: one conducted by Cycorp, involving Crisis Management questions, and one that was demonstrated by our integration team, involving pathways in Battlespace Management.

Following the HPKB Ontology Jumpstart meeting we finished integrating the relatively small (~300) PANGLOSS/SENSUS top-level into our larger top-level ontology (~3000 concepts and 20,000 assertions about them). The net effect of this integration was the introduction of 32 new terms into Cyc's Upper Ontology, plus 291 "alignment" axioms that map terms in one ontology to terms in the other. There are three separate alignment relationships: equality, more-or-less-the-same, and a more complicated relationship in which the connection is explicitly referencable and describable.

Towards the end of the first year work began to add to and extend some of the other theories which had an effect on the kinds of problems we were trying to solve. These theories were those dealing with:

- weather effects
- risks and rewards
- international codes of conduct
- order of battle
- vehicles and transportation
- weapons
- weapons systems
- weapons delivery
- planning
- ratios
- fractions
- percentages
- part/whole reasoning

Driven by the Challenge Problems we added a total of about 40,000 assertions (of the eventual 175,000 which was our target for the end of FY00). Of these, about 75% were problem-specific facts incorporating some source material content; and of those 30,000, about half were entered automatically by a new knowledge assimilation tool called "the slurper". The other half was entered through manual effort. Divided up by Challenge Problem, about 75% of this work was for the Crisis Management Challenge Problems and 25% for the Battlespace Challenge Problems. At the end of May 1998, it became clear that we (Cycorp) would need to write a Planner in order for our team to handle the Battlespace Workaround Challenge Problems, and we finished in June 1998.

The results of the June 1998 Challenge Problem Evaluations showed that in the Battlespace Workaround Challenge Problems, we handled about half the 20 or so Workaround problems; this was far in excess of our June 1, 1998 goal, which was to handle at least one problem. In the Crisis Management Challenge Problems, Cyc produced the team's answers to all questions, and we consistently outperformed (and outscored) the entire combined SAIC team (SRI, two Stanford groups, ISI, two MIT groups, Northwestern, CMU, Textwise). More importantly, we answered about 80% of the newly-posed queries even after the modification to the scenario to introduce the use of biological weapons into it. This was far in excess of our original goal, which was to answer 60% of the pre-modification and 40% of the post-modification queries.

It should be reported, however, that this focus on scoring well at answering these questions had reached the point of diminishing returns. Further emphasis on that task reduced the overall quality, value and size of the Task 1 product at the end of the contract. The reason for this is simple: every hour spent entering some detailed information about, e.g., the bases of operation of Hezbellah, was an hour *not* spent entering intermediate knowledge, which was one of our main goals in this project.

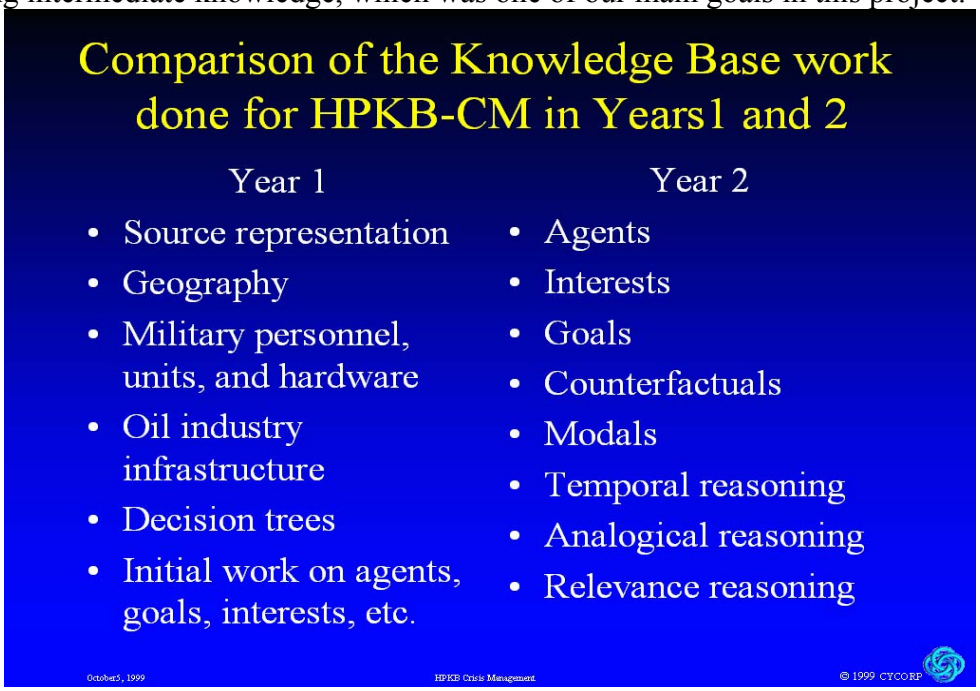


Figure 3

In Year 2, we significantly extended our Year 1 ontology work, (See Figure 3) as well as our work on knowledge-acquisition/formalization/entry/testing. The choice of theories to pursue was based in part on relevance to the evolving set of HPKB Year 2 Challenge Problems—i.e., the Course Of Action problems and ongoing address of the Crisis Management problems. Driven by the Challenge Problems, we added about 35,000 assertions to the Cyc KB during this period. In particular, we significantly expanded our representation of agents as interested rational actors, their related goals, motives and dispositions to act. Previously we had represented beliefs, goals, fears, desires, … of an agent, as well as other "propositional attitudes", like knowing and intending with limited representation for "internal parts", and limited ability to perform inference with those predicates.

Also in Year 2 we expanded our representation of agents' interests, their related goals, motives and dispositions to act, together with Cyc's ability to reason about the content of agents' goals, alternatives, etc. See Figure 4 for a sample of the types of Collective Goals we found necessary to codify.

We also expressed "interests" as first-class propositions, and there is now code-support for efficiently reasoning with them. We expect this work to be of general use in the future. We also expect to expand and revise it, building on design choices made and implemented during the past two years and especially in the last 9 months of the project.

Figure 4

Major knowledge base additions driven by Year 2 HPKB Crisis Management fell into four categories. The first of these concerned representation of agents' interests and relations between agents. Attention was given especially to characterization of interests by types of concerns, and to development of inference with interests, especially interests as reasons for acting. See Figure 5.

A second category of additions centered on the significance of agents' goals, including reasoning about the influence of goals on agents' actions, inferring an agent's goals from its agent type, and inferring likely actions based on known goals. A third category consisted of elaboration of modes of reasoning, specifically temporal reasoning, analogical reasoning and relevance reasoning. The fourth category concerned counterfactual modal representation, and the rules regarding them.

