Report • January 2000
DATA Item Number: A001

Ontology Construction Tool Kit

SRI Project No. 1633
Contract No. N66001-97-C-8550
ARPA Order Number: F114

Covering the period: June 1997 to December 1999

Prepared by:

Dr. Vinay K. Chaudhri, Senior Computer Scientist
Dr. John D. Lowrance, Program Director
Dr. Mark E. Stickel, Principal Scientist
Jerome Thomere, Research Engineer
Dr. Richard J. Waldinger, Principal Scientist
Artificial Intelligence Center
Computing and Engineering Sciences Division

Prepared for:

Naval Command, Control & Ocean Surveillance Center
53560 Hull Street
San Diego, CA 92138-0818

DARPA/ISO
3701 North Fairfax Drive
Arlington, VA 22203-1714

ABSTRACT

The goal of this project was to enable knowledge engineers to construct knowledge bases (KBs) faster. To achieve this goal, we investigated two techniques: knowledge reuse and axiom templates. The results were demonstrated by developing a question-answering system for the crisis management challenge problem (CMCP). The solution of the CMCP required addressing problems broader than just knowledge reuse and axiom templates. The technical issues addressed in the process can be broadly classified into two categories: knowledge base content development and knowledge server development.

We developed a geo-political KB for the CMCP. Specifically, we represented knowledge about international actions, terrorist groups, terrorist events, military capabilities, escalation and de-escalation of conflict, threats, and so forth. In the process of encoding the domain knowledge, we developed and adapted techniques for representing temporal knowledge and qualitative influences, and identified principles for taxonomy design.

The knowledge server development focused on extending several of our existing tools. Our theorem prover SNARK (Stickel, Waldinger et al. June 1994) was extended to accept Knowledge Interchange Format (KIF) and a subset of the OKBC knowledge model (Chaudhri, Farquhar et al. 1998), to reason efficiently with Meta classes and temporal knowledge, and to produce explanations in HTML. We developed a worldwide web (WWW) interface to the GKB-Editor, which is a graphical tool for browsing and editing KBs. We developed tools for translating, loading, and saving ontologies encoded in a subset of KIF. The ontologies were loaded into Ocelot, which is a frame representation system developed at SRI. We extended our collaboration system to deal with schema changes and to synchronize the divergent copies of a KB. Instrumentation was developed to compute statistics on KB size, reuse, and axiom creation time.

INTRODUCTION

It has been well known that encoding knowledge is time consuming and expensive. In spite of this, most Knowledge Base (KB) development efforts start from scratch, and once the project is over, the KB content is thrown away. As a result, it is difficult to amortize the cost of encoding knowledge over multiple projects and the KBs remain *brittle*, that is, limited in size and scope (Lenat and Guha 1990). To address this limitation, several complementary approaches have been attempted. Standards for representing, exchanging, and accessing knowledge have been developed (Genesereth and Fikes 1992). Large repositories of knowledge have been constructed so that they could serve as the starting point for new KB development efforts (Lenat and Guha 1990), (Knight and Luk August 1994). This project builds upon this earlier work in an effort to reduce the cost of developing KBs.

The project had three focus areas: content development techniques, new content development, and knowledge server extensions. There was a strong emphasis on content development techniques with the objective of increasing the KB construction speed. The techniques were tested in the process of constructing a KB for the crisis management challenge problem (CMCP).

A key hypothesis in the project was that if, instead of starting from scratch, we could start from an existing KB, it would be faster to construct the new KB. The new KB development task was specified using a set of *parameterized questions* that the system must be able to answer. To test the *reuse hypothesis*, we started our KB development from the HPKB upper ontology, derived from the Cyc KB, developed a new KB in the geo-political domain, and tested the KB on a set of unseen questions. We followed a reuse methodology involving the following steps: comprehension, translation, slicing, reformulation, and merging. The reuse process made extensive use of knowledge sharing standards that were already available. We made several measurements to show that KB construction by reuse is indeed possible.

After we reused as much knowledge as we could, new KB content had to be developed. To develop new content faster, we employed a technique based on axiom templates. We developed several models of commonsense reasoning for notions such as escalation, asymmetric threats, and viability of policies. By a post hoc analysis, we also identified several principles for content development, for example, guidelines for organizing taxonomies.

We performed several enhancements to our knowledge server tools. Our theorem prover SNARK was extended to accept Knowledge Interchange Format (KIF) (Genesereth and Fikes 1992) and a subset of the OKBC knowledge model, to reason efficiently with Meta classes and temporal knowledge, and to produce explanations in HTML. We developed a worldwide web (WWW) interface to the GKB-Editor (Karp, Chaudhri et al. 1999), which is a graphical tool for browsing and editing KBs. We developed tools for translating, loading, and saving ontologies encoded in a subset of KIF. The ontologies were loaded into Ocelot, which is a frame representation system. We extended our collaboration system to deal with schema changes and to synchronize the divergent copies of a KB. Instrumentation was developed to compute statistics on KB size, reuse, and axiom creation time.

The results on knowledge reuse were published and presented at the National Conference on Artificial Intelligence (Cohen, Chaudhri et al. 1999), (Chaudhri, Thomere et al. 2000). The practical issues arising in the content development were presented at the International Conference on Principles of Knowledge Representation and Reasoning (Pease, Chaudhri et al. 2000).

This report is organized as follows. We first discuss the CMCP that defined the framework for the research conducted in the project. Then, we discuss the content development techniques—knowledge reuse, axiom templates, and taxonomy design. Then, we describe some of the actual content that was developed during the project. A summary of knowledge server extensions is presented next, followed by empirical results from the evaluations. Finally, we discuss the technology transferred to project Genoa and conclude with a summary.

CRISIS MANAGEMENT CHALLENGE PROBLEM

The framework for research conducted in this project was defined by the *Crisis Management Challenge Problem* (CMCP). The CMCP was jointly developed by Information Extraction and Transport (IET) Corporation and Pacific Sierra Research (Schrag 1998), (Schrag 1999).

The CMCP was defined based on the requirements of the government analysts in the domain of crisis management. The CMCP also captured the KB content and the analyst requirements of project Genoa, a DARPA program focused on assisting analysts in the crisis management domain.

Teams led by Teknowledge and Science Applications International Corporation (SAIC) developed the systems for answering the questions. SRI was a part of the SAIC team. The primary contributors to the KB content on the SAIC team were Northwestern University, Formal Reasoning Group Stanford, Knowledge Systems Laboratory (KSL) Stanford, SAIC, and SRI. We report the work conducted primarily by SRI.

The CMCP was defined as a collection of test questions. The questions were specified using a question grammar. The grammar consisted of a set of parameterized questions, or PQs, each of which had several instantiations. A sample PQ follows.

PQ53 [During/After <TimeInterval>,] what {risks, rewards} would <*InternationalAgent*> face in <*InternationalActionType*>?
<*InternationalActionType*> =
{[exposure of its] {supporting,
sponsoring} <InternationalAgentType in <InternationalAgent2>, successful
terrorist attacks against <InternationalAgent2>'s <EconomicSector>,
<InternationalActionType>, taking hostage citizens of <InternationalAgent2>,
attacking targets <SpatialRelationship> <InternationalAgent2> with <Force>}
<*InternationalAgentType*> =
{terrorist group, dissident group, political party, humanitarian organization}

A sample instance of this PQ is, "What risks may Iran expect in sponsoring terrorist attacks in Saudi Arabia?" A sample answer would be that sponsoring terrorist attacks is a violation of international norms; therefore, sponsoring terrorist attacks may lead to sanctions against Iran.

The evaluation during the first year of the project had the following phases.

1. Some months before any testing began, a *crisis scenario* was released. The scenario bounded the domain and thus the scope of the problems to be solved.
2. Several weeks before testing, a batch of sample questions (SQs) was released. On the first day of the evaluation, a batch of 110 test questions, TQA, was released, and the Teknowledge and SAIC systems were immediately tested. After four days for improvements, the systems were retested on TQA.

3. Batch TQB was released at the time of the retest. The purpose of TQB, which contained questions similar to those in TQA, was to check the generality of the improvements made to the systems.
4. After a brief respite, a change was made to the crisis scenario, increasing the scope of the problems that the Teknowledge and SAIC systems would have to solve. Several days were allowed for knowledge entry prior to the release of a new batch of questions, TQC, reflecting the new scope. The systems were tested immediately.
5. Four days were allowed to extend the systems to the new crisis scenario, and then the systems were retested on TQC. To check the generality of these extensions, the systems were also tested on batch TQD, which was similar to TQC.

During the first year of the project, all the testing was done at year's end, which did not allow enough time for making improvements to the system in response to the problems encountered during the process. As a result, during the second year, the testing was spread over several months with two mini-evaluations after 8 and 10 months, and a final evaluation at the end of the year.

The primary focus of the evaluation was to measure the competence of the KB content developed during the project. The competence was measured by the quality of the answers produced by the system as graded by a panel of subject matters experts (SMEs). The competence scores were a function of three factors: correctness, explanation quality, and citation of knowledge sources. There were several optional criteria for scoring, for example, lay intelligibility of answers, online availability, and novelty of knowledge sources.

In addition to the competence scores, several metrics about the KB development process were computed. Two prominent metrics were KB size and reuse rates. We report the metrics on competence scores in the section on KB content and the metrics on the KB development in the section on knowledge reuse.

CONTENT DEVELOPMENT TECHNIQUES

Several technical issues arose while we were developing the KB content. Knowledge reuse and axiom templates were used extensively during the project. The principles for taxonomy design were developed by a post hoc analysis of the KB. KB modularization and compositionality were important in the process, but no specific solutions were developed to address them.

Knowledge Reuse

Our knowledge reuse methodology involved the following steps: (1) translation, (2) comprehension, (3) slicing, (4) reformulation, and (5) merging. The early focus of our work was on reusing the HPKB upper ontology (HPKB-UL). The HPKB-UL was the starting point for our KB development because it contains a broad coverage of concepts needed for the target application. The HPKB-UL was derived from the Cyc and Sensus KBs, and contains roughly 3000 terms. After the first year of the project, our KB was merged with the KBs of others. Thus, the merging step of knowledge reuse played a role only at the end of the first year. Here, we describe each of these steps in detail, and then give measurements on the kind of reuse achieved during this project.

Translation

The HPKB-UL was released in the MELD format (a language used by Cycorp) and was not directly readable by our system. In conjunction with the Knowledge System Laboratory (KSL) Stanford, we developed a translator to load the HPKB-UL into any OKBC-compliant server. This translator handles structural information in the KB. The structural information includes classes, functions, relations, class-subclass relationships, and facets. While doing this translation, we had to define equivalence between some relation names in the HPKB-UL and the OKBC ontology. For example, the relation *#$genls* in the HPKB-UL is equivalent to the relation *subclass-of* in the HPKB ontology. We also converted the case-sensitive names from the HPKB-UL to case-insensitive names. For example, the constant name *#$performedBy* from the HPKB-UL was mapped to *performed-by*. The syntactic translation was a low-effort engineering task, and accounted for a small fraction of the KB development time.

Comprehension

Before a knowledge engineer reuses an ontology, its contents and organization must be understood. Two things enabled the ontology comprehension. First, the extensive documentation accompanying the Integrated Knowledge Base Environment (IKB) clarified many design decisions and presented the KB content in many different ways. Second, once the HPKB-UL was loaded into our OKBC server, Ocelot, we used the GKB-Editor to comprehend it. The taxonomy and the relationship browsers of the GKB-Editor were instrumental in helping us understand the interrelationships between classes and predicates of HPKB-UL. During the KB development process, the GKB-Editor's browsing capabilities were extensively used to search for necessary concepts and to identify the proper location in the taxonomy for a new concept.

Slicing

Slicing involves selecting a portion of an input ontology for use in a new application. For many reasons, using all of the input ontology may not be desirable. First, all of the input ontology may not be needed for a new application. Second, importing all of it may make the resulting KB unnecessarily complex. Third, there may be aspects of the input ontology that the target application is unable to handle, and these must be removed. Finally, some of the representation decisions made in the input ontology may not be acceptable to the target application.

Two technical problems must be solved for slicing. First, we must decide what portions of the input ontology we need to slice. We call the portion of the input ontology that needs to be extracted the *seed*. Second, we need a computational procedure that extracts out just the right amount of terms from the input ontology.

More formally, an ontology contains a set O of sentences. The set O consists of sentences of the following form.

- *(class X),* where $X \in C$, and C is a set of classes
- *(relation X),* where $X \in R$, and R is a set of relations
- *(function X),* where $X \in F$, and F is a set of functions
- *(individual X),* where $X \in I$, and I is a set of individuals
- *(subclass-of X Y),* where $X,Y \in C$
- *(instance-of X Y),* where $X \in C \cup I$, and $Y \in C$
- *(arity X N),* where $X \in R \cup F$, and N is a natural number
- *(nth-domain X N Y),* where $X \in R \cup F$, N is a natural number, and $Y \in C$
- *(range X Y),* where $X \in F$, and $Y \in C$
- *(nth-domain-subclass-of X N Y),* where $X \in R \cup F$, N is a natural number, and $Y \in C$
- *(range-subclass-of X Y),* where $X \in F$, and $Y \in C$
- *(r X V),* where $r \in R$
- *(template-slot-value X Y V),* where $X \in C$, $Y \in R$, and *(arity R 2)* is in O

The relation symbols *class, individual, subclass-of, instance-of,* and *template-slot-value* have meanings as defined in the OKBC specification. The relation symbols *range, nth-domain, arity, range-subclass-of,* and *nth-domain-subclass-of* have meanings as defined in the Ontolingua frame ontology.

The seed S is a set containing sentences of the form *(class X), (relation X), (function X),* and *(individual X),* where $X \in C \cup R \cup F \cup I$. Based on the knowledge of the target application, we were able to identify S. It is natural that all needed terms may not be included in S in the initial estimate. As the KB evolves, if additional terms are needed, the seed can be revised, and the slice recomputed. In practice, it was sufficient to re-compute the slice once every six months.

The slice L is a subset of O. We would like to compute L in a way that all the useful information from the source ontology is incorporated into the KB being developed. We call a slice *maximal with respect to S* if any inferences involving S that can be performed using *O* can be performed using *L*. We call a slice *L* that is maximal with respect to S as *minimal with respect to S* if there is no $L' \subset L$ that is maximal with respect to S.

A trivial way to compute *L* is to simply return *S*. In general, *S* is not a maximal slice of *O* with respect to *S*. Let us define an algorithm to compute the maximal slice of *O,* with respect to *S.*

Algorithm *MaximalSlice*
Input: Input ontology *O,* and seed *S*
Output: *L,* a slice of *O,* with respect to *S*

```
1. Let $S' = \{X \mid (class\ X) \in S$, or $(relation\ X) \in S$, or $(function\ X) \in S$, or $(individual\ X) \in S\}$.
2. Set $L = S$.
3. For every $X \in S'$, if $(nth\text{-}domain\ X\ N\ Y) \in O$, add $Y$ to $S'$, and add $(class\ Y)$ and $(nth\text{-}domain\ X\ N\ Y)$ to $L$.
4. For every $X \in S'$, if $(nth\text{-}domain\text{-}subclass\text{-}of\ X\ N\ Y) \in O$, add $Y$ to $S'$, and add $(nth\text{-}domain\text{-}subclass\text{-}of\ X\ N\ Y)$ and $(class\ Y)$ to $L$.
5. For every $X \in S'$, if $X \in F$, if $(range\ X\ Y) \in O$, add $Y$ to $S'$, and add $(range\ X\ Y)$ and $(class\ Y)$ to $L$.
6. For every $X \in S'$, if $X \in F$, if $(range\text{-}subclass\text{-}of\ X\ Y) \in O$, add $Y$ to $S'$, and add $(range\text{-}subclass\text{-}of\ X\ Y)$ and $(class\ Y)$ to $L$.
7. For every $X \in S'$, if $X \in C$, if $(subclass\text{-}of\ X\ Y) \in O$, add $Y$ to $S'$, and add $(subclass\text{-}of\ X\ Y)$ and $(class\ Y)$ to $L$.
8. For every $X \in S'$, if $(instance\text{-}of\ X\ Y) \in O$, add $Y$ to $S'$, and add $(instance\text{-}of\ X\ Y)$ and $(class\ Y)$ to $L$.
9. For every $X \in S'$, if $(r\ X\ V) \in O$, add $(r\ X\ V)$ to $L$. If $(class\ V) \in O$, add $(class\ V)$ to $L$, and $V$ to $S'$. If $(individual\ V) \in O$, add $(individual\ V)$ to $L$, and $V$ to $S'$.
10. For every $X \in S'$, if $(template\text{-}slot\text{-}value\ X\ r\ V) \in O$, add $(template\text{-}slot\text{-}value\ r\ X\ V)$ to $L$. If $(class\ V) \in O$, add $(class\ V)$ to $L$, and $V$ to $S'$. If $(individual\ V) \in O$, add $(individual\ V)$ to $L$, and $V$ to $S'$.
11. Repeat steps 7 through 10 until $L$ does not change.
12. Return $L$.
```

Theorem 1: The algorithm *MaximalSlice* produces a slice *L* of *O,* which is maximal with respect to *S.*

Theorem 2: The algorithm *MaximalSlice* is polynomial in the size of *C.*

Intuitively, the algorithm *MaximalSlice* works by first determining all the relevant classes, and then computing their upward closure in the graph of taxonomic relationships.

It is possible to produce a smaller slice if one has additional knowledge about the sorts of axioms that are of interest for the target application. For example, suppose *X* is a *subclass-of Y*, and *Y* is a *subclass-of Z,* and that *X* is in the seed, but *Y* and *Z* are not. If *Y* does not have associated with it any (interesting) axioms, it may be dropped from the closure by asserting *X* as a *subclass-of Z.* This does not change any inferences of interest that can be performed about *X*.

