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.

1. HPKB PROJECT

1.1. 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 Battlespace Challenge Problem. 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.

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

1.2. 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).

2. 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:

2.1. 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.

3. 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.

4. 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.

4.1. 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}
```

3. 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.

4.2. 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:

a) Measure whether it is really worth decomposing a representation

b) Amortize the higher cost of decomposition over a number of applications

c) Make explicit the relationship between reuse and compositionality

Exploring this tradeoff is open for future work.

4.3. "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.

4.4. 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.

5. 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.

6. 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 ot 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.

7. 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 nontechnical 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.