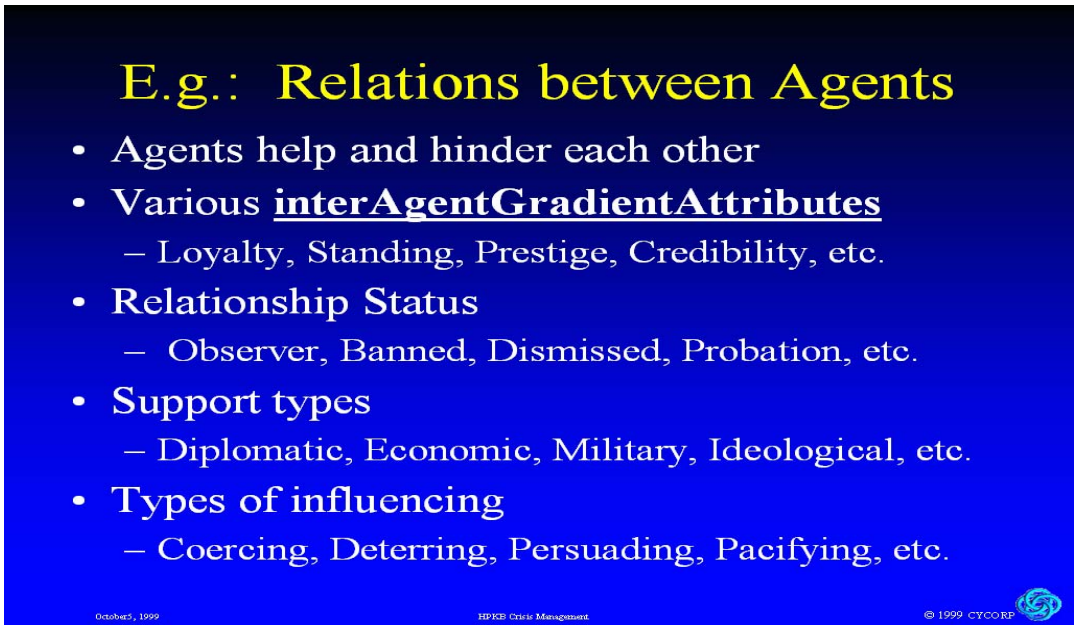


Figure 5

Work was done codifying things like the standard goals of a country. For instance, a national economy goal would be to ward off a recession. Logical goal representations were also expanded. These included such information as goal-affecting causal outcomes, the goal of maintaining circumstances the way they were before, wanting some desired state to become true, etc.

Task 2. Efficient reasoning on large KBs

For this task we began by producing and distributing initial versions of both the KB-maintenance API and the KB-reasoning API. This included posting them on the Web site, accessible both from cyc.com and from the Teknowledge-maintained central HPKB contractor web site. At the request of the HPKB management at DARPA as well as the other HPKB contractors, we divided these up slightly differently than planned, namely into a GFP-analogous (Generic Frame Package, an SRI product which in turn was based on an early (circa 1986) version of Cyc's representation language) external API specification and a KIF-analogous representation language and reasoning specification.

We also prepared and distributed executable code for both KB maintenance and inference, in Unix and Windows NT versions. When these were presented they represented a snapshot of existing technology rather than a newly developed one which

would appear later in the project. We also designed and began to implement a radical re-indexing scheme to more heavily cross-index the knowledge base assertions and, at the same time, to drastically reduce the space requirements (by a factor of slightly more than 2).

In related work, we built the first version of a Conceptual Graph (CG) → MELD translator, a MELD → CG translator, a KIF → MELD translator, and designed and built the first version of a GFP2.0 → CycAPI translator and a MELD → KIF translator (although that last was of necessity only partial, due to the limited expressive power of KIF.) We also edited, expanded, and released another version of the top-level Cyc ontology to serve as the basis for the IKB, and – as a result of the HPKB Ontology Jumpstart meeting – we integrated the small number (~ 300) of concepts from PANGLOSS/SENSUS top-level into our larger top-level ontology.

In some cases, the knowledge we added to Cyc should in principle have resulted in specific questions being answered correctly, and yet the inference engine failed to perform the necessary steps. This was occasionally due to a problem with incompleteness or a bug, but it was usually due to time constraints imposed by the user being exceeded by the system, so there was a "time out". We identified and added new heuristics and Heuristic Level (HL) modules, with associated special-purpose data structures and algorithms for maintaining and using them, to efficiently represent and reason about those unnecessarily slow cases. This led to acceptably fast responses.

We also committed to becoming an OKBC compliant site beginning with our end-of-February 1998 release of deliverables. However, since we did not receive one single OKBC request during the entire first year, we re-thought our effort to keep Cyc OKBC-compliant. Given the non-use of OKBC by the SAIC team itself, we decided to drop this effort and use the time on other more fruitful areas.

The early version of the MELD → KIF translator was improved upon, thanks to feedback from other HPKB participants, and it turned out to be adequate to generate KIF versions of all the Crisis Management Challenge Problem questions represented in MELD. We also improved translation work in the opposite translation direction.

Overall, the first year's work was largely driven by the two Challenge Problems. In particular (1) the need to efficiently answer – and correctly answer – the plethora of Crisis Management Challenge Problem questions, and (2) the need to efficiently plan – and correctly and completely plan – for the Workarounds problems. This in turn led to the creation of new Heuristic Level (HL) modules, the creation of an entirely new Planning module, and some experimental (and successful) changes to the fundamental heuristics used by the inference engine.

We produced a Java-based GUI front end, which was used by IET to formulate its queries, and which was used by both Integration teams, and their members, to pose sample test questions with which to exercise the developing systems. We also refined our MELD (CycL) → KIF translator. It performed very well, translating about 95% of

the Crisis Management Challenge Problem queries successfully into valid KIF, by the end of the first year.

We produced a template-query → MELD parser, which successfully translated about 90% of the Crisis Management Challenge Problem queries (from IET's grammar) into MELD. Some of these are really nontrivial translations, since some common English words can have many different meanings, and therefore each single template stood for potentially many different *types* of questions, not just many specific possible instantiations.

In Year 2 of this task, we made a number of improvements to the Cyc API and related tools. See Figures 6 and 7. The improved and better-integrated inference grapher provided a graphical display of the inference search which was very helpful to those using the system. We streamlined and made more robust the process of building smaller, specialized, client-centric KBs, such as the IKB. We added the ability to filter assertions displayed for a term by microtheory, and began ordering display of assertions in the browser by assertion time. This last feature was helped by improved precision of assertion timestamping, also developed under the HPKB rubric.

We identified a large set of internal methods for exposure as an external API. We created both a robust TCP server mechanism that allows for invocation of Cyc API services from external applications, and a Java toolkit designed to support Cyc API connectivity for Java applications. This improved upon our Java-based GUI front end, which was used by others to formulate its queries, and which was used by both integration teams (and their members) to pose sample test questions with which to exercise the developing systems.

In this connection, we also supplied a "reverse-parser" to generate legal parse-trees in the Challenge Problem grammar from query-instances. Additionally, we generated a large Cyc API document in HTML form, which described the protocol and functionality available in the Cyc API. These improvements provided enhanced integration with other HPKB participants.

We also enhanced aspects of our internal implementation to improve KB maintenance. A re-design of KB indexing reduced the system footprint by 75%. We implemented a new mechanism for identifying and creating KB subsets, which are self-consistent smaller knowledge bases generated from a larger knowledge base. This was used to provide regular HPKB versions of the Cyc knowledge base that were ultimately developed into the IKB. A Socratic (stepwise) ASK functionality was added to guide the asking and analysis of queries with very deep answers. Other additions to low-level system functionalities include English-from-CycL generation without the Lexicon, and new metadata for assertions. These last are exploited by the machine.