An example definition of *interestingness* is as follows.

Definition 1. A class *X* is of interest with respect to a seed S if one of the following holds.

- X is the root of the class-subclass graph
- The sentence *(nth-domain Y N X)* is in *O*, and *Y* is in *S*
- The sentence *(nth-domain-subclass-of Y N X)* is in *O*, and *Y* is in *S*
- The sentence *(range Y X)* is in *O*, and *Y* is in *S*
- The sentence *(range-subclass-of Y X)* is in *O*, and *Y* is in *S*
- There exists a sentence *(template-slot-value X r V)* or *(r X V)* in *O*

Using this definition, one can compute an interesting superclass of a class *X* as follows. For every *(subclass-of X Y)* sentence in *O*, the superclass *Y* is interesting if *Y* is of interest. If *Y* is not of interest, check to see if *Z* is of interest, where *(subclass-of Y Z)* is in *O*. If *Z* is of interest, *Z* is an interesting superclass of *X*. For a rooted and connected taxonomy this process is guaranteed to terminate. An interesting type of a class and an individual may be computed analogously.

Algorithm *MinimalMaximalSlice*
Input: Input ontology *O,* seed *S,* and slice *L* produced by *MaximalSlice*
Output: *L*, a slice of *O*, with respect to *S*

1. Let *S'* = {*X* | *(class X)* $\in$ *S,* or *(relation X)* $\in$ *S,* or *(function X)* $\in$ *S,* or *(individual X)* $\in$ *S.*
2. Set *L* = *S.*
3. For every *X* $\in$ *S',* if *X* $\in$ *R*$\cup$*F,* and if *(nth-domain X N Y)* $\in$ *D,* add *(class Y)* to *L,* and *Y* to *S'.*
4. For every *X* $\in$ *S',* if *X* $\in$ *R*$\cup$*F,* and if *(nth-domain-subclass-of X N Y)* $\in$ *D,* add *(class Y)* to *L,* and *Y* to *S'.*
5. For every *X* $\in$ *S',* if *X* $\in$ *F,* and if *(range X Y)* $\in$ *G,* add *(class Y)* to *L,* and *Y* to *S'.*
6. For every *X* $\in$ *S',* if *X* $\in$ *F,* and if *(range-subclass-of X Y)* $\in$ *G,* add *(class Y)* to *L,* and *Y* to *S'.*
7. For every *(subclass-of X Y)* in *L,* compute interesting parent Z, add *(subclass-of X Z)* and *(class Z)* to *L,* and *Z* to *S'.*
8. For every *(instance-of X Y)* in *L,* compute interesting parent Z, add *(subclass-of X Z)* and add *(class Z)* to *L,* and *Z* to *S'.*
9. For every *X* $\in$ *S',* if *(r X V)* $\in$ *O,* add *(r X V)* to *L.* If *(class V)* $\in$ *O,* add *(class V)* to *L,* and *V* to *S'*. If *(individual V)* $\in$ *O,* add *(individual V)* to *L,* and *V* to *S'.*
10. If *(template-slot-value X r V)* $\in$ *O,* add *(template-slot-value r X V)* to *L.* If *(class V)* $\in$ *O,* add *(class V)* to *L,* and *V* to *S'.* If *(individual V)* $\in$ *O,* add *(individual V)* to *L,* and *V* to *S'.*
11. Repeat steps 7 through 10 until *L* does not change.
12. Return *L.*

Theorem 3: The algorithm *MinimalMaximalSlice* produces a slice *L* of *O*, which is minimally maximal with respect to *S* assuming the interestingness of a class as defined in Definition 1.

During the first year of the project, we used the trivial slice of the HPKB-UL, that is, we just used the constant names and documentation strings. During the second year, we used *MaximalSlice*. The motivation for *MinimalMaximalSlice* was to argue the point that while reusing an ontology, it is not necessary that the end user agree with everything in the source ontology, especially those terms and representations that can be *sliced* away. The terms that can be sliced away are like the binary code that never needs to be exposed to the knowledge engineer. Slicing is, thus, a way to modularize a taxonomy in a way that hides the irrelevant details from the knowledge engineer reusing it. Slicing is not a replacement for the approaches to modularizing a KB that argue in favor of constructing a KB from representational components (Clark and Porter 1997). A sliced HPKB-UL could serve as a starting point for a KB that is constructed from components.

Reformulation

Reformulation is the process of taking an input theory and transforming its representation. A common reason for reformulation is that in the target system the reformulated theory may be more efficient to reason with than the original theory. We reformulated HPKB-UL to convert every *Functional Predicate* into a relation. We explain this reformulation in more detail.

HPKB-UL represents the functional relationships as predicates. For example, even though *mother* is a function, it is represented in HPKB-UL as a relation. Such predicates are instances of the class *Functional Predicate* in the HPKB-UL. There are two differences between using functions and using relations.

First, using functions gives a more compact representation in many situations. Functions are a more compact way to state that the relationship between two objects is single-valued and that when a function is applied to an object (or a set of objects), the value indeed exists. To assert the same information using relations, one needs to also specify that the cardinality of the relation is 1. Thus, when we represent *mother* as a function, we are guaranteed that every individual has one and only one mother. When we represent *mother* by a relation, we do not get any such guarantees unless we also assert the cardinality constraint on it.

Second, using functions, the paramodulation rule of inference can be applied. The benefits of using functions are enhanced while using equality reasoning. SNARK, like most theorem provers, uses the paramodulation rule of inference while reasoning with equality. The paramodulation rule, given an assertion such as $a = b$, allows us to simplify a formula such as *(R a)* to *(R b)*. If we use functions instead of relations, it introduces equality in the KB. For example, *(mother sue john)* is replaced by *sue = mother (john)*. SNARK is then able to use the paramodulation rule of inference for reasoning with such formulas. The paramodulation rule of inference can sometimes lead to faster and shorter proofs. But in other cases, the search space can become larger.

The tradeoff between functions and relations has created the following open problems for future research.

1. In the past, we just imported classes, functions, and relations from the HPKB-UL. If we were going to import general axioms, and were still going to reformulate functional predicates to relations, we would need to convert every occurrence of a functional predicate into a function. The technique to accomplish a two-way transformation to do this needs to be developed.
2. Another alternative would be to indicate to SNARK the single-valued-ness property of a relation without requiring that it be rewritten using functions. If that is possible, we do not need to do this reformulation at all.
3. Apart from the representation economy, the actual benefits of using functions over relations for a KB such as the one developed during this project remain unclear. This needs to be investigated further.

Merging

Merging involves ensuring that the merged KBs use the same terms when they mean the same thing, and that they represent the same information identically. Merging assumes the existence of two independently developed KBs. Translation, slicing, and reformulation steps usually precede merging. The merging effort for the SAIC team was led by KSL Stanford. We invested significant effort in making our KB available for merging, in resolving the conflicts arising from the merge, and in using the results of the merge.

For example, the merge process revealed that the KBs developed by SRI and KSL both represented situations in which one agent supports or opposes another. To represent an action, in which one agent supports another action, say a terrorist attack, there are two alternatives. First, one can define a class *supporting-terrorist-attack* and create an individual member of this class to represent an instance of a supporting action. Second, one can define a slot called *supports* on the class *action*, which can take an instance of *terrorist-attack* as a value. If the KBs to be merged use these different representations for supporting actions, then the merge phase should either use one representation over the other or add axioms defining equivalence between the two representations. In the current merge, a meeting was organized between KSL and SRI to resolve this difference, and the solution that defines the slot *supports* on class *action* was adopted. In general, the semantic merge phase can involve significant manual intervention.

We believe there has been an over-emphasis on trying to reach an agreement on the meaning of terms, which is not really a technical problem. Any tools designed to identify terms that mean the same thing are not likely to go very far—the term names are like variable names in a program, and one cannot make conclusions just by looking at the names. We believe that merging will not be a bottleneck in future KB development projects for two reasons. First, if we follow a KB development process that is strongly grounded in reuse, the question of terms meaning different things does not arise. Second, the same term meaning different things reflects a fundamental flaw in the design of the core theories in a KB. Therefore, one should first fix the core theories in the KB instead of fixing the consequences of the flaws in the design of core theories.

Experimental Results on Knowledge Reuse

We now present some experimental results to characterize the knowledge reuse achieved during the project. We have three objectives in presenting these results. First, we want to characterize the KB development process and the contribution of each of the steps in knowledge reuse. Second, we want to give specific examples of knowledge reuse to highlight that the results are based on some nontrivial examples of reuse. Finally, we will show the level of reuse for different domain areas.

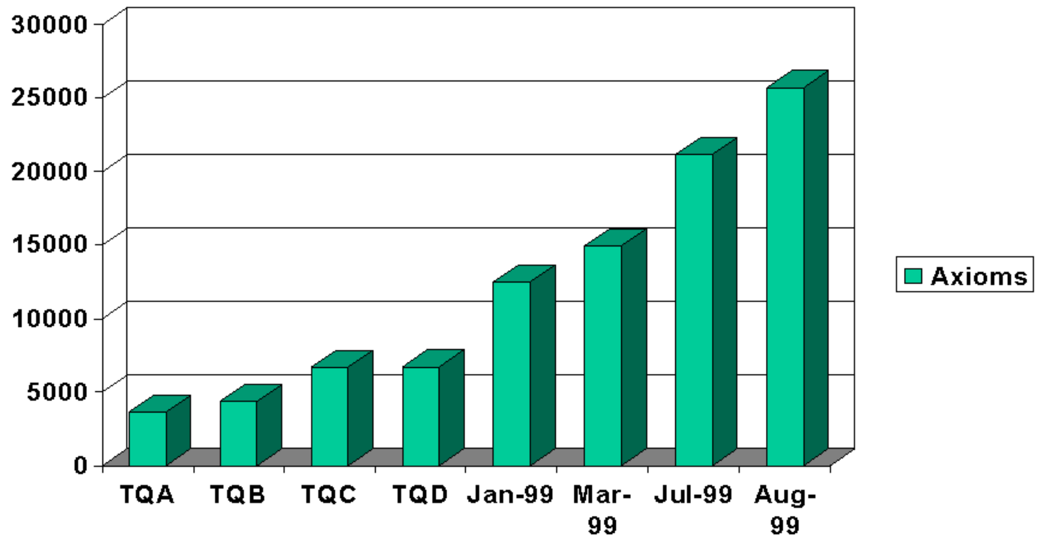


Figure 1. Growth in KB size during the project

In Figure 1, we show how the KB size grew during the project. TQA represents the time just before the testing began during the first year. At that time, the KB contained roughly 3000 axioms. TQD represents the point at the end of the first year when the KB contained roughly 7000 axioms. The final KB contained about 30,000 axioms.

An overview of the KB development process is shown in Figure 2. The KB development started from the HPKB-UL. We took a slice of the HPKB-UL that was extended to create the KB CMCP-98. Two other KBs were developed independently at Stanford – the *Supports* KB was developed by KSL, and the *Capability* KB was developed by the Formal Reasoning Group. At the end of the first year of the project, these KBs were merged to produce a new KB called the *SAIC merged ontology*. The development for the second year started by taking a slice of the SAIC merged ontology, which was extended to produce CMCP-99, the KB for the second year of the project.
The HPKB-UL had about 16,434 axioms, and our initial slice of it contained 446 axioms. The CMCP-98 KB had 5943 axioms. The SAIC merged ontology contained 21,223 axioms. (We have excluded many ground facts from this count.) The slice of the merged

ontology that was used for further development contained 5360 axioms. The CMCP-99 KB contained 22,902 axioms. The slice of HPKB-UL was recomputed three times over a period of two years. The final slice contained 2544 axioms. These numbers show that the slicing significantly reduced the size of the KB that we needed to work with, and it would have been a wasted effort to try to reach an agreement on all of the HPKB-UL.

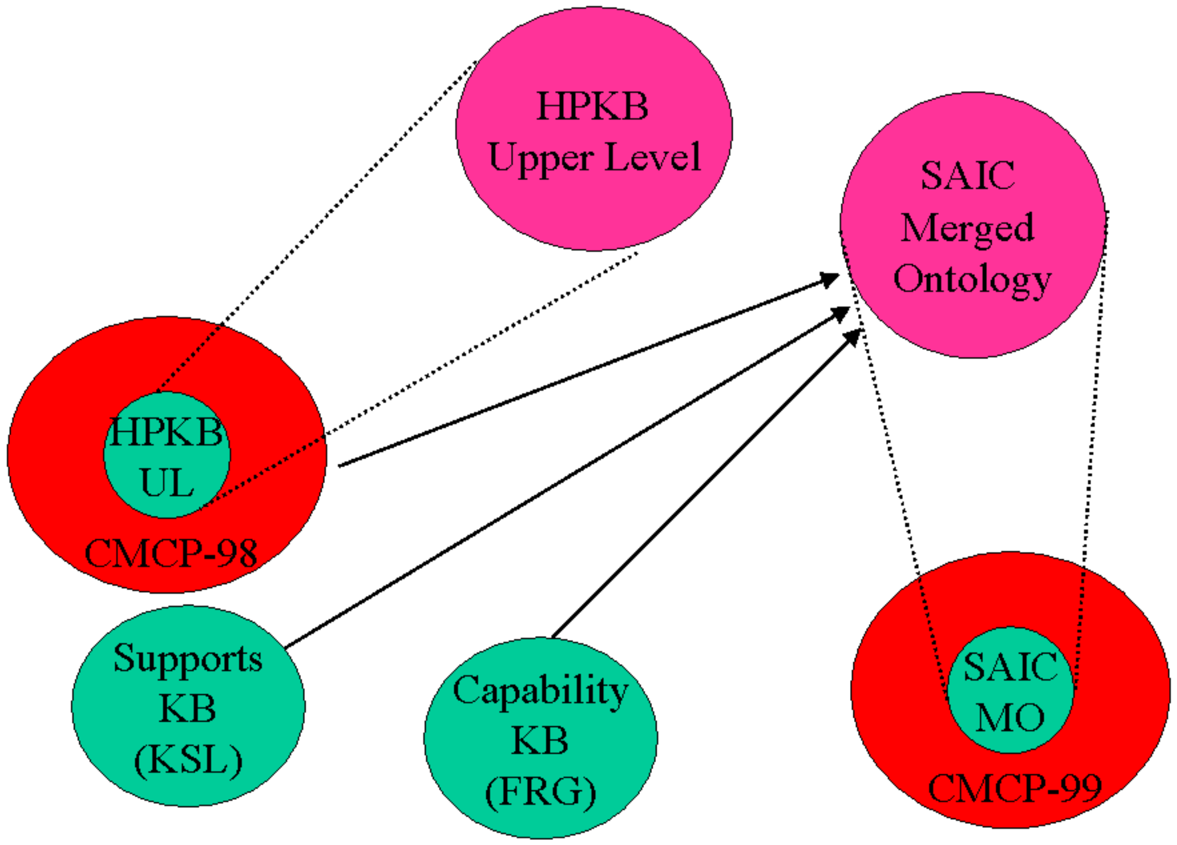


Figure 2. Overview of KB development process

To highlight the nature of reuse, we consider two example representations. One of the questions answered by our system was

Has post-Shah Iran launched ballistic missiles in wartime?

The upper ontology does not have a concept representing ballistic missiles, and therefore we created a new class to represent it. The *ballistic-missile* is a subclass of the class *weapons*, a class already existing in the HPKB-UL. The verb “launch”' in the questions was mapped to the action *attack*, which is a subclass of an already-existing class *hostile-social-action*. The class of actions in the HPKB-UL has several slots. The slot *performed-by* on an action specifies the doer of the action, and *device-used* specifies the tool that was used in performing that action. These slots are already defined in the HPKB-UL. Finally, we specify the temporal context of the question by defining a constant representing the time *post-shah-iran* and then using the temporal comparison operators *later-than* and *start-of,* which are available in the HPKB-UL. The resulting formalization is as follows.

```
(and
  (attack ?act)
  (performed-by ?act Iran)
  (value-type device-used ?act ballistic-missile)
  (later-than (start-of ?act)
  (start-of post-shah-iran)))
```

The formalization of this question, thus, reuses primitives for representing temporal knowledge and slots on actions. As another example, consider the question

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

To answer questions of this type, we developed a simple cause-effect model. Our model is based on five predicates—*cause-event-event* to represent the effects that are *definitely* caused by an action, *may-cause* to represent the effects that *may* be caused by an action, *may-prevent* to represent actions that are *prevented* by an action, *maleficiary* to represent the risks of an action, and *beneficiary* to represent the benefits. The predicates *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 empirical results that we present here are based on a reuse metric that has been proposed previously (Cohen, Chaudhri et al. 1999). The metric can be computed for either axioms or terms. Suppose that a knowledge engineering task requires *n* terms and *k* of those can be reused from an existing KB; then, *k/n* measures the extent of reuse of the KB. We measured *k/n* both for KB construction and for the axioms used for answering the questions. The results are shown in Table 1. We computed the reuse with respect to the HPKB-UL, the KB at the end of the first year, and the KB that existed just before the testing began. The reuse is computed for two kinds of objects in the KB: constants and structural statements. Constants include any class, relation, function, or individual in the KB. The structural statements include atomic statements that use the relations *subclass-of, instance-of, nth-domain, arity, subrelation-of, disjoint-with,* and *arity.* The reuse for other kinds of axioms—for example, implications or ground facts—was computed but is not reported here, primarily because the HPKB-UL does not contain any implications or non-ground statements. These numbers show approximately 15% reuse from HPKB-UL, 35% reuse from the Y1 KB, and 80% reuse from the pre-evaluation KB. The reuse from the HPKB-UL is especially interesting because it was not designed with the current application in mind. The reuse for axioms actually used in answering the questions and for the axioms in the KB remained roughly the same for HPKB-UL, but was somewhat different for the pre-evaluation KB.