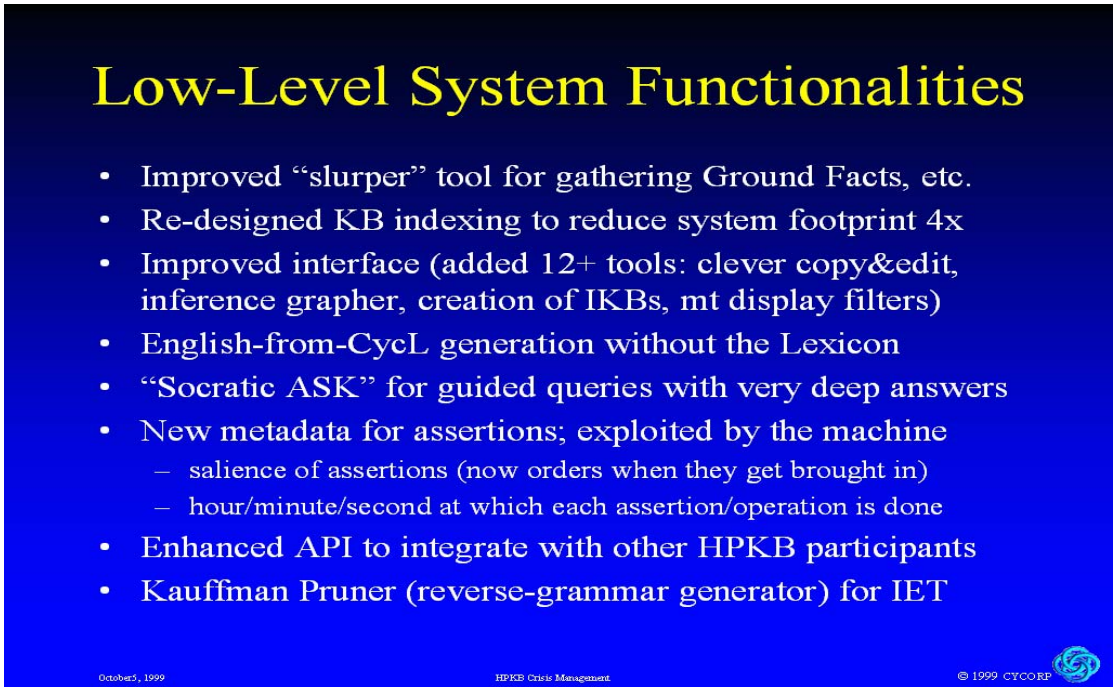


Figure 6

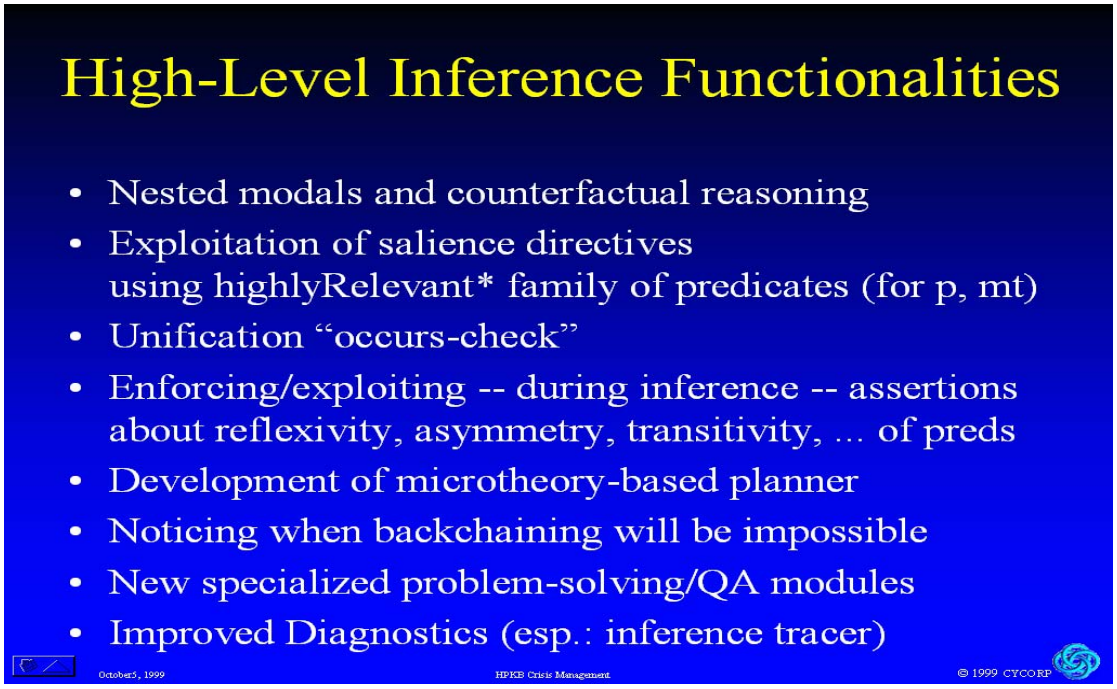


Figure 7

In the Course Of Action (COA) work, Cycorp and Teknowledge jointly created a facility for semi-automated uploading of COAs to the Cyc KB. This allowed for wholly automated translation of an iconic COA sketch into the KB, with additional supplementary knowledge added by human users working from the COA description. The sketch could then be assessed via a battery of critiquing queries.

We expanded the basic inferencing infrastructure in a number of ways, re-abstracting the Heuristic Level (HL) module declaration methodology and the portion of the inference harness that applies HL modules. Particularly, we improved support for closed world assumption reasoning, genlPreds (relational subsumption), transitivity and symmetry reasoning during inference, mathematical inferencing, and use of salience directives such as "except when" exception handling. We also increased canonicalizer robustness and implemented a distinction between natural kinds and sets, not just in the KB, but in the way Cyc inference treats them. These improvements allow scalability to hundreds and even thousands of HL modules. We then added hundreds of HL modules to provide efficient inference capabilities on common classes of subproblems of high utility. These improvements were driven largely by the Crisis Management (CM) and Course Of Action (COA) Challenge Problems; in particular, by (1) the need to both correctly and efficiently answer the plethora of Year 2 Crisis Management Challenge Problem questions, and (2) the need to analyze and critique a wide range of military COAs along multiple dimensions.

Performance evaluation: CM CP Challenge Questions and Issues

Performance on the Crisis Management evaluation was graded along a number of dimensions, including correctness of question formulations, quality of explanations, and, of course, correctness of answers. On all dimensions but one, Cyc outperformed the other team (and on the remaining dimension, the scores were identical). For the initial round of questions in the evaluations, Cyc's question/answer/explanation results averaged 2.5 on a scale of 4; after a short repair period, Cyc's answers averaged around 3.6 for a batch of variant questions. Both average scores were higher than the other team's.

One of Cyc's strengths was in eliminating the absurd when faced with Challenge Questions. For example the question "Can Iran successfully do a naval invasion of Afghanistan?" Cyc not only answered correctly, but we were able to develop shortcuts for inferring such impossibilities quickly. When faced with the question "How are typical interests of an ambassador of the US like those of a UN envoy?", Cyc created a hypothetical query and therefore created a new context. It gave this new context epistemological status of Hypothetical, postulated a US ambassador and a UN envoy, forward chained, and then asked itself for similarly-aligned interests which they held in common, which led to some backchaining. It was then able to generate many answers, almost all of which were derived from their common interests.

Overall, Cyc's performance was much more sophisticated than we had originally expected, with higher scores and a higher percentage of questions answered. Several novel solutions/answers/paths were obtained, which gave us ideas about making results comprehensible and accessible to future users.

Conclusions

We added slightly more assertions and terms to the KB than we had originally proposed, with a total addition of approximately 300,000 assertions and 50,000 concepts. The

additions were more specialized than we had expected (some topics were skipped), probably due to the Challenge Problems driving both years. The process was also more automated than we expected it to be (slurping, sparse-term and dialog tools). Importantly, the terms and assertions entered in Year 1 proved of considerable value in the Year 2 evaluations, showing that these assertions allow for significant re-use. Cyc's common-sense knowledge was also used extensively. In fact, approximately 80% of the *axioms* used to answer Crisis Management Challenge Problem questions were entered in the KB before Year 2, and approximately 95% of the *terms* used in the answers and the rules, etc., used to derive them were pre-Year 2. This work has been relevant for Command Post of the Future (CPOF), Rapid Knowledge Formation (RKF) and Evidence Extraction and Link Discovery (EELD), and we expect it to be of continuing general use in the future. We expect to expand and revise it, building on design choices made and implemented over the two years of the project, and especially the final nine months. See Figs. 8- 10 below.

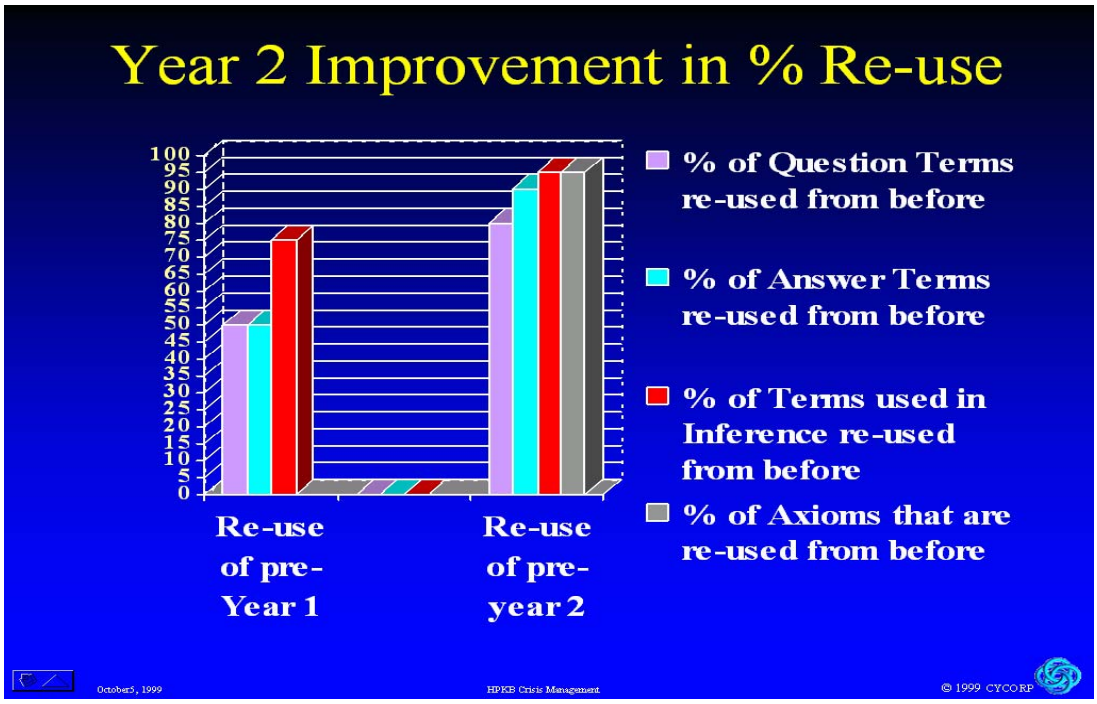


Figure 8

The process of improving knowledge base maintenance and processing capabilities for HPKB was one of many small enhancements rather than one of major breakthroughs, a circumstance demonstrating the fundamental appropriateness of the Cyc knowledge base in such a demanding context. The aggregate improvement resulting from these incremental advances was surprisingly good, a conclusion that holds true for inferencing functionality, interfaces and system utilities.