Table 1. Gross *k/n* for axioms in the KB and the axioms actually used to answer questions

With respect to KB	*Constant Reuse in Constructing the KB*					
	Structural	*Implications*	*Non-Implications*	*Ground Facts*	*Overall*	
HPKB-UL	0.22	0.29	0.13	0.12	0.16	
Y1 KB	0.18	0.33	0.19	0.22	0.21	
Pre-evaluation KB	0.49	0.78	0.39	0.62	0.62	
	Axiom Reuse in Answering the Questions					
	Constants	*Structural*	*Implications*	*Non-Implications*	*Ground Facts*	*Overall*
HPKB-UL	0.17	0.12	0.05	0.0	0.0	0.08
Y1 KB	0.21	0.09	0.05	0.0	0.01	
Pre-evaluation KB	0.67	0.76	0.43	0.0	0.68	0.71

We give here semiformal measurements as an indication of *k/n* by domain. (Since our KB was not organized that way, it was difficult to produce precise data by domain.) The HPKB-UL had a complete coverage for representing the temporal knowledge and for representing paths. Any primitive that we needed for representing time was present; therefore, *k/n* for temporal knowledge was 1.0. Similarly, we needed primitives to represent paths, such as pipeline from A to B, all of which were available in the HPKB-UL. For spatial knowledge, numerous primitives—for example, *borders-with*—were available, and others—such as *eastern, western*—needed to be defined. Similarly, for representing actions and interests, numerous required classes were available, whereas others needed to be defined.

Scope for Future Work

In this project, we reused only constants, and structural statements from the HPKB-UL, and did not reuse any statements containing implications and non-ground facts that are available in IKB. A hypothesis for future work would be that since our KB shares the upper structure with IKB, it would enable us to share the implications and non-ground facts from the HPKB-UL with greater ease. Exploring this hypothesis would also require us to extend our work on slicing and reformulation. The slicing techniques would need to be extended to slice out the relevant rules. The representation differences are likely to have a greater impact on rules than they had on the class-subclass structure, and therefore, new reformulation techniques will need to be developed.

Apart from the technical issues associated with reuse, there are human issues. Knowledge engineers prefer their own representations to reusing someone else's. In many cases, using a different representation does not necessarily contribute to the overall system and makes it difficult to scale the scope of a KB. We reuse other people's software routinely as long as it is well packaged, has a clear functionality, and adds value to our work.

Packaging knowledge components in a way that they offer a clear functionality, and can be reused without any concern for the internal design decisions has been an unrealized dream for the knowledge sharing community for a long time and is still open for future work.

Axiom Templates

After reusing the content available from the existing sources, we developed new KB content needed for the solution of the CMCP. While developing the new content, we explored ways to allow faster creation of content, and to make it easier for a non-logic expert to contribute to content creation. The axiom templates were developed in this process. The axiom templates were also motivated by the fact that the CMCP questions were structured as instantiations of a template, and several instantiations of a template could be answered using a similar reasoning pattern.

Examples of Axiom Templates

To design an *axiom template,* we started by generating a handful of instances and by representing knowledge to answer them. In the process, we identified *the axiom templates* that were common to several instantiations. These axiom templates were then populated. Some instances of certain PQs did not fit into the template structure. Custom solutions were designed for such cases. To illustrate this technique, let us consider the parameterized question 236 (PQ236) as an example.

PQ236 What {types of interest, <AnalyticalFactor>} typically underlie an <InternationalAgentType>'s decision to <InternationalActionType>?

Here are a few example instantiations of this question.

PQ236-1 What types of interest typically underlie a country's decision to attack targets in a country?

PQ236-2 What types of interest typically underlie a country's decision to conduct a diplomatic action favoring a country?

PQ236-3 What types of interest typically underlie a terrorist group's decision to conduct a military action against a country?

PQ236-4 What economic interests typically underlie a country's decision to conduct an economic action against Iran?

PQ236-5 What types of interest typically underlie the decision of an enemy of Iran to oppose Iran's interest in construction of a major oil pipeline to Bandar Abbas, Iran from Baku, Azerbaijan through Iran?

PQ236-6 What military capability typically underlies the decision of a leader of Iran to conduct navy maneuvers?

PQ236-7 What potential diplomatic reward typically underlies a country's decision to impose economic sanctions against Iran?

PQ236-8 What country's interest in the Baku-Ceyhan pipeline typically underlies the decision of a leader of Iran to increase military sponsorship of the Arabian Liberation Army (ALA)?

PQ236-9 What national interests typically underlie a country's decision to share intelligence about a terrorist group with a country?

PQ236-10 What types of interest typically underlie a terrorist group's decision to take hostage a citizen of a country?

Agents act to further one or more of their interests. Therefore, we need a general model for representing interaction of a class of actions with a class of interests. We expect a large number of instantiations of this PQ to be answerable, using that general model. To capture this interaction, we introduce two relations *may-positively-influence* and *may-negatively-influence*. The interests typically underlying an action are those that are positively influenced by that action. To capture the action interest interaction, we designed the following axiom template.

If an ?agent performs an ?action
then
There must exist an ?interest such that ?action positively influences ?interest (A1)

Here is an example instantiation of this template:

```
(forall ((?attack attack) (?country country) (?country1 country))
  (=>
   (and                                                            A
     (performed-by ?attack ?country)
     (opposing ?attack ?country1))
   (exists ((?deterring-aggression deterring-aggression))
     (and                                                                  B
     (interest-of ?deterring-aggression ?country)
     (may-positively-influence ?attack ?deterring-aggression)))))          (A2)
```

This template by itself does not directly give the desired inference. We also need the following rule.

```
(forall ((?action action) (?interest interest) (?agent agent))
   (=>
     (and
       (may-positively-influence ?action ?interest)
       (has-mental-object-of ?interest ?agent)
```

```
        (performed-by ?action ?agent))
(typically-underlies ?interest ?action))                    (A3)
```

With reference to Figure 3, the antecedant represents the set of literals *A*, and the consequent the set of literals *B.* Both *A* and *B* directly map to objects of interest in the domain. Later in this section, we will show how by a rearrangement of quantifiers, a different relationship between the same set of literals holds.

From the above instances of PQ236, PQ236-1 through PQ236-4, PQ236-9, and PQ236-10 are amenable to solution by this axiom template. PQ236-5 is highly context dependent and requires the system to have knowledge about the economic benefits of pipelines. Furthermore, the system must know that enemies oppose actions that benefit their opponents. In addition, we may need to use some instantiation of the above rule template. PQ236-6 is semantically different from other instantiations because it asks about military capabilities, not interests. PQ236-7 asks about rewards, not interests, and PQ236-8 about a specific country and not about an interest.

We focused on those instances that were amenable to solution by the rule template for three reasons. First, the rule template made extensive use of the International System Framework (ISF) document (Jermano and Picarelli 1998), which is one of the knowledge sources supplied with the CMCP document. Second, the rule templates can be easily populated by a relatively less skilled knowledge engineer (KE). Third, once an instance of the template has been debugged with the theorem prover, new instances do not entail any extra debugging cost.

As another example use of the axiom template, consider PQ237.

PQ237. What types of international action does a desire toward <InterestEffectType><InterestType> typically lead to?

Here are a few example instances of this PQ:

PQ237-1. What types of international action does a desire toward strengthening of deterrence against {...} aggression typically lead to?

PQ237-2. What types of international action does a desire toward weakening of stability in a region typically lead to?

PQ237-3. What types of international action does a desire toward development of weapons of mass destruction {...} typically lead to?

PQ237-4. What types of international action does a desire toward containment of Iran's {...} regional influence {...} typically lead to?

PQ237-5. What types of international action does a desire toward improvement of domestic economic stability {...} typically lead to?

PQ237-6. What types of international action does a desire toward strengthening of military typically lead to?

PQ237-7. What types of international action does a desire toward improvement of managing immigration and emigration typically lead to?

PQ237-8. What types of international action does a desire toward increase in {...} international influence {...} typically lead to?

PQ237-9. What types of international action does a desire toward increase in conventional weapons typically lead to?

PQ237-10. What types of international action does a desire toward improvement of diplomatic commitments to a country typically lead to?

To answer this PQ and several of the instantiations listed above, one can use an axiom template as follows.

If an agent has an interest ?interest
then
there exists an ?action such that ?action positively influences that ?interest
(A4)

```
(forall ((?deterring-aggression deterring-aggression) (?country country) (?country1 country))
  (=>
    (and
     (interest-of ?deterring-aggression ?country)
     (may-positively-influence ?attack ?deterring-aggression))
  (exists ((?attack))
     (and
       (performed-by ?attack ?country)
       (opposing ?attack ?country1)))))                     (A5)
```

```
(forall ((?action action)(?interest interest))
 (=>
  (may-positively-influence ?action ?interest)
  (typically-leads-to ?interest ?action)))                  (A6)
```

Axioms A2 and A5 have the same collection of literals except that the literals that appear in the antecedent of A2 appear in the consequent of A5 and vice versa. Axioms such as A2 and A5 can be automatically generated from the input such as

(action-interest-interaction deterring-global-threats imposing-sanctions country)

```
(action-interest-interaction INTEREST-IN-INFLUENCE-IN-OTHER-STATES
imposing-sanctions country)
(action-interest-interaction INTEREST-IN-PUNISHING imposing-sanctions country)
(action-interest-interaction MAINTAIN-OPEN-MARKETS imposing-sanctions country)
(A7)
```

It is quite easy for a non-expert in KR to produce the input as shown above, which can then be expanded into axioms such as A2 and A5.

The solution of PQ237-2 requires the use of a template that uses a *may-negatively-influence* relation.

```
(forall ((?maintaining-regional-stability maintaining-regional-stability)
         (?country country))
  (exists ((?supporting-terrorist-groups supporting-terrorist-groups))
     (=>
       (HAS-MENTAL-OBJECT-OF ?maintaining-regional-stability ?country)
     (and
       (may-negatively-influence ?supporting-terrorist-groups ?maintaining-regional-stability)
       (performed-by ?supporting-terrorist-groups ?country)))))          (A8)
```

```
(forall ((?action action) (?interest interest))
(=>
  (may-negatively-influence ?action ?interest)
  (has-typical-action-effect ?action ?interest weakening-effect)))      (A9)
```

If an action may negatively influence an interest, then that action has a decreasing effect on that interest.

```
(forall ((?action action)
            (?effect effect)
            (?interest interest)
            (?thing thing))
           (=>
            (and
             (has-action-effect ?action ?thing ?effect)
             (has-mental-object-of ?interest ?agent)
             (performed-by ?action ?agent)
             (has-interest-in-thing ?interest ?effect))
            (may-positively-influence ?action ?interest)))              (A10)
```

An interest in an effect is positively influenced by an action that can achieve that effect. The positive influence in this context has the connotation of ‘enabling’. For example, if an agent is interested in decrease of production, and an action has the ‘negative’ influence on the production, by decreasing it, then the influence of that action on the interest is indeed positive.

The solution of PQ237-2 requires the use of axioms over and above the ones derived from the template. This suggests that in many cases, the axiom template could be a part of lengthier inference chains. Based on these examples, we are now in a position to give a definition to axiom templates.

Definition of an Axiom Template

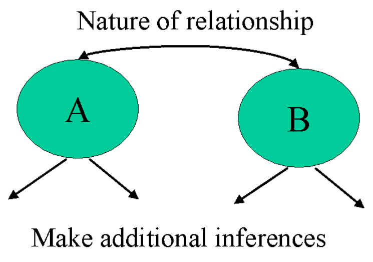


Figure 3. Abstract view of an axiom template

An axiom template is a rule with a special structure: it contains a collection of literals A and B that are related by an implication symbol. Sometimes, an implication in the opposite direction may exist by a minor rearrangement of quantifiers over the same set of literals. It should be possible to systematically generate the sets of literals A and B. For example, they may correspond to a set of objects that exist in the domain of application. The inference involving the axiom template may be a part of a lengthier chain of inference.

Experimental Results in Using Axiom Templates

The ISF document listed the actions and interests in the international domain. We developed an ontology of actions and interests based on this document. Using that ontology, we prepared a table listing classes of interests on one side and classes of actions on the other side. The table was then sent to the SMEs at IET, who filled in the interaction between the classes of action and the classes of interest. Based on the table, SAIC populated the axiom template. The resulting instances of the axiom template were then loaded into SNARK and tested during the evaluations.

The results of this approach were encouraging. The most positive aspect of the results was that, for each test question for this PQ, we were able to produce several answers. We were, however, not able to get a perfect score for correctness and quality of answers. Our average score for correctness and accuracy was two out of a maximum of three. The general criticism was that our answers were too general. That was to be expected because the only knowledge that was represented was the interaction between actions and interests.

Here is a specific example of PQ236 that was asked during the evaluation:

What types of interest typically underlie a country's decision to impose economic sanctions against an ally?

It was possible for us to return an answer suggesting the interests that might underlie a country's decision to impose sanctions against another country, but the answer did not cover the situation when the sanctions are imposed against an ally.

Scope for Future Work

A KB derived from axiom templates cannot deal with nuances of a new situation unless significantly more detail is added. A possible way to add more detail is to represent the conditions under which the influences hold by using techniques from qualitative reasoning (Forbus 1984), (Rickel and Porter 1999). Such techniques have to be accompanied by a large collection of models representing the domain knowledge. In the above example, we need representation for allies, and the fact that countries will not normally perform negative actions against allies. We hope to investigate the use of qualitative models, along with models of domain knowledge, in our future work.

Taxonomy Design

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 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, some concepts had incorrect positions in the taxonomy.

- Some concepts were too high. A class A is too high in a taxonomy if there exists a class B in the taxonomy such that B is a subclass of C, where C is the direct superclass of A.
- Some concepts had missing links. A class has a missing superclass 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 encountered 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}

1. Some concepts had very large numbers 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.

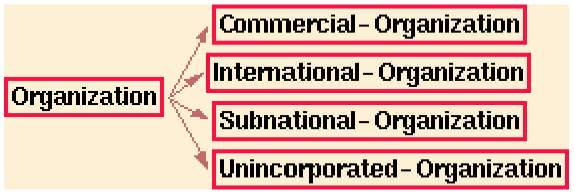


Figure 4. Snippet from the taxonomy representing organizations showing orthogonal categorizations

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 is a categorization based on extent of operations. Mixing such orthogonal categorizations adds to the complexity of the taxonomy.

2. If two classes are disjoint, the disjoint-ness relationship must be declared.
3. 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. Problems 2 and 3 suggest that there are several concepts in the taxonomy that can be systematically defined in terms of the other. One could conceivably use a simple classifier, a classifier that just uses the knowledge about the slots that a frame can have, to compute the taxonomic relationships. If we define away as many concepts as we can, we are left with a much smaller number of concepts. 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.

The process of creating a taxonomy of primitive concepts can be further simplified by following an additional principle: the concept names in a taxonomy should not be equated with their meanings in natural language. For example, in the HPKB-UL, the concept *country* is a subclass of *geographical-region* and *agent.* If country acting as an agent and the geographical region of a country are represented as two separate concepts, those concepts will have only one superclass giving a much simpler taxonomy. Therefore, we argue that while creating a taxonomy of primitives, one should represent each word sense or inferential property by using a separate concept. This gives much simpler taxonomies that are easier to manage and develop, and are more correct than otherwise possible.

Assume that we follow the two principles for taxonomy design as suggested above. We still need to define correspondences between natural language words and the concepts in our taxonomy, and the defined concepts are needed for ease of reference and navigation.

Both of these problems can be solved using viewpoints as a mechanism for organizing the taxonomies. Let us explain this in more detail.

We introduce a new kind of object in the KB called a *component.* Some of the components may be concepts in a taxonomy. For example, the natural language word *country* will be a component with two viewpoints: agent and a geographical region. The complete definition of *country* is a composition of the two viewpoints. This component could import axioms from the concepts *agent* and *geographical region* but not by using inheritance.

Composing several other concepts may create some of the components. Suppose we have a concept category, which has three subconcepts—diplomatic, military, and economic. One can define a viewpoint on "action" based on *category* to automatically obtain classes *diplomatic-action*, *military-action*, and *economic-action*. We do not necessarily have to keep these viewpoints in the taxonomy. This considerably reduces the size of the taxonomy, as well as the manual effort required to insert new concepts into the taxonomy. For example, the taxonomy of organizations shown above could be viewed as shown in Figure 4.

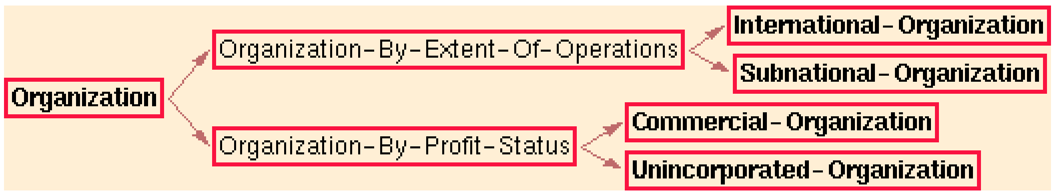

Organization
Organization-By-Extent-Of-Operations
International-Organization
Subnational-Organization
Organization-By-Profit-Status
Commercial-Organization
Unincorporated-Organization

Figure 5. Organization taxonomy using viewpoints

In the figure, *Organization-by-extent-of-operation* and *Organization-by-profit-status* are two viewpoints respectively created using slots representing the *extent* of operations and the *profit status* of an organization. In this example, such viewpoints enable us to meet the navigational-aid objective of a taxonomy.