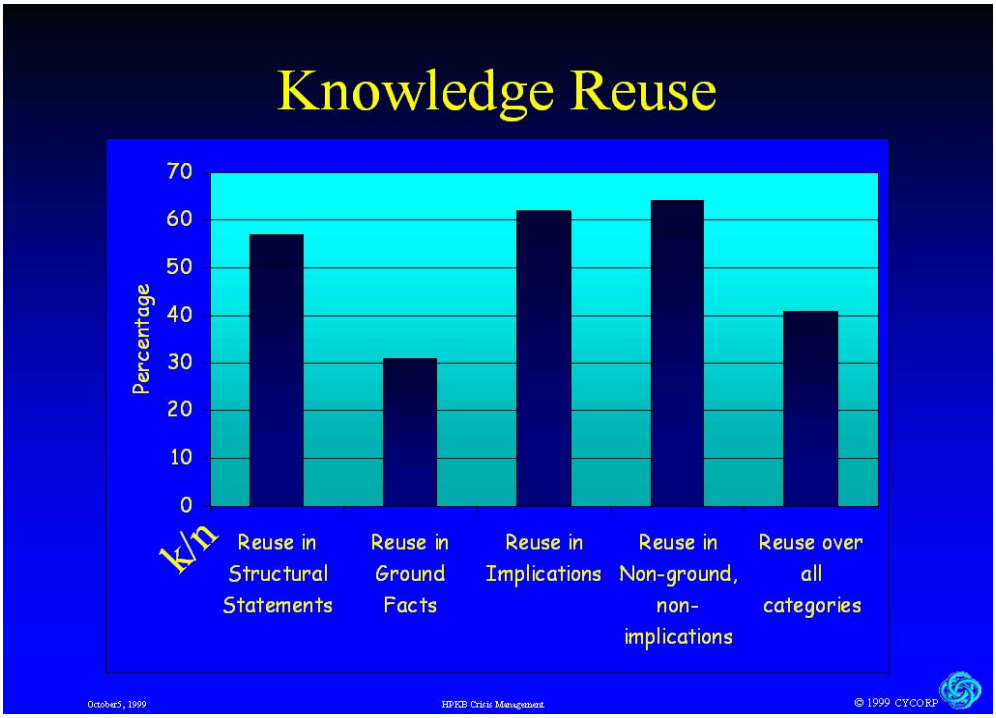


Figure 9

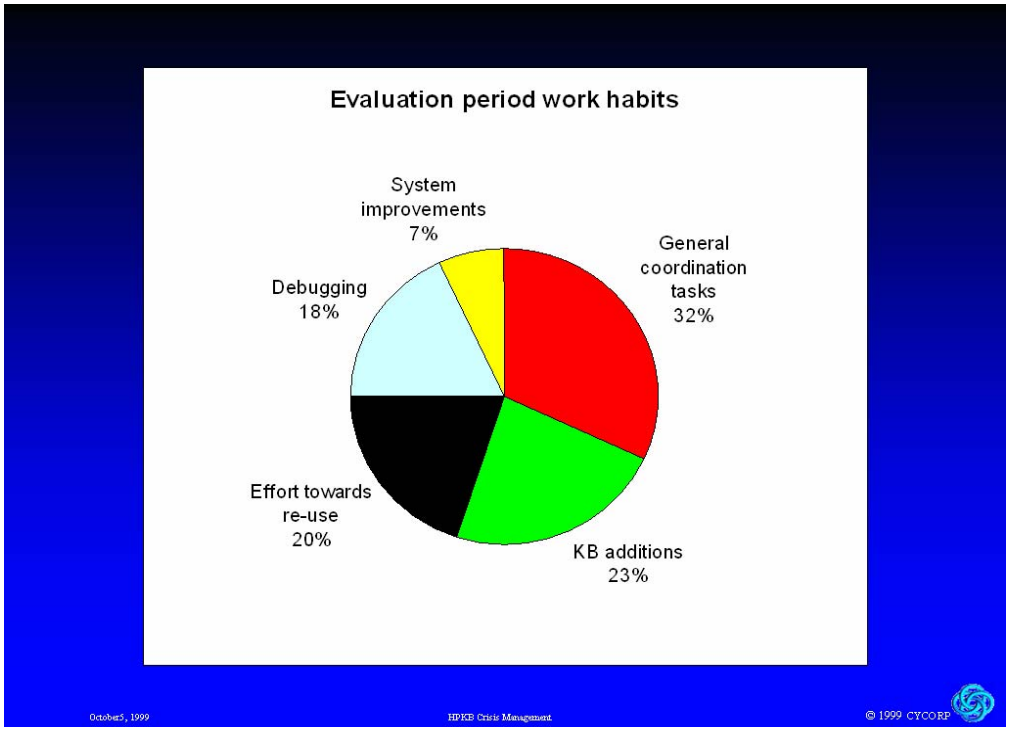


Figure 10

Appendix 1: Copy of AI Magazine Publication: "Practical Knowledge Representation and the DARPA High Performance Knowledge Bases Project"

Adam Pease
Teknowledge
1810 Embarcadero
Palo Alto, CA 94303
USA
apease@teknowledge.com

Vinay Chaudhri
SRI International
333 Ravenswood Ave
Menlo Park, CA 94025 USA
chaudhri@ai.sri.com

Fritz Lehmann
Cycorp
3721 Executive Cntr Dr
Austin, TX 78731
USA
fritz@cyc.com

Adam Farquhar *
Schlumberger
8311 North FM 620 Road,
Austin TX 78726
USA
afarquhar@slb.com

Abstract

We address the experiences of the DARPA High Performance Knowledge Bases (HPKB) (Cohen et al., 1998) project in practical knowledge representation. The purpose of the HPKB project was to develop new techniques for rapid development of knowledge bases. The goal of this paper is to describe several technical issues that arose in creation of practical KB content.

EXPERIMENTS

The project had two main objectives: first, to advance the science of Artificial Intelligence Knowledge Representation and Knowledge Base content creation, and second, to apply these technologies to create applications with utility to the Department of Defense. The applications were specified as two Challenge Problems (CPs). The first was the Crisis Management CP, an effort to develop an automated question answering system that met the needs of analysts who must be informed about emerging world crises. The second was ~~the B~~battlespace Challenge Problem.

* The author performed this work while a member of the Knowledge Systems Laboratory, Stanford University

This effort covered two knowledge-based systems. One reasoned about battlefield engineering tasks such as workaround computation; the other critiqued battle plans. This paper addresses issues primarily from the experiences of the Crisis Management CP.

PROJECT ORGANIZATION

Two teams worked on these challenge problems. In the Crisis Management CP, one team used Cyc (Lenat, 1995) and its MELD (Cycorp, 1997) representation language. Another used KIF (Genesereth & Fikes, 1992) and the SNARK (Stickel et al., 1994) and ATP theorem provers.

HPKB was a very large project and many aspects are not even mentioned in

this paper. The interested reader should refer to the HPKB web site (HPKB Web, 1999) and publications list (HPKB Pubs, 1999).

TRADEOFFS IN THEORY CREATION

There is a cost in creating reusable representations. It is more costly to create representations that will be reusable across multiple domains than it is to create a representation that is suitable for just one application.

We believe there is a need for a more formal development process that is built on some of the best practices from the software engineering community. It is always easier to create specific and limited content as opposed to crafting a general domain theory. The challenge is to build time into the development process for planning and systems analysis, design, implementation, testing, and *rework and generalization*. Much like the spiral development model advocated by Booch (Booch, 1994) and others, a good development process iterates through these stages several times during a development process. One possible instantiation of this process would be as follows:

DEVELOPMENT PROCESS

Planning and systems analysis. It is essential to determine the need that the knowledge must fulfill. Will it be used for inference? To define a semantics for natural language interpretation? As an interlingua for cooperating agents or software modules? Each of these applications will entail a different emphasis on the richness of the formalization.

Also considered should be the performance requirements of the implementation. How fast should the resulting inference be? Will the knowledge base need to be augmented with a significant amount of instance data? Is logical completeness a necessity? Answering these questions will help to determine how expressive the knowledge representation can be, which will in turn partially determine the inference engine that needs to be employed.

We should note that in the HPKB project, a great deal of the systems analysis phase was done for the knowledge base developers by providing them with a Challenge Problem (Schrag, 1999:2) that specified and detailed the scope and purpose of the experiments that were to come. A great deal of informally specified knowledge was also provided.

Design. One way to design a knowledge base is initially to specify it informally. The engineer creates English examples illustrating sample reasoning chains. Glossaries with English definitions are created. It can also be useful to create a taxonomy as a skeleton on which the theory can be developed.

Implementation. As in software development, if the two previous phases are done properly, the implementation phase can proceed quickly. It is important that all members of the development team participate in the first two phases. Also helpful is a formal review process led by a chief knowledge *architect*.

Knowledge architects, software architects, and building architects all have similar roles. While they do not control every detail of a project, they set the overall design, standards, and aesthetics. A knowledge architect provides guidance to his team about how to meet project requirements, find a balance in tradeoffs between development speed and implementation

generality, maintain consistent approaches across diverse team members, and set standards for reviews and documentation. A good architect manages by objectives and standards, which result in an implementation that speaks with one voice while allowing participants the freedom to innovate.

Testing. While this phase is obvious for any knowledge base that is to be used in a computational system, performing systematic testing is often ignored. If the knowledge base has been developed in a modular manner, an equivalent to *unit testing* can be performed on each small theory. Unit testing allows for testing of greater coverage than final *integration testing*.

Rework and Generalization. This phase is the most often ignored simply because of the dynamics of most research projects. Once the practical objectives of the sponsors have been achieved, little time or money remains in the project to correct shortcuts that may have been made. However, this phase is possibly the most important if incremental scientific results are to be achieved.