We hope to apply these taxonomy design principles to a large taxonomy and experimentally evaluate their effectiveness in future work.

KB Modularization

KB modularization means dividing the content of a KB into conceptual partitions that serve as the basis for KB development and inference. During this project, 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 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. The modularization may be significantly influenced by the design of core theories in a KB. We realized the KB modularization by simply placing the axioms from different modules in separate files. It is quite easy to improve upon the implementation approach.

Modularization also can be used as a representation tool, for example, for representing conflicting sets of axioms. In the knowledge-representation literature, such modules have been represented. Context-like modularization, even though important, was not the focus of our work in this project. We hope to investigate these issues in future work.

Compositionality

The HPKB evaluation team introduced the notion of *compositionality*. Informally, a representation is said to be compositional if it atomically represents each individual concept in the domain of discourse, and the representation of a composition of concepts can be obtained by composing the representations of individual concepts. To illustrate this, consider the following example from the PQ grammar.

conduct {peacekeeping, counter-terrorism, search and rescue, combat support, evacuation, patrol, humanitarian} mission.

In this example, we can use two alternative representations. The first representation contains classes *conduct*, *mission* and, *mission-types*. The class *mission-types* has instances *peacekeeping*, *counter-terrorism*, and so forth. With the class *conduct*, we associate a slot *has-thing-conducted* whose values are instances of class *mission*. With the class *mission*, we associate a slot *has-mission-type* whose values can *be peacekeeping*, *counter-terrorism*, and so forth. With this representation, an instance of "conduct peacekeeping mission" will be represented by

```
(and
  (instance-of ?x conduct)
  (instance-of ?y mission)
  (has-thing-conducted ?x ?y)
  (has-mission-type ?y peacekeeping))                                    (R1)
```

One could be even stricter and argue that peacekeeping should be broken into peace and keeping, but we will ignore that for the moment.

An alternative representation would be to create just one class*: conduct-peacekeeping-mission*. An instance of "conduct peacekeeping mission" is then represented by

```
(instance-of  ?x conduct-peacekeeping-mission)                           (R2)
```

The two representations are not mutually exclusive. The expression R1 can be viewed as a definition of the class *conduct-peacekeeping-mission*. The question, however, is if a representation contains only the class *conduct-peacekeeping-mission,* is it qualitatively inferior to the one that represents things more atomically?

Schrag has proposed the following compositionality hypothesis: Non-compositional 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 second representation is inferior. In the current example, there is no strong basis for the proposed criticism of the second representation. To generalize the second representation one would simply add additional terms to the KB and give a more complete definition to it. Thus, even if the second representation is non-compositional, it is amenable to generalization if an application requires it.

Another way to compare the two representations is to say that in the second representation, *conduct-peacekeeping-mission* is a *primitive* concept, whereas in the first representation, it is a *defined* concept. A representation is compositional if every non-atomic concept is a defined concept.

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 second representation is adequate. One can certainly argue that the second representation is less reusable. That depends on the next application. If we use the second 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 *conduct-peacekeeping-mission*. This may be studied more formally with an analytical model as follows.

Consider a knowledge fragment from the CMCP specification. 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

Representation 1: *c*n1*
Representation 2: *c*n2*

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

Representation 1: *c*n3*
Representation 2: *c*n4*

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.

If one reviews compositionality with the viewpoint of taxonomy design, one can claim that it is easier to construct taxonomies if one uses compositional representations. That is because compositionally constructed concepts can be automatically placed in the taxonomy. If that hypothesis were true, then it is not really necessary to amortize the extra cost of constructing a compositional representation over multiple applications. Even if we were to create a representation for one application, it could be cheaper to use a compositional representation because it can make it easier to produce well-designed taxonomies. A more systematic investigation of compositionality remains open for future work.

KNOWLEDGE BASE CONTENT

The geo-political domain requires representing knowledge about time, space, interests, actions, agents, capabilities, threats, escalation, benefits, and risks. The knowledge about time and space is general and cuts across many different domains. The knowledge about agents, actions, capabilities, and so forth is a mixture of domain-independent and domain-specific knowledge. Designing representation for each of these cases required us to identify the classes, relations, and functions needed to represent the basic notions in the domain. The classes had to be organized into a taxonomy for efficient representation of axioms. We also needed to design efficient reasoning procedures for many cases.

Representing Temporal Knowledge

The representation of time was pervasive in the CMCP domain. In many cases, we needed representation of calendar dates, whereas in other cases, we needed an ability to reason with incomplete temporal knowledge. For example, the question, "Has Iraq launched ballistic missiles since the Persian Gulf War?", requires reasoning with calendar dates as well as reasoning with incomplete temporal knowledge. The answer to this question uses the fact that Iraq has been prohibited from possessing ballistic missiles since the Persian Gulf War, which is represented using an interval that has a starting point but no end point. Let us explain in more detail our approach for developing temporal representation.

Choosing Primitives for Representing Temporal Knowledge

Extensive research results on temporal representations are available and, therefore, our goal in choosing primitives to encode temporal relationships was to reuse an existing representation instead of inventing a new one. We reviewed the temporal representations available in the HPKB knowledge servers. The Ontolingua server has an ontology called "Simple Time". The Simple Time ontology extends Allen's representation by defining primitives for points, and defining dates and calendar months. The HPKB Upper Ontology (HPKB-UL) has an extensive collection of temporal primitives. Many of the primitives in the HPKB-UL, such as starts-after-starting-of, are a combination of Allen's primitives. It is useful to have them in the vocabulary, because they are used often in practice. The HPKB-UL also includes primitives for discontinuous intervals. Both HPKB-UL and Simple Time include primitives for representing points and intervals.

A conceptual view of the temporal representation used by us is shown in Figure 6. The core of our representation is based on Allen's intervals. We chose primitives from the HPKB-UL to create intervals that represent calendar dates. We also support those primitives from the HPKB-UL that can be translated in terms of Allen's primitives. To limit the scope of work during the current project, we decided not to support discontinuous intervals, which we left for future work.

In the CMCP domain, it was common to use "Scenario Day" as a unit of time, for example, "On Day 35, the United Nations imposed sanctions against Iraq." Scenario days are like calendar days except that the reference point for their measurement is not related to a calendar. The HPKB-UL represents a calendar day, such as, January 1, 2000, by *(day-fn 1 (month-fn january (year-fn 2000)))*. Using a similar representation approach, we introduced a relation *scenario-day-generic* that takes an integer and reference point as arguments. For example, "Day 35 of the HPKB scenario" can be represented as *(scenario-day-generic 35 hpkb-scenario).*

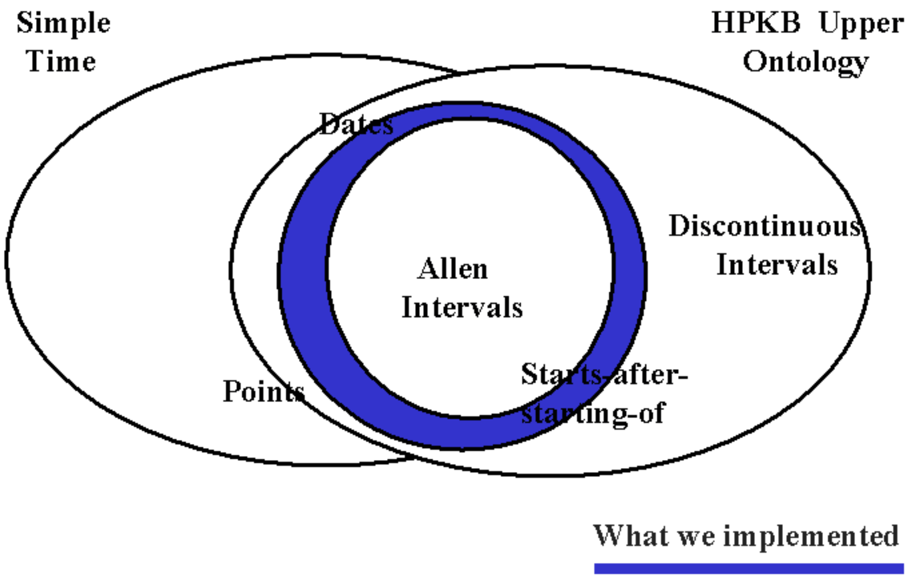


Figure 5. Conceptual view of the temporal representation

Let us now illustrate the application of this representation to the PQ grammar. A portion of the PQ grammar concerning time is

```
<TimeSpec> = {<TemporalQualifier> <TimeInterval>}

<TemporalQualifier> =
      {during, throughout,
       before, after,
       {starting, ending} at the {end, beginning} of,
       a short time {before, after}}
```

The variable <TemporalQualifier> in the PQ grammar is used between two time intervals. The first column in Table 2 shows the text that will be generated from the PQ grammar, assuming A and B are instantiations of <TimeInterval>. The second column shows the corresponding representation using the primitives from the temporal representations. The primitive *starts-after-starting-of* is a disjunction of several of Allen's primitives (during, starts, starts-inverse, finishes, meets inverse, and overlaps inverse) and proved to be useful shorthand for practical use. The function *stif* is a primitive from the HPKB-UL representing one short time interval following another.

Table 2. Representation of text involving temporal knowledge

Text from the PQ Grammar	Representation
A before B	(*starts-after-ending-of* B A)
A after B	(*starts-after-ending-of* A B)
A during B	(*temporally-subsumes* B A)
A throughout B	(*temporally-subsumes* A B)
A starts at the end of B	(*contiguous-after* A B)
A starts at the beginning of B	(*temporally-coorginating* A B)
A ending at the end of B	(*temporally-coterminal* A B)
A ending at the beginning of B	(*contiguous-after* B A)
a short time before A	(*stib* A)
a short time after A	(*stif* A)

Associating Time with Assertions

After choosing an adequate collection of temporal primitives, the next step was to decide how to associate a temporal extent with assertions in a KB. For example, consider the assertion "John possesses a house in 1995". Using the relation *possesses*, we can formalize "John possesses a house" as *(possesses John House-1).* To associate temporal extent with an assertion, the HPKB-UL provides at least two solutions: *holds* predicate or *temporal subabstractions.* Using the *holds* predicate, we may write

(holds (possesses John House-1) (Year-fn 1995))

Using the *holds* predicate makes the representation second order because *(possesses John House-1)* is not a term. Since SNARK is a first-order theorem prover, we did not want to directly adopt this solution. Instead of the *holds* predicate, one may define temporal subabstractions *John-in-1995* and *House-1-in-1995,* and assert that

(possesses John-in-1995 House-1-in-1995)

If we use temporal subabstractions, the representation remains first order, but there is an extra overhead of defining and implementing them.

A solution commonly used in the literature is to add an extra time argument to a predicate wherever necessary. For example, one could write

(possesses John House-1 (Year-fn 1995))

By adding an extra argument to *possesses*, the representation remains first order and has well-understood semantics. The predicate *possesses* satisfies the property

```
(forall (?time-interval)
        (implies
          (and
            (possesses John House-1 (Year-fn 1995))
            (temporally-subsumes (Year-fn 1995) ?time-interval))
        (possesses John House-1 ?time-interval)))
```

The gain in clarity and first-order representation by adding the time argument to predicates is at the cost of some inconvenience while entering knowledge. It forces a KE to always specify the temporal extent of an assertion even when it is not of interest. Furthermore, adding an extra argument to *possesses* makes its definition different from its corresponding definition in the HPKB-UL. Therefore, we cannot directly use the predicate definitions from HPKB-UL for a predicate such as *possesses*. We need to *reformulate* their definitions to include a time argument. Such reformulation is natural and expected when a KB developed by one person is *reused* by someone else.

Another issue that arises while associating time with assertions is to specify whether if an assertion holds true over an interval, then it holds true for each of its subintervals. Put another way, that something is true of an interval does not automatically entail that it is true of all the subintervals. (This should not be surprising, since it also does not hold, in general, for other kinds of parts—for example, the parts of an automobile are not themselves automobiles.) If a relation is 'inherited' by subintervals (for example, being alive) then this needs to be stated explicitly. This is implemented in SNARK by specifying a keyword argument while defining functions.

Efficient Reasoning with Time

Having decided on how to associate a temporal extent with assertions, the final step was to realize an implementation to support the chosen representation. In our initial work, we encoded several axioms representing the temporal knowledge and used *resolution* on them to derive necessary conclusions. A resolution-based approach to temporal reasoning is inherently slow, when compared to using temporal reasoners that are available for temporal reasoning. Therefore, we undertook the task of interfacing our theorem prover SNARK with a temporal reasoner.

We needed to decide whether to use a point-based or an interval-based reasoner for implementation. From a purely representation viewpoint, the two are interchangeable. A point-based representation is simpler than an interval-based representation because only three relationships between two points are possible (before, after, or equal) compared to thirteen relationships between two intervals (as in Allen's interval algebra). When it comes to choosing a temporal reasoner, an interval-based representation for time is more expressive than a point-based representation for stating disjoint-ness relationships. To state disjoint-ness using a point-based representation, one has to use disjunction of conjunctions of temporal relationships, which is not supported by any of the existing point-based reasoners. Disjoint-ness of temporal relationships is a primitive relationship in an interval-based representation and is naturally supported in Allen-style reasoners. We decided to use an interval-based representation because of its greater expressiveness.

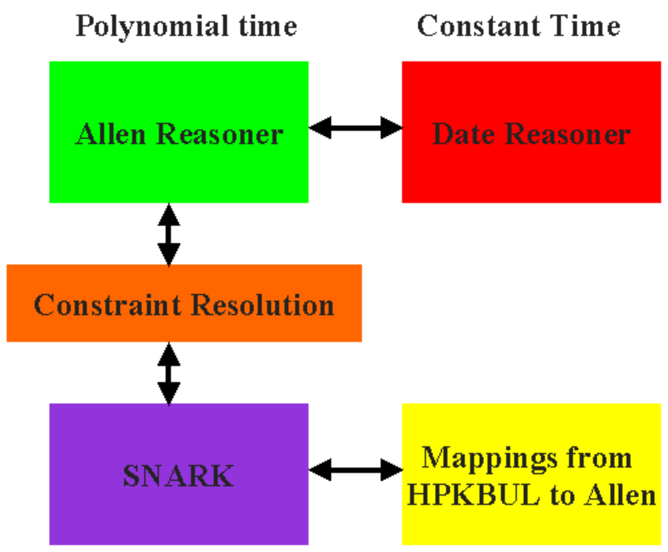


Figure 6. Implementation architecture for the temporal reasoner

The implementation architecture is shown in Figure 6. We implemented Allen's algorithm for reasoning with temporal intervals, which requires polynomial time and space for local consistency checking. For intervals representing absolute dates, it is possible to do computations in constant time. Therefore, we extended Allen's reasoner to invoke a constant time date reasoner whenever comparison between absolute dates is to be done. The date reasoner also supports comparisons between intervals that are specified as an absolute distance from a reference point, for example, *(scenario-days 3 5).* Since our representation includes primitives from the HPKB-UL that are not directly Allen's primitives, but can be translated to them, our implementation includes translations for such primitives.

The interface between SNARK and Allen's reasoner uses *constraint resolution*. Each formula containing a predicate with time argument is broken into two parts: a temporal constraint and a formula that is true whenever the temporal constraint is satisfied. For example, the formula

(possesses John House-1 (year-fn 1995))

is broken into a pure formula

(possesses John House-1 ?time-interval)

and a temporal constraint

(temporally-subsumes (year-fn 1995) ?time-interval)

The pure formula is evaluated using the ordinary resolution and the temporal constraint using Allen’s reasoner. A resolution step succeeds if the resolution on the pure formula succeeds and the evaluation of the temporal constraint succeeds.

Our temporal reasoner supersedes numerous axioms from our Y1 KB that were used to support temporal reasoning using resolution. Consequently, the number of axioms in the KB went down. It is, however, possible to establish an equivalence between the temporal reasoner and the axioms that would be needed to support the same reasoning. Developing a temporal reasoner is slower than actually writing the equivalent axioms. Developing reasoners does not necessarily lead to speedup in KB construction time. A temporal reasoner, however, is expected to speed up the inference. Thus, speeding up inference and speeding up KB construction time are often conflicting goals.

Our choice of adding a time argument to every relation is yet to be tested in practice. It is possible that adding a time argument might increase the inference time. Furthermore, one needs to go through the whole KB and identify the relations for which the time argument should be added. We hope to undertake that exercise in our future work.

Representing Knowledge about Geographical Directions

The fragment of the PQ grammar corresponding to the directional information is

```
<DirectionalPart> =
      {eastern, western, northern, southern, central,
       {north, south, east, west}<DirectionalPart>}
```

The HPKB-UL has a extensive collection of spatial predicates to represent relations such as "object A covers object B" or "object A is inside object B". It distinguishes between different forms of covering and containment. It also defines several predicates to define paths, for example, primitives to represent the end points of a path and subpaths of a path. For our CMCP work, we made use of the primitives to represent paths and geographic proximity that were already available in the HPKB-UL. To represent directional knowledge, we needed primitives that were not defined in the HPKB-UL.

We defined functions to represent each direction: *eastern, northern, southern, western, central, northeast, northwest, southeast,* and *southwest.* Using these functions, Northern Iran could be represented as *(northern iran),* assuming *iran* is a constant representing the geographical region of Iran. The composite directions such as northwest could be constructed by successive function applications of primitive direction functions. For example, northwest Iran could be represented as *(northern (western Iran))).* The PQ grammar allows the possibility of directions such as "north south", "south south", but we did not represent such cases as they are not very common. Even though the phrase "north south pipeline" was used in the domain, the sense in which "north south" is being used does not correspond to the "north south" of a region, and thus requires separate representation.

We defined the relationship of the directional primitives to the relation *geograpical-subregion-of* that already existed in the HPKB-UL. These axioms are

```
(forall ((?place place)) (geographical-subregion-of (southern ?place) ?place)))
(forall ((?place place)) (geographical-subregion-of (eastern ?place) ?place)))
(forall ((?place place)) (geographical-subregion-of (northern ?place) ?place)))
(forall ((?place place)) (geographical-subregion-of (western ?place) ?place)))
(forall ((?place place)) (geographical-subregion-of (north-east ?place) ?place)))
(forall ((?place place)) (geographical-subregion-of (north-west ?place) ?place)))
(forall ((?place place)) (geographical-subregion-of (south-east ?place) ?place)))
(forall ((?place place)) (geographical-subregion-of (south-west ?place) ?place)))
```

The definition of directional relations in our KB was adequate for the CMCP work but far from complete. We hope to extend these definitions in our future work.

Taxonomy of Interests

A significant portion of the PQ grammar involves representing interests. Ontologically, interests are similar to goals. In the HPKB-UL, goals are represented as a subclass of *mental-object*. A goal is something that an agent is interested in and is actively trying to achieve. Therefore, goal is a special kind of interest. In our ontology, we defined interest as a subclass of *mental-object* and as a superclass of *goal*.

In logic, goals and interests are sentences and are normally represented using a meta knowledge operator. For example, Iran's goal of achieving OPEC leadership may be represented as *(goals Iran ^(leads Iran OPEC)),* where *(leads Iran OPEC)* is a sentence and ^ is a *meta* knowledge operator. Our representation of goals during the first year of the project used the *meta* knowledge operator for representing goals. The use of the *meta* knowledge operator has two drawbacks: it is difficult to explain to non-AI experts, and requires a special reasoning method to reason with terms within the scope of a *meta* knowledge operator. To deal with these problems, we designed an alternative representation for goals. We now represent each goal by a frame and associate a slot with it to specify its properties. For example, the goal *(leads Iran OPEC)* can be represented by a frame whose name may be *leadership-goal,* which may have two slots—*has-agent* and *has-leader-of* with values *Iran* and *OPEC,* respectively. One can then use *(goals Iran leadership-goal)* to represent Iran's goal of achieving OPEC leadership. This representation is easier to explain to relatively unskilled users, and does not require any special reasoning support. It is, however, open to the objection that it allows substitution of equals by equals for terms that appeared inside the scope of the meta knowledge operator earlier. We dismiss this objection by defining *goals* in a way that substitution of equals by equals is allowed.

Many interests—for example, national—are naturally represented by a class. For other interests, such as gross domestic product, the representation is not obvious, because the ontology already contains a class representing it. In such cases, an alternative representation is to create a new class representing interest in gross domestic product and define it as a subclass of a class of interests. Another alternative is to overload the class representing gross domestic product to also represent interest in gross domestic product. The former approach requires us to introduce an additional class in the KB, whereas under the latter approach, all the slots associated with an interest get associated with gross domestic product. We opted for the former approach to maintain the semantic distinction between an object and an interest in that object.

The PQ grammar had two other concepts similar to an interest. These concepts include intention and motivation. We defined motivation as a synonym of interest, and intention as a subclass of goal. This interrelationship is shown in Figure 8. For the solution of the CMCP, these distinctions did not prove to be relevant.

While developing the interest ontology, we made extensive use of the International System Framework (ISF) document (Jermano and Picarelli 1998) that was supplied to us by IET. Our initial design of the KB contained those classes of interest that were identified in the ISF document. Additions to the initial design were made based on the requirements of the PQ grammar.

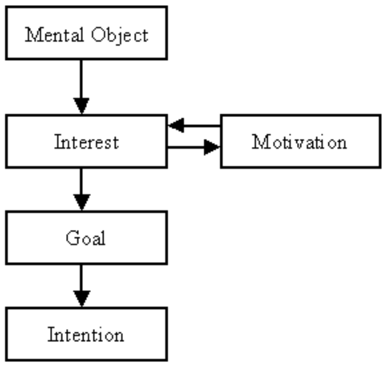


Figure 7. Interrelationship among mental objects

A snapshot of the interest ontology is shown in Figure 8. The national interests had four broad categories: economic, national security, ideological, and status related. As stated earlier, this categorization was derived from the ISF document, and then refined to take into account the requirements of the PQ grammar. Even though the concepts have long hyphenated names, the concept definitions included component slots. For example, the interest *deterring-global-threats* has a slot *has-deters-against* whose value specifies the object against which the deterrence is desired. In other cases, concept names were created without giving detailed definitions, because the details were needed for the CMCP solution.

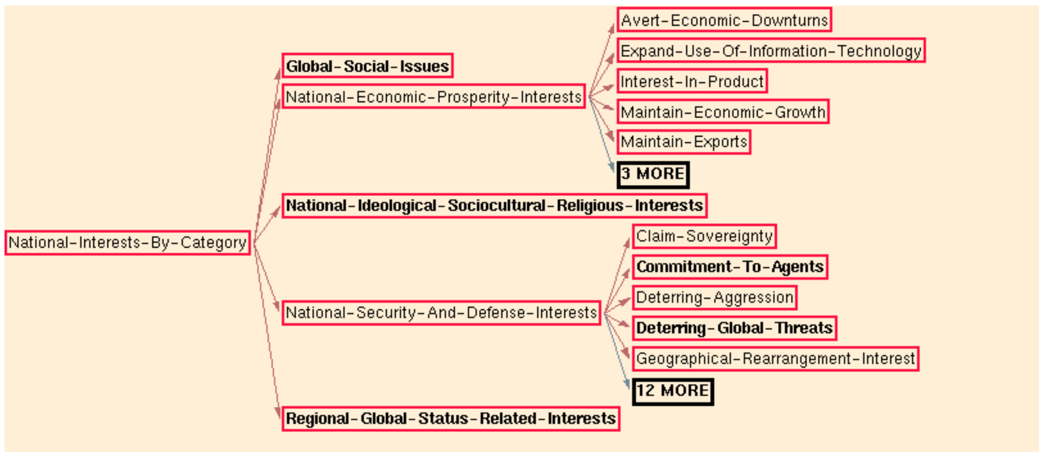

National-Interests-By-Category
Global-Social-Issues
National-Economic-Prosperity-Interests
National-Ideological-Sociocultural-Religious-Interests
National-Security-And-Defense-Interests
Regional-Global-Status-Related-Interests
Avert-Economic-Downturns
Expand-Use-Of-Information-Technology
Interest-In-Product
Maintain-Economic-Growth
Maintain-Exports
3 MORE
Claim-Sovereignty
Commitment-To-Agents
Deterring-Aggression
Deterring-Global-Threats
Geographical-Rearrangement-Interest
12 MORE

Figure 8. Snapshot from the interests ontology

Taxonomy of Actions

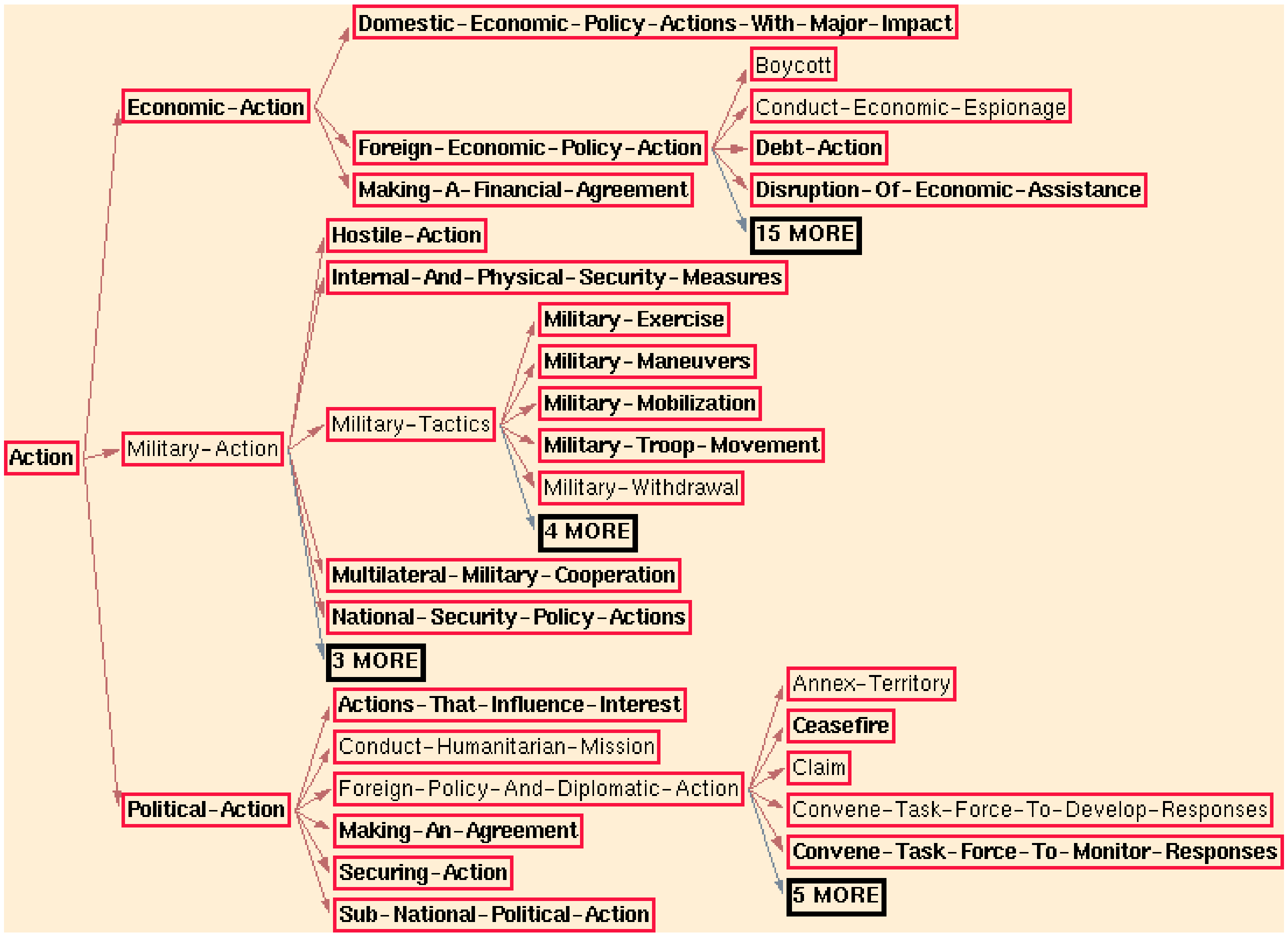

Action
Economic-Action
Domestic-Economic-Policy-Actions-With-Major-Impact
Foreign-Economic-Policy-Action
Boycott
Conduct-Economic-Espionage
Debt-Action
Disruption-Of-Economic-Assistance
15 MORE
Making-A-Financial-Agreement
Military-Action
Hostile-Action
Internal-And-Physical-Security-Measures
Military-Tactics
Military-Exercise
Military-Maneuvers
Military-Mobilization
Military-Troop-Movement
Military-Withdrawal
4 MORE
Multilateral-Military-Cooperation
National-Security-Policy-Actions
3 MORE
Political-Action
Actions-That-Influence-Interest
Conduct-Humanitarian-Mission
Foreign-Policy-And-Diplomatic-Action
Annex-Territory
Ceasefire
Claim
Convene-Task-Force-To-Develop-Responses
Convene-Task-Force-To-Monitor-Responses
5 MORE
Making-An-Agreement
Securing-Action
Sub-National-Political-Action

Figure 9. Snippet of the taxonomy representing actions

The actions for the geo-political domain were specified in the ISF document. We used the ISF document and the PQ grammar to define the scope of different actions that needed to be represented. The document proved to be useful in identifying basic categories of actions. A snippet of the taxonomy of actions is shown in Figure 9. The top three categorizations shown in this figure were identified based on the ISF document. Various subcategorizations of actions were also derived from the document.

Many long constant names—for example, *convene-task-force-to-develop-responses*—were based on the action descriptions from the ISF document. In those cases where the detailed definitions were required by the PQ grammar, slots were defined to capture the detail. For example, to support the PQ grammar, the class *military-withdrawal* had a slot called *has-withdrawal-from* to indicate the geographical location from which the military is withdrawn.

The taxonomies of actions and interests served the engineering needs of the project, but clearly motivated the need for a principled approach to developing taxonomies. These principles were discussed earlier in the section on taxonomy design. Orthogonal to the design of the taxonomy is the decision on how to choose the primitive concepts in a taxonomy. There is substantial work on libraries of verbs (Levin 1993) that can be used as an inspiration for choosing primitive action concepts. We hope to investigate this in our future work.

Action-Interest Interaction

Many of the CMCP questions involved inference with interests and actions, for example, the interests that might underlie an action or the actions an agent can take to achieve an interest.
The inference for action-interest interaction centered around two qualitative influence relations: *may-positively-influence* and *may-negatively-influence.* We *say* that an action *A* may *positively* influence an action *B*, if by performing an action *A*, an agent could likely strengthen its interest *B.* For example, by imposing sanctions against a country, an agent could strengthen its interest in upholding international law. This relationship is purely qualitative, and does not attempt to capture the magnitude of the qualitative influence. Also, it does not imply that after performing the action *A*, the interest *B* is fully achieved. We *say* that an action *A* may *negatively* influence an interest *B*, if by performing an action *A*, an agent could likely weaken its interest *B.* For example, by supporting terrorist groups in a region, an agent could weaken its interest in maintaining regional stability.

Interests Underlying an Action

A subset of the PQ236 was

What type of interest typically underlies an <InternationalAgentType>'s decision to <InternationalActionType>?

To answer this question, we needed to formalize the notion of "typically underlie". This can be done as follows.

```
(forall ((?action action) (?interest interest) (?agent agent))
         (=>
          (and
           (may-positively-influence ?action ?interest)
           (has-mental-object-of ?interest ?agent)
           (performed-by ?action ?agent))
          (typically-underlies ?interest ?action)))
```

Actions Leading to an Interest

PQ237. What types of international action does a desire toward <InterestEffectType> <InterestType> typically lead to?

To answer this question, we needed to formalize the notion of "typically lead to". This can be done as follows.

```
(forall ((?action action)(?interest interest))
 (=>
  (may-positively-influence ?action ?interest)
  (typically-leads-to ?interest ?action)))
```

Example values of <InterestEffectType> were

```
<InterestEffectType> =
        {{increase, decrease} in,
         {strengthening, weakening, containment, maintenance,
           development, improvement, degradation} of}
```

In our ontology, we had a class of effects, with subclasses representing different effects, such as increase and decrease. Many of these effects could also be reduced to the basic model of positive and negative influences as follows.

If an action may negatively influence an interest, then that action has a weakening effect on that interest.

```
(forall ((?action action) (?interest interest))
 (=>
    (may-negatively-influence ?action ?interest)
   (has-typical-action-effect ?action ?interest weakening-effect)))
```

If an action may negatively influence an interest, then that action has a decreasing effect on that interest.

```
(forall ((?action action) (?interest interest))
 (=>
  (may-negatively-influence ?action ?interest)
  (has-typical-action-effect ?action ?interest DECREASING-EFFECT)))
```

If an action may negatively influence an interest, then that action has a containing effect on that interest.

```
(forall ((?action action) (?interest interest))
(=>
  (may-negatively-influence ?action ?interest)
  (has-typical-action-effect ?action ?interest CONTAINMENT-EFFECT)))
```

If an action may negatively influence an interest, then that action has a DEGRADATION effect on that interest.

```
(forall ((?action action) (?interest interest))
 (=>
  (may-negatively-influence ?action ?interest)
  (has-typical-action-effect ?action ?interest DEGRADATION-EFFECT)))
```

If an action may negatively influence an interest, then that action has a damaging effect on that interest.

```
(forall ((?action action) (?interest interest))
 (=>
  (may-negatively-influence ?action ?interest)
  (has-typical-action-effect ?action ?interest DAMAGING-EFFECT)))
```

If an action may positively influence an interest, then that action has an increasing effect on that interest.

```
(forall ((?action action) (?interest interest))
(=>
  (may-positively-influence ?action ?interest)
  (has-typical-action-effect ?action ?interest increase-effect)))
```

If an action may positively influence an interest, then that action has a strengthening effect on that interest.

```
(forall ((?action action) (?interest interest))
 (=>
  (may-positively-influence ?action ?interest)
  (has-typical-action-effect ?action ?interest strengthening-effect)))
```

If an action may positively influence an interest, then that action has a maintaining effect on that interest.

```
(forall ((?action action) (?interest interest))
 (=>
  (may-positively-influence ?action ?interest)
  (has-typical-action-effect ?action ?interest MAINTENANCE-EFFECT)))
```

If an action may positively influence an interest, then that action has a developing effect on that interest.

```
(forall ((?action action) (?interest interest))
 (=>
  (may-positively-influence ?action ?interest)
  (has-typical-action-effect ?action ?interest DEVELOPMENT-EFFECT)))
```

If an action may positively influence an interest, then that action has an improving effect on that interest.

```
(forall ((?action action) (?interest interest))
 (=>
  (may-positively-influence ?action ?interest)
  (has-typical-action-effect ?action ?interest IMPROVEMENT-EFFECT)))
```

Competing and Agreeing Interests

PQ239. In what ways do [interests in] <InterestType1> typically {compete, agree} with interests in <InterestType2> for an <InternationalAgentType>?

Two interests agree if they are of the same kind or achieving one of them can also lead to achieving the other. The interests compete if they conflict in the sense that achieving one of them makes it difficult to achieve another.