Any large scale project will necessarily go through the above phases several times. A good knowledge engineering process has many similarities to a good software engineering process.

THEORY REUSE

Both teams reused the HPKB upper level (HPKB-UL) ontology, derived from Cyc, during the project. The representation for the temporal knowledge available in the HPKB upper ontology was very well designed. From the HPKB-UL, we also used representation for communicative actions, slots on actions (agent roles), and the primitives for representing paths.

For one team, reusing these theories required translating the representation, extracting portions of the input ontology for use, and doing limited reformulation. There was also the need to further extend the library of the representation primitives for causality, scales, actions, processes, and qualitative influences.

The Cyc-based team had access to the entire Cyc knowledge base. In addition to areas mentioned for the upper level, there are good theories for concrete physical domains of all sorts. Theories of belief, goals, trust, and the expression of causality in nondeterministic human events are essential and less well developed.

HPKB had a good record of reusing terms and basic statements about terms. Developers gained a great deal of value from inheriting a large set of precise distinctions about things in the world, such as the differences among a goal, a plan, and a desire. However, comparatively little reuse of general rules was evident. This can be explained in several ways:

It's hard to write truly general rules.

Insufficient effort has been placed into writing general rules because of the pressures of day-to-day results.

Practicalities of inference are such that a long chain of reasoning involving general rules doesn't work in a reasonable amount of time. One has to "short-circuit" the deep reasoning with special-purpose rules that make the inference tractable.

As an example of reuse, consider the following inference task performed by our system:

What risks can Iran expect in sponsoring a terrorist attack in Saudi Arabia?

To answer questions of this type, one team developed a simple cause-effect model. All the predicates below,

including *cause-event-event*, *beneficiary*, and *maleficiary* were reused from the HPKB-UL. Even though we capture only direct effects of an action, this simple model was effective in practice. This example illustrates the reuse of notions of causality that were already conceptualized in the HPKB-UL. The following is an example application of these representation primitives.

```
(forall ((?terrorist-attack
           terrorist-attack)
         (?agent agent))
(=>
  (performed-by ?terrorist-attack
?agent)
  (exists
     ((?punishment punishment))
     (and
        (causes-event-event
            ?terrorist-attack
            ?punishment)
       (maleficiary ?punishment
?agent)
        (object-acted-on
            ?punishment ?agent)))))

(forall ((?action action)
         (?action1 action1))
 (implies
   (and
      (causes-event-event ?action
?action1)
      (performed-by ?action ?agent)
      (beneficiary ?action1 ?agent))
   (benefit-of-action
      ?action ?action1 ?agent)))
```

A detailed description of technical problems encountered in reuse is available in (Cohen et al., 1999) (Chaudhri et al., 2000). Even though we reused representations for actions and casuality from HPKB-UL, significant additional representation work needed to be done. This suggests that a representation library for actions, causality, and qualitative influences needs to be extended. The theoretical KR community is invited to study the HPKB-UL and propose representational modules to be included in it.

PRACTICAL REPRESENTATIONAL ISSUES

There was a lack of principles for designing taxonomies. As a result, creating and maintaining a taxonomy of primitive concepts became increasingly difficult as its size grew. Conventional description logic techniques do not help in creating taxonomies that contain a large number of primitive concepts. Better principles for taxonomy design are needed.

There was also the need to "hand-compile" deep reasoning out into special-purpose theories that had tractable inference chains.

TAXONOMY

Like many other KBs, the class-subclass taxonomy was an overarching organizing principle in our HPKB KB. A *class-subclass* taxonomy serves as an indexing aid to find knowledge and add new knowledge, and to serve as a method to efficiently write axioms by using inheritance.

While designing the taxonomies for the HPKB project, we encountered the following problems:

1. As the taxonomy got bigger, it became increasingly difficult to add new concepts to it. As a result, there were concepts that had incorrect positions in the taxonomy:

Some concepts had missing links. A class has a missing super-class link if it is a subclass of another class B, but the subclass relationship is not declared.

Some concepts had wrong links. A class has a wrong link in a taxonomy if it is a direct subclass of B, but the subclass relationship does not hold true.

2. We were encountering concepts that were being created by a cross-product of two sets of concepts, for example:

```
{International, transnational,
subnational, national} x
{organization, agent}
{Support, oppose} x {attack,
terrorist-attack, chemical-attack}
{Humanitarian, political, military,
diplomatic} x {Organization, Action}
```

Some concepts had a very large number of subclasses. In some cases, this was due to orthogonal ways to categorize a concept. As a result, such categorizations were not mutually disjoint. Large fan-outs made it cumbersome to navigate through the taxonomy. As an example, consider the following snippet from the taxonomy representing organizations.

Figure 1. A portion of a taxonomy representing organizations showing orthogonal categorizations

While the categorization of commercial organization and unincorporated organization is based on the legal status of an organization, the categorization of international organization and subnational organization is based on extent of operations. Mixing such orthogonal categorizations adds to the complexity of the taxonomy.

4. If two classes are disjoint, the disjointness relationship must be declared.

5. There should be no redundant classes representing identical concepts.

A taxonomy is well designed if it is free from all the problems mentioned above. Ensuring these properties in a small taxonomy is easy even if it is done manually. However, as the taxonomy size grows, making taxonomy well structured manually is very time consuming. These problems are indicative of a poor design methodology for developing taxonomies. We argue below that these problems go away if one takes a more principled approach to developing these taxonomies and supports additional constructs to structure the taxonomies.

If every concept has necessary and sufficient definitions, one can use a classifier to help alleviate Problem 1. In practice, we found that too many concepts were primitive and did not have necessary and sufficient definitions. Therefore, we cannot use a classifier. Problem 1 stems from the fact that the taxonomy itself is getting too complex. For example, a concept is linked or needs to be linked to too many different places. As a result, defining a new primitive concept involves manually encoding its relationship to numerous other primitive concepts -- a process that is error prone. One would hope that the process of organizing such concepts into a taxonomy would be considerably simpler than doing the same thing for the original concepts.

We need principles for taxonomy design that can enable us to economically create and maintain large taxonomies of primitive concepts.

COMPOSABLE REPRESENTATIONS

We believe that representations are more *reusable* if they are *compositionally* constructed. A representation is compositional if it represents each individual concept in the domain of discourse, and the representation of complex concepts is obtained by composing representations of individual concepts. To illustrate this, consider the representation of the following:

```
The USA conducts a peacekeeping
mission.
```

In this example, we can use several different representations. One degenerate representation might be

```
UsConductOfPeacekeeping
```

This representation compiles all the semantic features of the English statement into a symbol. A more reasonable representation might be

```
(and
  (instance-of ?Y
PeacekeepingOperation)
  (performedBy ?X ?Y)
  (members ?X USMilitaryOrganization))
```

in which the action has been expanded to describe an action type and detail about the performer of the action. We can further decompose the action by describing it as an event that has the purpose of maintaining a particular state.

```
(and
  (toMaintain ?Y PeaceAccord)
  (instance-of ?Y MilitaryOperation)
  (performedBy ?X ?Y)
  (members ?X USMilitaryOrganization))
```

(Schrag, 1999:1) has proposed the following compositionality hypothesis: noncompositional representations are inexpensive to build but they are brittle with respect to weak problem generalizations and must be re-engineered (for example, into compositional representations) or replaced.