Here is an attempt to formalize these notions:

If two actions of the same type positively influence two interests, then those interests agree with respect to that type of action.

```
 (forall ((?action action) (?action1 action) (?interest interest) (?interest1 interest))
           (=>
            (and
             (may-positively-influence ?action ?interest)
             (may-positively-influence ?action1 ?interest1)
             (same-instance-type ?action ?action1)
             )
            (and
             (agrees-with  ?interest ?interest1 ?action)
             (agrees-with  ?interest1 ?interest ?action)
             )))
```

If there are two actions of the same type such that one of them negatively influences an interest and the other positively influences the same interest, then those interests compete regarding that type of action.

```
 (forall ((?action action) (?action1 action)
                                  (?interest interest) (?interest1 interest))
           (=>
            (and
             (may-positively-influence ?action ?interest)
             (may-negatively-influence ?action1 ?interest1)
             (same-instance-type ?action ?action1)
             )
            (and
             (competes-with  ?interest ?interest1 ?action)
             (competes-with  ?interest1 ?interest ?action)
             )))
```

These axioms turned out to be quite problematic in obtaining the correct inferences. The SME feedback on these axioms was that just because an action can be taken to further more than one type of interest does not mean that those interests have anything in common. For example, economic sanctions are a typical kind of measure used in support of a wide variety of foreign policy, national security, and trade goals. One cannot argue that all of these goals “agree” with each other—many, in fact, are competing. Because of these problems, we tried the following axiom.

```
(forall ((?interest1 interest) (?interest2 interest))
 (=>
  (= ?x (nca ?interest1 ?interest2))
  (agrees-with ?interest1 ?interest2 ?x)))
```

The function *nca* computes the nearest common ancestor of two interests in the taxonomy of interests. If the two interests are the sucessors of the same interest, they are likely to agree. The inferences produced using this axiom met with greater approval of the SMEs even though this axiom is as flawed as the earlier ones: just because two interests are the successors of the same interest does not mean that they agree—in fact, they could be competing.

Action Responses

PQ219. <ContextSpec>, what actions might <InternationalAgent1> take [with respect to <InternationalAgent2>] [in response to <EventSpec>]?

Most of our reasoning with interests and actions has been on the premise that the agents perform actions to further one or more of their interests. The PQ219 makes the reasoning more specific by setting a specific context and by giving a specific event that needs to be responded to. We can apply our qualitative model of influences by first identifying the interests of an agent in a situation, then checking which of those are negatively influenced by the given action, and then identifying how those negative influences could be positively influenced. This intuition is captured in the following axiom.

If ?action1 performed-by ?agent2 negatively influences ?interest1 of ?agent1, and ?action2 positively influences ?interest1, then ?action2 is a viable response to ?action1 with respect to ?agent2.

```
(forall ((?action1 action)  (?interest1 interest) (?action2 action)  (?agent1 agent)  (?agent2
agent))
          (=>
           (and
            (may-negatively-influence ?action1 ?interest1)
            (has-mental-object-of ?interest1 ?agent1)
            (may-positively-influence ?action2 ?interest1)
            (not (equal ?action1 ?action2))
            (performed-by ?action1 ?agent2))
           (and
            (may-respond-to-action ?agent1 ?action1 ?action2)
            (starts-after-starting-of ?action2 ?action1)
            (has-with-respect-to-agent ?action2 ?agent2))))
```

Action Options

PQ220. What [[non-]{diplomatic, military, political, violent}] actions can <InternationalAgent1> take to foster an <InterestEffectType> <InterestType> [of <InternationalAgent2>]?

This question asked for possible options available to an agent to achieve an interest or an effect on interest. Even though there can be many possible variations of <InterestType> and <InterestEffectType>, the source material was available to support only the instantiation "strengthen relationship". This instantiation was supported using a knowledge fragment that was formalized as follows.

A country will use positive incentives to strengthen relations with other countries.

```
(forall
 ((?interest-in-foreign-policy-between-agents interest-in-foreign-policy-between-agents))
 (and
   (has-foreign-policy-between ?interest-in-foreign-policy-between-agents ?country)
   (has-foreign-policy-between ?interest-in-foreign-policy-between-agents ?country1)
   (has-interaction-type  ?interest-in-foreign-policy-between-agents ?relations)
   (has-mental-object-of ?interest-in-foreign-policy-between-agents ?country)
   (has-mental-object-of ?interest-in-foreign-policy-between-agents ?country1)
   (not (equal ?country ?country1))
   (has-action-polarity ?action positive))
 (and
   (performed-by ?action ?country)
   (supporting ?action ?country1)
   (may-positively-influence ?action ?interest-in-foreign-policy-between-agents)))
```

This axiom, however, required us to associate a polarity with actions to indicate whether they are considered generally positive. This was done using axioms of the form

```
(exists
  ((?trade-and-aid trade-and-aid))
     (has-action-polarity ?trade-and-aid positive))
```

Finally, we used the axiom to define action options in terms of positive influences.

A possible option to achieve an interest is to do an action that will positively influence it.

```
(forall ((?agent agent) (?action action) (?interest interest))
 (=>
  (and (may-positively-influence ?action ?interest)
       (has-mental-object-of ?interest ?agent)))
 (agent-action-options ?agent ?action ?interest))
```

Demand/Supply Actions

Two variations of PQ237 required lengthier chains of inference than the common cases and, therefore, are worthy of separate discussion. These questions were

What types of international action does a desire toward decrease in price of oil typically lead to?

What types of international action does a desire toward increase in price of oil typically lead to?

An interest in an effect is positively influenced by an action that can achieve that effect.

The basis for the solution is the inverse relationship between production and price. We represented this as follows.

According to the laws of supply and demand (other things being unchanged): (1) if the supply of a product increases, the price of that product will decrease, and (2) if the supply of a product decreases, the price of that product will increase. (Source: Knowledge Fragment A8)

```
(forall
 ((?product product) (?agent agent)
  (?economic-measure-of-product economic-measure-of-product)
  (?increase-effect increase-effect) (?money money) (?action))
 (=>
  (and (has-product-measure ?product ?economic-measure-of-product)
       (has-production-capacity ?agent ?economic-measure-of-product)
       (has-price ?product ?money)
       (has-action-effect ?action ?economic-measure-of-product ?increase-effect))
  (forall ((?decreasing-effect decreasing-effect))
   (and (has-action-effect ?action ?money ?decreasing-effect)))))

(forall ((?money money) (?action))
 (=>
  (forall
   ((?agent agent)
    (?economic-measure-of-product economic-measure-of-product)
    (?decreasing-effect decreasing-effect) (?product product))
   (and (has-product-measure ?product ?economic-measure-of-product)
        (has-production-capacity ?agent ?economic-measure-of-product)
        (has-price ?product ?money)
        (has-action-effect ?action ?economic-measure-of-product
         ?decreasing-effect))
   (forall ((?increase-effect increase-effect))
    (and (has-action-effect ?action ?money ?increase-effect))))))
```

Having specified the relationship between production and price, we needed to link this to our qualitative model of influences so that we could use our already-existing framework for inferring interests that lead to an action. We did this by using the axiom

```
(forall
 ((?action action) (?effect effect) (?interest interest)
  (?thing thing))
 (=>
  (and (has-action-effect ?action ?thing ?effect)
       (has-mental-object-of ?interest ?agent)
       (performed-by ?action ?agent)
       (has-interest-in-thing ?interest ?effect))
  (may-positively-influence ?action ?interest)))
```

The positive influence in this context has the connotation of ‘enabling’. For example, if an agent is interested in decrease of production, and an action has the ‘negative’ influence on the production, then by decreasing it, the influence of that action on the interest is indeed positive.

Using these axioms, we could infer that to increase the price of oil an agent would typically try to decrease the production and vice versa.

Modeling Escalation

PQ210. <ContextSpec>, is <EventSpec1> an {{escalation, de-escalation} of conflict, retaliation for <EventSpec2>}?

Given a context, we needed to determine if a given event leads to escalation of a conflict, de-escalation of conflict, or is retaliation for some earlier event. The contexts include both historical incidents and fictional scenarios for CMCP. Intuitively, an action in a context is escalation if it is in response to another, less hostile action. Conversely, an action is de-escalation if it is in response to another, more hostile action. Finally, an action is retaliation for another if it is a reaction to the other and the two actions are opposed to each other or have contrary interests.

The formal representation of scenario consists of a series of event descriptions. Each event description included the casual relationship between events, and thus it was straightforward to formalize retaliation as follows.

```
(=>
(and
   (performed-by ?action1 ?agent1)
   (performed-by ?action2 ?agent2)
   (occurs-in ?action2 ?context)
   (cause-event-event ?action1 ?action2))
(retaliation ?action2 ?action1 ?context)))
```

To infer whether an event leads to escalation or de-escalation, we associated a hostility level with each action. Using the hostility level, we defined escalation and de-escalation as follows.

An ?action2 is an escalation in ?context if it is in response to an ?action1 and is of greater hostility level than ?action1.

```
(=>
  (and
     (occurs-in ?action2 ?context)
     (cause-event-event ?action1 ?action2)
     (greater-hostility ?action2 ?action1))
    (escalation ?action2 ?context))
```

A similar axiom for de-escalation is

An ?action2 is a de-escalation in ?context if it is in response to an ?action1 that has a greater hostility level.

```
(=>
 (and
  (occurs-in ?action1 ?context)
  (cause-event-event ?action1 ?action2)
  (greater-hostility ?action1 ?action2))
 (de-escalation ?action2 ?context))
```

In associating hostility levels with events. Herman Kahn (“On Escalation”) proposed one notion of hostility level based on a forty-four-stage linear scale. This idea was modified by a CMCP subject-matter expert during an interview (“Knowledge Acquisition for Crisis Management: Interests and Actions”, by John Kingston, AIAI, University of Edinburgh, http://www.ai.sri.com/hpkb/intact.html). He suggested a multidimensional scale for hostility levels, because of the difficulty in comparing actions of different kinds. Our own scale currently has three component levels: damage, weapon, and proximity. A conceptual view of the scale that we used is shown in Figure 10.

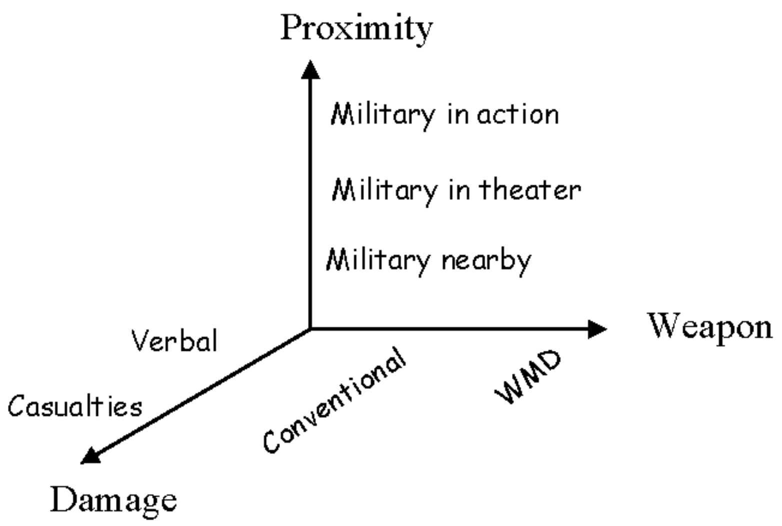


Figure 10. Hostility level defined along three orthogonal scales

The damage level is an estimate of what kind of damage the action involves. A military attack, for example, is likely to involve population damage, which is the most severe level. Public criticism of one government by another is likely to reach only the verbal damage level, which is much less severe.

The weapon level reflects the kind of weapons the attack involves. For example, an attack using biological weapons is most severe. An attack that involves no weapons has the least severity on the weapon scale.

The proximity level concerns the location of the attack. An attack on the heart of another country has the most severe proximity level, and an attack that is outside the borders of the target country has the lowest proximity level.

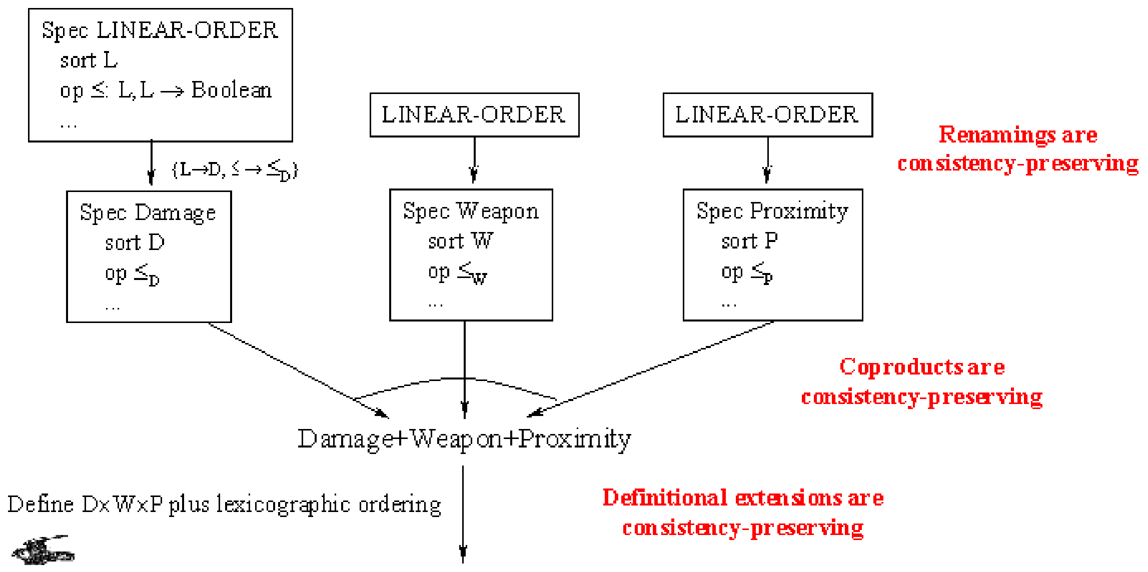


Figure 11. Compositional construction of the hostility model from the theory of linear orders

Hostility levels of two actions can be compared with a “lexicographic ordering”. More precisely, the action with the higher damage level has the higher hostility level; if the damage levels are equal, the action with the higher weapon level has the higher hostility level; if both the damage level and the weapon level are equal, the action with the higher proximity level has the higher hostility level.

Figure 11, created by Kestrel Institute, illustrates how the computation of escalation can be compositionally constructed from the theory of linear orders. Constructing each of the damage, weapon, and proximity scales amounts to renaming the sort names in the theory of linear orders. Such renaming is consistency preserving. The lexicographic ordering of the three scales is a co-product that also is consistency preserving. Using the hostility model in conjunction with the escalation axiom is a definitional extension that is consistency preserving as well. Whenever we can reduce a reasoning task to a well-understood formal theory, it greatly eases the task of ensuring its formal properties.

Representing Capabilities

Several CMCP questions required reasoning about capabilities of an agent to perform a certain task. For example,

PQ35. Can <InternationalAgent1> sponsor <InternationalActionType> inside <InternationalAgent2>'s borders?

PQ76. Does <InternationalAgent> have a <MilitaryOrganizationType> that can perform <InternationalActionType>?

Many of the instantiations of PQ35 involved the ability of an agent to sponsor terrorist attacks. Instantiations of PQ76 involved actions that were military in nature. Capability of an agent to perform military actions is mostly determined by whether it possesses the necessary resources. The capability for performing terrorist attacks is dependent on not just resources but on the ideology of a state and the past history of doing such acts. We describe some of the axioms that we used to reason about capabilities.

States that sponsor acts of terrorism generally have ideological ties to terrorist groups.

```
(forall ((?supporting-action supporting-action) (?country country)
          (?terrorist-group terrorist-group))
   (implies
     (and
      (performed-by ?supporting-action ?country)
      (object-acted-on ?supporting-action ?terrorist-group))
     (capable-of-because ?country supporting-terrorist-attack ?supporting-action))))
```

Having ideological ties to a terrorist group necessarily involves supporting it; therefore, representing "ideological ties" by "supporting actions performed in the past" is a reasonable simplification. The historical knowledge about the supporting actions performed by a country necessary for this axiom was encoded based on the terrorism fact sheets. The second argument of the relation *capable-of-because* is a class meaning that the ?country is capable of performing actions in that class (which in this case is *supporting-terrorist-attack*).

A country is capable of sponsoring terrorist attacks if it has performed terrorist attacks in the past.

```
(forall ((?terrorist-attack terrorist-attack) (?country country))
    (implies
     (performed-by ?terrorist-attack ?country)
     (capable-of-because ?country supporting-terrorist-attack ?terrorist-attack))))
```

If a country has been involved in the illegal trade of weapons, it is capable of sponsoring terrorist actions.

```
(forall ((?act illegal-export) (?country country) (?weapon weapon))
    (implies
     (and
      (performed-by ?act ?country)
      (object-of-possession-transfer ?act ?weapon))
     (capable-of-action-class ?country terrorist-attack))))
```

Let us now consider a few axioms representing capabilities for actions that are military in nature.

A country may perform an air strike if it possesses the requisite aircraft and if the target is within its range.

```
(forall ((?country country) (?aircraft-class aircraft-class) (?place place) (?time time))
    (=>
     (and
      (possesses ?country ?aircraft-class)
      (=< (in-km (distance-between ?country ?place))
            (in-km (range ?aircraft-class)))
      (object-found-in-location ?thing ?place))
   (exists ((?air-attack air-attack))
     (and
        (capable-of ?country ?air-attack)
        (performed-by ?air-attack ?country)
        (device-used ?air-attack ?aircraft-class)
        (event-occurs-at ?air-attack ?place))))
```

The application of this axiom required ground facts about the airplanes owned by a country and their ranges. The ground facts were entered into the KB based on the source material included in the CMCP specification.