According to the compositionality hypothesis, the first representation is inferior to the later versions. However, although many knowledge engineers would have a strong intuition that the later representations are superior, there is no strong empirical basis for the proposed criticism of the first representation. One approach that would admit the first representation as acceptable would be to add additional terms to the KB and give a more complete definition to it. Thus, even if the first representation is noncompositional, it is amenable to generalization if an application requires it.

The relative comparison between the two representations is unlikely to have a context-independent answer. If in the current application we never need to represent or reason with *conduct*, *mission*, or *peacekeeping*, other than talking about "conduct peacekeeping mission", the less expressive representation is adequate. One can certainly argue that the first representation is less reusable. However, that depends on the next application. If we use the first representation, and the next application requires us to represent or reason with *conduct, mission,* or *peacekeeping*, it is possible to add them to the KB and use them to define `UsConductOfPeacekeeping`. This may be studied more formally with an analytical model as follows.

Suppose we design two representations, one of which uses *n1* terms and the other uses *n2* terms. Suppose cost/term is *c* and is constant in both cases. The cost for building a KB for the two cases is *c*n1*and *c*n2,* respectively.

If speeding up KB construction time for just one application is the objective, a compositional representation can be bad! However, if we also care about reuse, that may not be necessarily so. Does compositionality enable reuse? We cannot find out until we run replicated trials.

Suppose we reuse the KB for a new application. This new application requires the same knowledge fragment that we have already coded but requires a different compositionality, and we end up defining *n3* new terms for the first representation and *n4* new terms for the second representation. It is possible that either of *n3* or *n4* is zero. The cost for the new application is *c*n3* and *c*n4,* respectively.

The objective should be to minimize *c*(n1+n3)* or *c*(n2+n4)*. The model can be generalized to N applications.

The parameter *c* can be viewed as time to construct a KB, and thus linked directly to the program goal of speeding up the KB construction time. Further, this model allows us to do the following:
Measure whether it is really worth decomposing a representation
Amortize the higher cost of decomposition over a number of applications
Make explicit the relationship between reuse and compositionality
Exploring this tradeoff is open for future work.

"COMPILED" REPRESENTATIONS

One of the HPKB Challenge Problems dealt with reasoning about economic actions. One might encode the following chain:

```
There exist economic actions
which open markets -
    opening markets encourages
    exports -
        increasing exports improves
        a country's trade balance -
            positive trade balance
            improves economic health -
                all countries are
                interested in
                economic health
```

However, it may be that in practice, because of the complexity and compositionality of each of the encoded statements, and the depth of the inference, such a reasoning chain does not terminate in a reasonable amount of time. While an inference of depth five may not seem very taxing, consider the fact that this set of rules exists in a very large KB along with tens of thousands of others. The task of matching these particular rules and determining that huge numbers of others are irrelevant is time consuming.
The result is that to create a reasoning system that reaches a conclusion in a short amount of time, one might have to encode

```
There is a set of actions
which open markets -
    opening markets contributes
    to economic health -
        all countries are interested
        in economic health
```

along with defining a set of actions as subclasses of "opening markets" actions.
The goals of a project can strongly bias a knowledge engineer to the second representation. If a research team is scored, or a development team is paid on the basis of "correct" answers, compositionality and deep reasoning will be sacrificed.

METRICS

For any practical KB content creation work, there is a need to state crisply the competence level of a KB, and to make claims about increasing competence as the time goes along. Even though we know that there is an intuitive relationship between the size of a KB and its competence, there is no foolproof way functionally to relate the size to competence. As an approximate measure, we used the axiom count in a KB as one measure of competence.
An early challenge during the project was to define what counts as an axiom. Given that there is no universal way to count axioms, and that the axiom counts are sensitive to the modeling style and the language, we developed the following scheme for categorization of axioms in a KB.
Constants are any names in the KB, whether an individual, class, relation, function, or a KB module
Structural statements are ground statements using any of (Cyc term/Ontolingua term) #$isa/instance-of, #$genls/subclass-of, #$genlPreds/subrelation-of, #$disjointWith/disjoint, #$partitionedInto/disjoint-decomposition, #$thePartition/partition, #$genlMt, #$argXIsa/nth-domain (where X is a digit), #$argXgenls/nth-domain-

subclass-of (where X is a digit), #$arity/function-arity/relation-arity, #$resultIsa/range, #$resultGenls/range-subclass-of

Ground facts are any statement without a variable.

Implications include any non-ground statement that has an `#$implies` (note that a ground statement that contains an `#$implies` is counted as a ground statement)

Non-ground, non-implications are statements that contain variables but not an implication.

This categorization is imperfect, but it is easy to implement and was applicable to both of the crisis management systems developed during the HPKB project.

The structural statements have an intuitive status in most systems: for SNARK the structural information is sort information, for Cyc the structural information is called definitional, and for description logic systems the structural relations are usually called *concept constructors*. The statements with implications are rules. Ground facts often represent knowledge that can be found in an almanac or database.

A weakness of this categorization is that it counts many statements as ground statements even though they are not actually ground. For example, the statements involving `template-slot-value`, and `#$relationAllExists` are counted as ground. Further refinement to this categorization is left open for future work.

The axiom categorization scheme gave us an empirical tool to compare content across the two systems developed in the project. We would welcome proposals from the theoretical KR community, detailing more systematic ways to measure the competence of a large KB.

STANDARDS

Having a standard syntax is a necessity, but standard syntax plays a relatively small role in addressing the practical challenges facing the knowledge engineer. There is a need to move from an emphasis on standards of syntax, or on defining a precise semantics for tiny theories, to standard large theories and style guides for axiom writing.

For example, the subclass relationship can be either stated as

1. *(subclass-of A B),* or as
2. *(=> (A ?x) (B ?x))*

Both of these forms are ANSI KIF. The first form uses *subclass-of* as a relation to compactly encode information that could also be written as in Form 2. The first form also has the advantage that a reasoner supporting taxonomic inference can take advantage of this form, which can be quite difficult for the second form.

As another example, consider three commonly used ways to specify the type information of variables in an axiom: (1) using ANSI KIF-style typed quantifiers, (2) using *instance-of* relations, or (3) using the class as a relation. Here is an example axiom encoded in these three forms:

```
(forall ((?x action)
         (?y action)
         (?z country))
    (=>
       (and
          (may-cause ?x ?y)
          (performed-by ?x ?z)
          (maleficiary ?y ?z))
          (risk-of-action
              ?x ?z ?y)))

(forall (?x ?y ?z)
    (=>
       (and
          (instance-of ?x action)
          (instance-of ?y action)
          (instance-of ?z country)
          (may-cause ?x ?y)
          (performed-by ?x ?z)
          (maleficiary ?y ?z))
       (risk-of-action ?x ?z ?y)))
```

```
(forall (?x ?y ?z)
    (=>
       (and
          (action ?x)
          (action ?y)
          (country ?z)
          (may-cause ?x ?y)
          (performed-by ?x ?z)
          (maleficiary ?y ?z))
       (risk-of-action ?x ?z ?y))
```

One additional factor might be that may-cause and risk-of-action could be defined as referring to types of actions rather than instances in different knowledge bases.