A country is capable of biological warfare if and only if it possesses biological weapons.

```
(forall ((?biological-weapon biological-weapon) (?country country))
     (<=>
      (exists (?biological-weapon)
                 (possesses ?country ?biological-weapon))
      (capable-of-class ?country biological-warfare)))
```

It is common to state that a country is capable of performing an action, even though it is usually a specific group or organization that actually performs the action. The following axiom captures this intuition.

If an agent possesses an organization with a certain capability, the agent also has that capability.

```
(forall ((?agent agent) (?organization organization) (?action action))
  (=>
    (possesses ?agent ?organization)
    (=> (capable-of ?organization ?action)
          (capable-of ?agent ?action))))

(forall ((?agent agent) (?organization organization) (?action-class actiono-class))
   (=>
    (possesses ?agent ?organization)
    (=> (capable-of-class ?organization ?action-class)
          (capable-of-class ?agent ?action-class)))))
```

Representing Benefits and Risks

PQ39. [{During, After} <TimeInterval>], what {risks, rewards} would <InternationalAgent> face/expect in <InternationalActionType>?

To assess the benefits and risks of performing an action, we developed a cause-effect model. This model is based on five predicates.

1. *cause-event-event* to represent the effects that are *definitely* caused by an action
2. *may-cause* to represent the effects that *may* be caused by an action
3. *may-prevent* to represent actions that may be *prevented* by an action
4. *maleficiary* to represent the negative effects
5. *beneficiary* to represent the positive effects

The predicates *cause-event-event*, *beneficiary*, and *maleficiary* were reused from the HPKB-UL. Our example axioms use these predicates.

The first two axioms below were based on the proceedings of a Paris ministerial conference on terrorism cited in the source material for the CMCP specification.

> Take steps within their power to immediately review and amend as necessary their domestic anti- terrorist legislation to ensure, inter alia, that terrorists' acts are established as serious criminal offenses and that the seriousness of terrorists' acts is duly reflected in the sentence served. (Paris Ministerial Conference on Terrorism)

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

> Adopt effective domestic laws and regulations including export controls to govern manufacture, trading, transport, and export of firearms, explosives, or any device designed to cause violent injury, damage, or destruction in order to prevent their use for terrorists' acts. (Paris Ministerial Conference on Terrorism)

```
(forall ((?terrorist-attack terrorist-attack) (?country1 country)
          (?country2 country2) (?weapon))
(implies
    (performed-by ?terrorist-attack ?country1)
    (exists ((?action action))
    (and
        (may-prevent ?terrorist-attack ?action)
        (performed-by  ?action ?country2)
        (maleficiary ?action ?country1)
        (object-of-possession-transfer ?action ?weapon)
        (starts-after-starting-of ?action ?terrorist-attack)
        (from-possessor ?action ?country2)
        (to-possessor ?action ?country1)))))
```

> Counter-terrorism is a top priority for the Clinton Administration as it has sought aggressively to track down and punish terrorists worldwide and to fight international crime to the fullest extent of the law. (White House Fact Sheet on Counter Terrorism)

```
(forall ((?terrorist-attack terrorist-attack) (?agent agent))
   (implies
     (performed-by ?terrorist-attack ?agent)
     (exists ((?action action))
     (and
       (cause-event-event ?terrorist-attack ?action)
       (opposing ?action ?agent)
       (maleficiary ?action ?agent)
       (performed-by ?action united-states))))
```

Next, we give the axioms defining the benefits and risks.

If an action may cause another action, and that action benefits the doer of the first action, then that second action is a benefit of the first action.

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

If an action may cause another action, and that action is bad for the doer of the first action, then that second action is a risk of the first action.

```
(forall ((?action action) (?action1 action1) (?agent agent))
 (implies
   (and
     (may-cause ?action ?action1)
     (performed-by ?action ?agent)
     (beneficiary ?action1 ?agent))
 (risk-of-action  ?action ?action1 ?agent)))
```

Representing Viability

PQ44. Would [the {veiled, explicit} threat of] <InternationalActionType> [against <InternationalAgent2>] make sense as a foreign policy tool for <InternationalAgent1>?

A general analysis of when a foreign policy is viable can be complex. It is, however, possible to reduce the inference required for this question to capability and benefit risk analysis. We divided the reasoning required for this question into the following steps.

1. Determine if the agent is capable of executing the threat. If the answer is no, the threat is not viable.
2. Determine the benefits of the threatening action.
3. Determine the risks of the threatening action.
4. Weigh the benefits and risks to determine if the action makes sense. If an action has only risks but no benefits, it is not a viable option, and vice versa.

Whether the threat is veiled or explicit made a difference in some cases. For example, although the overt sponsorship or threat of terrorism is not a sensible foreign policy tool, a credible, veiled threat can be effective. For some actions, such as propaganda, or embargo on products that a country does not import, the difference did not make sense, and the detailed reasoning suggested above was not necessary.

Representing Asymmetric Threats

PQ95 Does <Force> of <Country> pose an asymmetric threat to [other] countries in <InternationalAgent>?

Asymmetric threats arise from inequality in power, which can be characterized by the range, number, and capacity of weapons and forces. We formalized this intuition by using the following axioms.

A weapon represents an asymmetric threat to a country if the country does not possess the same weapon and cannot defend against it.

```
 (forall ((?country1 country) (?weapon-class 1weapon-class))
 (=>
    (and
       (possesses ?country1 ?weapon-class1)
       (not (possesses ?country2 ?weapon-class1))
       (forall (?weapon-class2)
              (not
               (and
                (possesses ?country2 ?weapon-class2)
                (counters ?weapon-class2 ?weapon-class1)))))
      (deters ?country1 ?country2 ?weapon-class1))))
```

A missile represents a credible threat to a country if it has a longer range and larger payload than any of that country's own missiles.

```
(forall ((?country1 country) (?country2 country) (?missile-class1 missile-class))
   (=>
    (and
       (not (equal ?country1 ?country2))
       (possesses ?country1 ?missile-class1)
       (forall (?missile-class2)
             (=>
              (possesses ?country2 ?missile-class2)
              (and (< (in-km (range ?missile-class2))
                       (in-km (range ?missile-class1)))
                   (< (in-kg (payload ?missile-class2))
                       (in-kg (payload ?missile-class1)))))))
      (deters ?country1 ?country2 ?missile-class1))))
```

Representing Competition

PQ200. <ContextSpec>, what actors {have as an interest, compete regarding} <InterestSpec>?

The scenario KBs created by the SAIC team contained descriptions of interest of each agent in the scenario. Each interest was represented as an individual frame, and the *interest-of* slot on it indicated the agent who had that interest. The answer to the "have as an interest" part of this question reduced to a simple lookup of a ground fact. There are, of course, many cases when all the interests are not explicitly entered in the KB and need to be inferred. For example, if an agent performs a military action in an area, it suggests an interest in military presence in that area. This can be formalized with the following axiom.

```
(forall
 ((?military-action military-action) (?place place) (?agent agent))
 (=>
  (and
   (event-occurs-at ?military-action ?place)
   (performed-by ?military-action ?agent))
  (exists
   ((?military-presence military-presence)
    (?interest interest))
   (and
    (has-in-location ?military-presence ?place)
    (interest-of ?interest ?agent)
    (interest-in-thing ?interest ?military-presence)))))
```

To determine the answer to whether two agents compete in a scenario, two approaches are possible. First, for each event in the scenario, one could manually enter the agents that compete, and answer the question by a simple lookup. Second, we could draw inferences from objects already in the KB to determine which agents compete. That is indeed possible, as the description of each event included the two slots *supports-interest* and *opposes-interest* that represent the interests supported or opposed by that event. The competition between the two agents can be then inferred by using the following axiom.

```
(forall
 ((?action action) (?context context) (?agent agent)
                  (?agent1 agent) (?thing thing))
 (=>
  (and
   (occurs-in ?action ?context)
   (supports-interest ?action ?agent)
   (opposes-interest ?action ?agent1)
   (interest-of ?interest ?agent)
   (interest-of ?interest1 ?agent1)
   (has-interest-in-thing ?interest ?thing)
   (has-interest-in-thing ?interest1 ?thing))
  (competes ?agent ?agent1 ?context)))
```

Evaluation of KB Content

We tested the axioms we have described in this report in the annual CMCP evaluations. The evaluation during the first year of the project, CMCP-98, primarily focused on questions involving benefit/risks, military and terrorism capability, asymmetric threats, and viability of actions. The questions in the evaluation during the second year, CMCP-99, involved action, interest, interaction, and escalation.

The test questions were not previously seen by the developers of the KB, but derived from the question grammar that was known in advance. SMEs, who had degrees in political science, graded the answers. The correctness of the answer, recording the sources and explanation quality, was the criterion for scoring. Here, we primarily focus on the correctness scores of the answer.

Table 3. Correctness scores for the reasoning tasks

Reasoning Task	PQ Identifier	Tested in Evaluation	Number of Test Questions	Questions with non-zero Correctness Score	Average Correctness Score
Benefit/Risks	PQ39	CMCP-98	8	6	73%
Viability	PQ44	CMCP-98	2	2	100%
Asymmetric Threats	PQ95	CMCP-98	1	1	100%
Capability	PQ35, PQ77	CMCP-98	5	4	100%
Interests underlying an action	PQ236	CMCP-99	3	3	66.67%
Interests leading to an action	PQ237	CMCP-99	2	2	100%
Action responses	PQ219	CMCP-99	4	2	50%
Competing/Agreeing Interests	PQ239	CMCP-99	5	5	33%
Action Options	PQ220	CMCP-99	4	4	50%
Escalation	PQ210	CMCP-99	4	3	77%
Competition	PQ201	CMCP-99	2	2	100%

A summary of correctness scores is shown in Table 3. The first column specifies the reasoning task tested. The second column specifies the PQ identifier used to identify the question in the CMCP specification. The third column specifies the evaluation in which the question was tested. The fourth column gives the number of test questions asked during the evaluation. The sample sizes are small because the SME time needed to generate and grade the answers was expensive. The scores are reported only for the final evaluation. For CMCP-98, we report the scores for the final batch, that is, TQD. For CMCP-99, we report the scores for the year-end evaluation, except for the PQ236, for which the scores are reported from the first mini-evaluation. The mini-evaluation for PQ236 was considered because, as a matter of coincidence, the axioms used in the final evaluation for PQ236 did not make use of the axiom template discussed in this report. The fifth column of specifies the questions with non-zero scores. The scores were given on a scale from 0 to 3. The last column shows the average correctness score for the questions with non-zero scores.

We can see that the axioms for viability, asymmetric threats, capability, competition, and interests leading to actions worked well at least for the questions on which they were tested. For questions involving interests underlying an action, the average score was 2 (or 66.67%); for questions involving benefits and risks, it was 2.2 (or 73%). The main reason for the less-than-perfect score was lack of precision in the answer: either too many answers were returned or the answers did not take into account the details of the situation. For example, one of the questions asked about the interests underlying sanctions imposed against an ally. As an answer to this question, the SMEs expected the system to take into account the fact that the sanctions are being imposed against an ally; the axiom template did not capture that fact.

The average score for the question involving action responses was 1.5 (or 50%). The axiom driving the reasoning “If ?action1 performed-by ?agent2 negatively influences ?interest1 of ?agent1, and ?action2 positively influences ?interest1, then ?action2 is a viable response to ?action1 with respect to ?agent2” did not meet the SME approval. For example, the imposition of U.S. sanctions on China would damage Chinese economic interests, whereas a trade agreement would support China’s economic interests. But a Chinese trade initiative to the U.S. would not be “a viable response” to the U.S. sanctions.

The axioms for competing/agreeing interests had a similar problem. The SME feedback on these axioms was that just because an action can be taken to further more than one type of interest does not mean that those interests have anything in common. For example, economic sanctions are a typical kind of measure used in support of a wide variety of foreign policy, national security, and trade goals. One cannot argue that all of these goals “agree” with each other—many, in fact, are competing.

The average score for reasoning with escalation was 2.3 (or 77%). For the case that had less than the perfect score, the error was in the KB encoding that gave the incorrect inference. The inferences produced by the model were accurate.

KNOWLEDGE SERVER DEVELOPMENT

We made extensive use of our KB development environment during this project: the theorem prover SNARK, knowledge representation system Ocelot, graphical browser GKB-Editor, and OKBC. Our tools had to be extended in many directions. Some of the extensions were motivated by the requirements of the challenge problems. Others were motivated by a long-term plan to incorporate enhanced functionality in our tools.

Ocelot Enhancements

Ocelot is a frame representation system and provides efficient storage and retrieval of large KBs. It allows storage of a KB in ORACLE and supports multi-user access. It also is the server for the GKB-Editor. It can be accessed using Open Knowledge Base Connectivity (OKBC) a generic API for accessing knowledge servers that was developed jointly by KSL Stanford and SRI.

Loading HPKB Upper Ontology

Cycorp released the HPKB upper ontology in MELD format, which had to be loaded into Ocelot—our in-house OKBC-compliant frame representation system (FRS). Therefore, we wrote a loader program that reads the upper ontology and generates OKBC commands to define the upper ontology in Ocelot. With this loader, the HPKB upper ontology can be loaded into any OKBC-compliant knowledge server, such as Ontolingua.

Mapping many of the MELD constructs into OKBC constructs was straightforward. For example, *isa* in MELD is equivalent to *instance-of* in OKBC, and *genls* is equivalent to *subclass-of*. OKBC does not have direct support for nary relations. Therefore, there was no analog in OKBC for the MELD construct *arg3isa* that defines the type of the third argument of a relation. To handle such cases, we introduced a special slot called *:sentences* that can be used to record any logical expressions. For example, MELD assertion *(arg3Isa RectangularSolidFn Distance)* can be represented by a value *(nth-domain RectangularSolidFn 3 Distance)* of the slot *:sentences*. Any construct that could not be directly mapped to an equivalent OKBC construct was recorded in the *:sentences* in the slot to prevent any loss of information.

We developed several utility programs to selectively load and store a KB. The KB was kept in multiple flat files, some of which were edited using emacs and at other times using the GKB-Editor. When a file containing a portion of a KB is to be edited with the GKB-Editor, it must be loaded into Ocelot, and one must deal with the fact that it does not contain all the frame definitions. Our utility programs were sensitive to this situation, by assuming definitions for missing frames and ignoring them while saving the KB.

Instrumentation for Metrics Collection

During the project, we collected empirical data on KB size, reuse rates, object age, and the usage of axioms in answering the questions. The metrics collection software was implemented using Ocelot and OKBC.

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. (It was developed in cooperation with Adam Pease of Teknowledge.)

- 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.

Even though this categorization is imperfect, 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 represent knowledge that can be found in an almanac. 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 explanation printing routines were instrumented to output the constants and axioms used during a proof. We did not have a direct time stamping of axioms in the KB, but approximate creation dates were derived from the backups of a KB kept at different dates. By comparing two KBs, we determined which new axioms and constants were added and assigned them a creation date at the midpoint of the old backup date and the new backup date.

SNARK Enhancements

SNARK is one of the leading theorem provers in the world, and has been developed at SRI under the sponsorship of several Government projects including HPKB. The deductive capabilities of SNARK were extensively exploited during the CMCP evaluations. During HPKBY1, minimal changes were needed in the SNARK kernel, as it is a solid and stable inference technology. We needed, however, to address numerous interface issues to use SNARK effectively. For example, we developed a partial KIF interface to SNARK, a module to help users pose CMCP questions as theorems, and a module to generate an explanation of an answer.

KIF/OKBC Support

Since several users are familiar with KIF and the OKBC knowledge model, we developed an interface to SNARK for accepting input in KIF syntax and to recognize OKBC relation names. The KIF interface of SNARK accepts both ANSI KIF and KIF 3.0.
For most practical uses, KIF is usually extended by defining several short hands or helper relations. The KIF interface supports many such relations, most of which are based on the OKBC knowledge model, and some of which are based on Ontolingua.

The KIF/OKBC interface of SNARK assumes that the domain of discourse consists of classes, relations, functions, individuals, and named assertions. A class, as defined in OKBC, is the same as a unary relation. The slots, as defined in the OKBC knowledge model, are the same as binary relations.

Declaring Classes

A class is declared using the KIF *defrelation* form. A *defrelation* form for a class A must include an assertion of the form (class A). A *defrelation* form can include atomic sentences defined using the relations *subclass*-of, *instance*-of, :*documentation*, and *template-slot-value* defined in the OKBC knowledge model. An example class declaration is the following.

```
(defrelation Person
   "Collection of all human beings."
 (class Person)
 (subclass-of Person Human)
 (instance-of Person Biological-Classification-Type)
 (template-slot-value Person Average-age 70)
 (synonymous-external-concept Person Sensus-Information-1997 "Person")
 )
```

ANSI KIF allows the specification of a documentation string as the second argument of the *defrelation* form, whereas KIF 3.0 does not. For KIF 3.0, the documentation string can be specified using the :*documentation* relation. Any other slot value axioms about a class, such as the one involving synonymous-*external*-concepts, may be embedded in the *defrelation* form. Even though KIF (and our KIF interface) will allow any other arbitrary axioms in the defrelation form, we recommend not doing that to retain the object-oriented feel of the input.