These three forms are equivalent and follow the ANSI KIF standard. In spite of the standard, people come up with sufficiently different ways to write axioms to make the knowledge exchange difficult. Therefore, the standards must be accompanied by a style guide before they can enable knowledge exchange. In the above example, the style guide could require that the type information for axioms should always be stated in the quantifier specification.

USING A VERY EXPRESSIVE REPRESENTATION

Expressive representations enable a degree of generality and reuse not possible with more restricted representations. Because of interactions among axioms, the inference time can become very high. The most general and reusable theory is not useful if inference on those theories is not tractable for your inference engine. Some ways of addressing this problem are by partitioning the KB into modules to isolate the interactions among axioms, and by compiling knowledge by hand into more efficient representations.

One team had the goal of keeping the inference time for answering a question to less than 2 minutes. If all the axioms were loaded at the same search space, it was not possible to meet this requirement. Therefore, we *modularized* the KB to limit the interactions among axioms and achieve the desired response time. This problem would have been less critical had we limited the representation to horn clauses.

KB modularization means dividing the content of a KB into conceptual partitions that serve the basis for KB development and inference. We experimented with two ways to modularize a KB: subject based and task based. A *subject-based modularization* organizes a KB by subject area and can enable easier sharing and development of KB content. A subject area can be assigned to a knowledge engineer to direct its development. While reusing a KB, one can select a KB in the subject area of interest. A *task-based modularization* organizes a KB by the rules and individuals that are relevant to a task, thus significantly reducing the search space. The class, function, and relation definitions do not affect the search space, and therefore need not be modularized to speed up inference.

Modularization of a KB based on the subject-based criteria and the task-based criteria can be different and can coexist. We used both subject-based and task-based modularization during the project. For example, three major subject areas covered in our KB are *actions, agents,* and *interests*. We also created task-specific partitions in the KB based on specific parameterized questions (PQs). For example, for answering questions about interaction between interests and actions, there was no need for knowledge about specific terrorist groups in the KB that were kept in a separate partition. The approach to modularization described here was clearly engineering driven, and better principles to arrive at the modularization

are needed. Techniques to develop modules for a KB in a way that isolates independent reasoning chains are clearly of special importance.

ISSUES IMPEDING PROGRESS

Inference engine performance is one crucial technical issue. While it is not easy to develop inference modules for very expressive features, it is incredibly hard to get those modules to perform well.

Despite the program's name, execution speed was not an issue under investigation in HPKB. Many researchers have studied algorithms, speed, and complexity. HPKB was extremely important because it focused on content. Much research on inference performance has not been undertaken in the context of practical reasoning on large knowledge bases. The challenge now is to focus on merging research on creating and reasoning with large knowledge bases with research on inference performance.

The most important non-technical issue is research parochialism. The need to "own" a language, ontology, theory, or protocol is very powerful, whether in terms of building a research identity or a commercial base. However, this fragmentation is hampering progress.

Allen's seminal work (Allen, 1984) (Allen, 1994) on representing temporal knowledge is a good example of the kind of results that we need, and it is also well referenced and adopted in the applied AI community. Allen's work identified the primitives necessary to represent a sufficiently large class of temporal information and proposed inference procedures. If we could do the same for other domains such as actions, space, and causality, etc, it would greatly speed the practical KB construction. It is also the case that careful theoretical work has been done in these areas but may not be well known or adopted in the applied AI community. This work includes (Cohn et al., 1997), (Giunchiglia & Lifschitz, 1998), (Giunchiglia & Lifschitz, 1999), (Lifschitz, 1987), (McCain & Turner, 1997).

The KR community is still theoretically focused. Few people are interested in working on creating KB content. The time is right for a new focus on practical KB content creation.

Acknowledgments

We wish to acknowledge our DARPA sponsor, Murray Burke, for funding and guiding this work. We also wish to acknowledge the essential contribution of Robert Schrag at IET, who specified the Challenge Problem that made this research possible. Cleo Condoravdi provided a very helpful review of the paper.

References

Allen, J. (1984). "Towards a General Theory of Action and Time", Artificial Intelligence 23, pp 123-154.

Allen, James and George Ferguson (1994). "Actions and Events in Interval Temporal Logic", Journal of Logic and Computation 4, 531-579.

Booch, G. (1994). Object-Oriented Analysis and Design With Applications, Addison-Wesley

Chaudhri, V., J. Lowrance, J. Thomere, M. Stickel, and R. Waldinger (2000). Ontology Construction Toolkit. Artificial Intelligence Center, Technical Report.

Cohen, P, V. Chaudhri, A. Pease, and R. Schrag (1999). "Does Prior Knowledge Facilitate the Development of Knowledge Based Systems", Proceedings of AAAI-99.

Cohen, P., R. Schrag, Jones, A. Pease, Lin, Starr, Gunning, and Burke (1998). "The DARPA High Performance Knowledge Bases Project", AI Magazine, Vol. 19 No.4, Winter.

Cohn, A., B. Bennet, J. Gooday, and N. Gotts (1997). Representation and Reasoning with Qualitative Spatial Relations about Regions. http://www.scs.leeds.ac.uk/spacenet/leedsqsr.html

Cycorp (1998). "Features of the CycL Language", on-line report at http://www.cyc.com/cycl.html .

Genesereth, M., and R. Fikes (Editors) (1992). Knowledge Interchange Format, Version 3.0 Reference Manual, Computer Science Department, Stanford University, Technical Report Logic-92-1, June.

Giunchiglia, E., and V. Lifschitz (1998). An action language based on causal explanation: preliminary report. In Proceedings AAAI-98, pp. 623-630.

Giunchiglia, E., and V. Lifschitz (1999). "Action Languages, Temporal Action Logics and the Situation Calculus". In Working Notes of the IJCAI-99 Workshop on Nonmonotonic Reasoning, Action, and Change.

HPKB Web (1999). "HPKB Web Site", http://projects.teknowledge.com/HPKB/

HPKB Pubs (1999). "HPKB Publications Page", http://projects.teknowledge.com/HPKB/Publications.html

Lenat, D., 1995, "Cyc: A Large-Scale Investment in Knowledge Infrastructure". Communications of the ACM 38, no. 11, November.

Lifschitz, V. (1987). "Formal Theories of Action". The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop. Los Altos, CA: Morgan Kaufmann Publishers.

McCain and Turner (1997), "Causal Theories of Action and Change", Proceedings of AAAI-97, pp 460-465.

Schrag, R. (1999:1), email communication.

Schrag, R. (1999:2). "HPKB Year 2 Crisis Management, End-to-end Challenge Problem Specification", Version 1.2, February 5, Information Extraction and Transport, Inc. and Pacific-Sierra Research Corp. Rosslyn, VA. http://www.iet.com/Projects/HPKB/Y2/Y2-CM-CP.doc

Stickel, M., R. Waldinger, M. Lowry, T. Pressburger, and I. Underwood (1994). "Deductive Composition of Astronomical Software from Subroutine Libraries", in Proceedings of the Twelfth

International Conference on Automated Deduction (CADE-12), June, 341-355