Declaring Relations

Relations are declared using the KIF *defrelation* form, and cannot include any sentence with the relation symbol *class* in the body. A relation declaration must include an arity assertion. The arity can be declared using either *arity* or *relation-arity*. The type restriction on the relation arguments can be declared in several equivalent ways. For binary relations, one can use domain and *slot-value-type* as defined in the OKBC knowledge model. For relations with arity greater than 2, one can use *nth-domain* and *nth-domain-subclass-of* relations. Alternatively, one can use the *domains* relation to give the list of type restrictions.

```
(defrelation beneficiary
   "(beneficiary act agt) means that the >agent agt benefits from the performance of the action act."
  (relation-arity beneficiary 2)
  (domain beneficiary event)
  (slot-value-type beneficiary agent)
  )
```

Instead of using domain, and slot-value-type, the same type restriction can be asserted as

```
(nth-domain beneficiary 1 event)
(nth-domain beneficiary 2 agent)
```

Alternatively, one can assert

```
(domains beneficiary (event agent))
```

The same declaration can also be made without using the helper relations.

```
(defrelation beneficiary (?event ?agent)
  "(beneficiary act agt) means that the >agent agt benefits from the performance of the action act."
  :=>
  (and
   (event ?event)
   (agent ?agent)))
```

The disadvantage of using the above form is that it does not allow any additional axioms in the body, and is also less familiar to OKBC users. Therefore, we recommend using the helper relations.

Declaring Functions

Relations are declared using the KIF *deffunction* form. A function declaration must include an arity assertion. The arity can be declared using either arity or function-arity relation. The type restriction on the function arguments can be declared in several equivalent ways. For unary functions, one can use the domain relation. For functions with arity greater than 1, one can use the *nth-domain* and *nth-domain-subclass-of* relations. The return value is specified using the range relation. The following is an example declaration.

```
(deffunction month-fn
    "(Month-Fn ?M ?YR) denotes a Calendar-Month -- in particular, the month of type ?M
during ?YR."
  (function-arity month-fn 2)
  (nth-domain month-fn 1 month-of-year-type)
  (nth-domain month-fn 2 calendar-year)
  (nth-domain-subclass-of month-fn 1 calendar-month)
  (range month-fn calendar-month))
```

The axiom *(nth-domain-subclass-of month-fn 1 calendar-month)* means that the first argument of *month-fn* should be a subclass of *calendar-month*. The *relation range-subclass-of* has a similar meaning.

The *nth-domain* axioms in the above declaration can be replaced by

```
(domains month-fn (month-of-year-type calendar-year))
```

An alternative but not equivalent declaration for this function is

```
(deffunction month-fn (?month ?year)
    "(Month-Fn ?M ?YR) denotes a Calendar-Month -- in particular, the month of type ?M
during ?YR."
  :-> ?calendar-month
  :=>
  (and
   (Month-of-year-type ?month)
   (calendar-year ?year)))
```

It is not possible to include the range-subclass-of restriction in this declaration. Therefore, the declaration using the helper relations is preferable and recommended.

Declaring Individuals

Individuals are declared using the KIF *defobject* form. An individual declaration includes *instance-of* assertions and several slot values. The following is an example individual declaration.

```
(defobject gcc
   "Gulf cooperation council"
 (instance-of gcc multilateral-agent)
 (instance-of gcc group)
 (opposed-diplomatically gcc iran persian-gulf-war-of-1991 after-1991)
 (group-members gcc saudi-arabia)
 (group-members gcc united-arab-emirates)
 (group-members gcc bahrain)
 (group-members gcc oman)
 (group-members gcc qatar)
 (group-members gcc kuwait))
```

Declaring Named Assertions

The *assertion* form is an extension to KIF. It can include any axiom and is primarily intended to define named axioms. The following is an example named assertion.

```
(assertion
 (forall ((?action action) (?action1 action) (?country country))
        (=>
         (and
          (may-cause ?action ?action1)
          (performed-by ?action ?country)
          (maleficiary ?action1 ?country)
          )
         (risk-of-action ?action ?country ?action1 )
         ))
 :name risks-of-action-1
 :documentation "If an action1 may possibly cause another action2,  and
 that action2 harms the doer of action1, then that action2 is a risk of action1
 Commonsense axiom CMCP-98/SRI.")
```

OKBC Features not Supported

The two mandatory features of OKBC are not currently supported. The first involves the use of the *slot-of* relation and the second involves value restriction along the *subclass-of* hierarchy.

The *slot-of* relation associates a slot with a frame; *(slot-of S F)* means that the frame *F* can have the slot *S*. The KIF/OKBC interface of SNARK assumes that if *F* is an instance of the domain of *S*, *F* can have the slot *S*. This is stronger than required by the OKBC knowledge model. In the OKBC knowledge model, the browsing and indexing considerations that were not really an issue for SNARK motivated the *slot-of* relation.

In the OKBC knowledge model, it is possible to restrict the values of a relation as one goes down the *subclass-of* hierarchy. For example, one could have the following assertions.

(value-type car fueled-by fuel)
(subclass-of honda car)(value-type honda fueled-by petrol)

The *value-type* assertion for *honda* restricts the value type of the slot *fueled-by* to values that are of type *petrol*. Such value restriction is not supported at present**.**
The following nonmandatory relations are not supported in the interface: *:inverse, :slot-inverse, :cardinality, :slot-cardinality, :minimum-cardinality, :slot-minimum-cardinality, :maximum-cardinality, :slot-maximum-cardinality, :same-values, :slot-same-values, :not-same-values, :slot-not-same-values, :subset-of-values, :slot-subset-of-values, :numeric-minimum, :slot-numeric-minimum, :numeric-maximum, :slot-numeric-maximum, :some-values, :slot-some-values, :collection-type,* and *:slot-collection-type.*

Other Useful Features of the KIF/OKBC Interface

For every class, a typed variable with the same name as the class is automatically declared. For example, for the class action, *?action* (?*action1*, ?*action2*, etc.) is automatically declared as a variable of type action. It is not mandatory to name the variables in this fashion, but this feature is convenient shorthand. (It also speeds up inference and makes the formulae shorter.)

For the writing of axioms, both *(R ?x)* and *(instance-of R ?x)* are accepted even though not preferred. The recommended way to specify the type of a variable is to use the quantifiers. For example,

```
 (forall ((?action action) (?action1 action) (?country country))
        (=>
         (and
          (may-cause ?action ?action1)
          (performed-by ?action ?country)
          (maleficiary ?action1 ?country)
          )
         (risk-of-action ?action ?country ?action1 )
        ))
```

This axiom can be written in either of the following two ways.

```
 (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)
         ))

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

The KIF/OKBC interface also has a sort-coercion feature. For example, if the above axiom omits the type restrictions, and is written as

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

the variables are automatically coerced to appropriate type. Invoking (use-well-sorting t) can enable this feature.

Some relations apply to classes. For example, *the (device-class-used action-1 pen)* means that some instance of *pen* was used in performing *action*-1. For any axioms involving *device-class-used* that require a variable, one needs to specify the type of that variable. Consider one such axiom asserting that if an agent performs an action using a device of a certain type, it possesses a device of that type.

```
(=> (and
     (performed-by ?action ?agent)
     (device-class-used ?action ?device))
    (possesses-class ?action ?device))
```

The type restriction on a relation such as *device-class-used* is specified using *value-type-subclass-of*. Suppose the *value-type-subclass-of* for *device-class-used* is *physical-object*. Then only the subclasses of *physical-object* can be in the second argument position. Several alternatives enforce this restriction. First, one can create a *meta*-class called *physical-object-type* such that all subclasses of *physical-object* are its instances. If such a meta-class is needed for representing the knowledge of interest, it is a reasonable solution, but otherwise it is artificial. Another alternative is to use a variable of the most general type and then restrict its type by using the *subclass-of* relation. For example,

```
(=> (and
     (performed-by ?action ?agent)
     (subclass-of ?device physical-object)
     (device-class-used ?action ?device))
    (possesses-class ?action ?device))
```

This solution defeats the purpose of introducing the sort subsystem of SNARK because it requires adding an extra literal in the axiom to encode the sort information. To deal with this problem, the KIF/OKBC interface creates additional declarations for every class. For a *(class X)* assertion, it automatically *declares (class subclass-of-X)* and *(instance-of X subclass-of-X).* For a *(subclass-of X Y)* assertion, it declares *(subclass-of subclass-of-X subclass-of-Y).*

Using these additional sorts, the same axiom can be written as

```
(=> (and
     (performed-by ?action ?agent)
     (subclass-of ?device ?subclass-of-physical-object)
     (device-class-used ?action ?subclass-of-physical-object))
    (possesses-class ?action ?subclass-of-physical-object))
```

The above encoding exploits the compactness of the sort subsystem. It is, of course, not necessary to use the special variable names. Instead, one can embed this restriction in the quantifiers as follows.

```
(forall ((?action action) (?agent agent)
                    (?device :subclass-of physical-object))
        (=> (and
```

(performed-by ?action ?agent)
(subclass-of ?device ?device)
(device-class-used ?action ?device)
(possesses-class ?action ?device))))

The resulting form falls outside ANSI KIF, but is necessary in practice.

The KIF/OKBC accepts the following top-level forms. Placing the above forms in a source file enables the KIF interface.

(in-language :hpkb-with-ansi-kif)
(in-language :hpkb-with-kif-3.0)

Placing the following forms in a source file associates <string1> as an author and <string2> as the source of every axiom in that file.

(has-author <string1>)
(has-source <string2>)

Efficient Reasoning with Sorts

When HPKB-UL was initially loaded into SNARK's sort system, the loading time was unacceptable. At that time, SNARK was using Binary Decision Diagrams (BDDs) to reason with sorts. Because of the poor performance of BDDs, we switched to the Davis Putnam Procedure (DPP), which remarkably improved the performance of reasoning with the sort structure of HPKB-UL. This is an interesting empirical result.

More generally, one can rely on an external reasoner to support common taxonomic inferences. Even though the DPP is able to support *subclass-of* and *instance-of* inferences, it cannot support many other common frame system inferences such as range and cardinality constraints. It is also possible that a special-purpose taxonomic reasoner will be much faster than the DPP. Also, the DPP is not incremental and does not allow any changes in the sort theory once it has been loaded. In our future work, we hope to undertake a more detailed comparison of DPP with a specialized taxonomic reasoner.

Explanation Generation

For an answer produced by a system to be credible, it must be accompanied by an explanation of how the answer was obtained. The system must also cite the source from which the knowledge was obtained. We developed an explanation module for SNARK to meet these objectives. The output of our explanation module shows all the axioms used and the intermediate inference steps. Each inference step shows the axioms, inference method, and rewrites used in that step, the conclusion derived, and the current answer term. If an English description of an axiom is available, it is shown. The output is produced in the HTML format, making it easier for a user to navigate among several inference steps.

Based on the feedback received from the evaluation team, the most useful part of the explanation was an English description of the axioms used during the inference process. In most cases, the evaluators did not look at the logical representation of axioms. We gave a separate explanation for each answer. The evaluators preferred to see a combined explanation for all the answers. We improved the explanation module to incorporate this feedback. A summary of the axioms used in all the answers to a question is printed at the top. The summary is computed by printing an axiom only once. The next possible refinement would be to organize the axioms in a coherent manner.

We also developed an initial version of "drill-down" capability that allows a user to get more information about a term used in a SNARK answer. It is achieved by invoking the GKB-Editor from the HTML output produced by SNARK.

All the terms used in a proof must be defined. We do this by including the documentation string for each term used in a proof in the HTML page for that proof. Sometimes the documentation refers to other terms in the KB. The HTML page does not contain definitions for the terms that are indirectly referenced, because, if it did, it could potentially include a large number of terms. We embed a link to our GKB server in the HTML page, which when clicked shows that term in the taxonomy browser of the GKB-Editor. The user can then explore the KB further and get to indirectly referenced terms.

The GKB connection also helps in giving answers that are more abstract. For example, for the question, "What actions can the United States take to foster a strengthening of economic commitments to Azerbaijan?" there can be either one answer – trade actions— or several answers, each of which is a subclass of trade actions. It is difficult to anticipate the level of detail expected by a user. With a GKB connection, we can support both levels of detail: we can return trade actions as an answer, and make the information about subclasses available in the taxonomy browser. We expect the graphical capabilities of the GKB-Editor to be increasingly useful in producing intuitive explanations for SNARK proofs.

Organizing the output of the explanation subsystem in a coherent and intuitive manner is clearly the most immediate priority for future work. We plan to investigate the use of explanation design plans to achieve this objective (Lester and Porter 1997).

Collaboration System

Our collaboration system PERK allows multiple users to make changes to a shared KB. PERK maintains a public copy of a KB and records changes made by a user in a log. When the user has finished making all the changes, PERK compares that user's log with the logs of any other concurrent users who have changed the KB in the meantime. It identifies any conflicting operations and informs the user. Our previous work tracked only the changes that did not involve any updates on the KB schema. During this project, we enhanced PERK to deal with schema changes. Some of the KB editing is done using emacs, and the PERK model of access control does not apply. To deal with such situations, we developed KB synchronization tools.

Supporting Schema Changes in the Collaboration System

Two difficulties had to be addressed in the process of supporting schema changes: taking into account the indirect effects and checking conflicts.

Schema changes invariably cause both *direct* and *indirect* changes to a frame. A *direct* change to a frame is one that does not update any frame other than itself. An *indirect* change is one that updates a frame other than the frame on which an operation is executed. For example, an operation that renames a frame from *A* to *B* will update not only that frame, but also every other frame that references it. For example, if *A* is a slot value of another frame *C*, it must be changed to B. Even though less common, there can be indirect effects of operations that do not necessarily involve changing the schema. For example, consider the slots *father-of* and *child-of* that are defined as the inverse of each other. When we add a *father-of* value *A* for frame *B*, it also asserts a value *B* for a *child-of* slot of *A*.

It is necessary to do a special treatment of schema changes for two reasons. First, at the time of conflict checking, we must be able to accurately compute both the direct and indirect effects of an operation. The indirect effects depend on the state of the KB in which it is executed. The KB states at execution time and conflict checking time are not necessarily the same. Therefore, we need to explicitly record all the indirect effects of an operation. Second, while storing the log entries in a database, we index them on the frame on which the operation is executed. Because of indirect effects, a log entry must be indexed on all the frames that it updates, so that we can determine all the updates on a frame over a time interval.

A possible way to record the indirect effects is to record them as a "before image" in each log entry and index it on all the frames it updates. Each log entry is a row in the Log Table in the database, and records only one value of a frame for each row. Indexing a row on multiple frames means duplicating it for each different value of a frame. While faulting a frame, PERK checks for any updates that have been applied on that frame since the user started and reverses them so that the user always sees the state of the KB from which he started. To successfully reverse the updates on a frame, we must also reverse any indirect effects of that update. Therefore, we must also fault in all the frames involved in the indirect update. In some cases, this can mean faulting in numerous frames.

An alternative solution is to log each indirect effect as an independent direct effect, and while faulting a frame, only reverse the direct effects. For example, consider an operation that renames a frame *A* to *B*. Suppose its indirect effect on a frame *C* is to change the value of slot *S* from *A* to *B*. The direct effect on frame *A* will be logged as a *rename-frame* operation. It will be indexed on *A* and *B*. The indirect effect on frame *C* will be logged as an independent *replace-slot-value* operation. It will be indexed on *C*. If B is faulted in, and the *rename-frame* operation needs to be undone, the update on *B* will be reversed, renaming it back to *A*. Similarly, when the frame *C* is faulted in, the *replace-slot-value* on *C* will be reversed. This solution avoids redundancy in the Log Table and allows PERK to fault a frame without necessarily faulting every other frame involved in some indirect change with that frame. Therefore, we implemented this solution for PERK.

For checking conflicts caused by schema changes, we use the approach we use for checking conflicts caused by non-schema changes. To check conflicts, we translate each log operation into an operation on a graph representation of the KB. We implement translations from the schema operations in OKBC to graph operations. Since the indirect effects of an operation are independently recorded, the translation for the indirect effects of an operation does not require any separate effort. We have previously developed a matrix for checking conflicts between operations on a KB graph. The same conflict matrix applies to graph operations resulting from schema changes.

The work on conflict checking is not complete; much more can be done. For example, it is possible to take into account the constraint information available in the schema to do better conflict checking. An important aspect of the KB is deductive rules. We do not yet have a scheme that allows concurrent updates of rules.

We have completed the design and implementation work to support schema changes in PERK. Using PERK in a production setting remains open for future work. This work will allow us to test the adequacy of the conflict detection scheme and of the user interface for reporting conflicts.

Knowledge Base Synchronization

The current design of our collaboration system is based on the assumption that all the editing of a KB is done using OKBC. If that is the case, our collaboration system controls the multiuser updates to a KB. That assumption does not always hold because some of the editing of a KB is not done by OKBC, but by the use of conventional text file editing tools such as emacs. In such cases, we need a facility to compare the updates of multiple users and resolve any potential problems. Motivated by this need, we developed an interactive KB file comparison and merging tool that helps multiple KEs work together.

The interactive KB file comparison and merging tool works on the assumption that the KEs start from the same KB source file and make independent edits to it. Once they are done, it loads their results into an OKBC knowledge server and compares the two KB files. It generates a report listing frames that are in one KB but not in the other (suggesting deletions or insertions) and the frames that differ in slot and facet values (suggesting conflicting updates). It also allows KEs to step through each difference and gives them an option of favoring an update from one KB over the other. It finally produces a third KB that is the result of the merge. The tool provides much greater support than the conventional Unix *diff* tools, as it takes advantage of the structured nature of the KB in doing the comparisons.

PC Ports

We have ported the GKB-Editor to run under a Windows environment on a PC. This was done to support SAIC's HPKB team that primarily uses PCs for development. This was a significant effort because it also required us to port several other support subsystems, for example, Grasper and the Library. Ocelot, SNARK, and OKBC were previously ported by us and are in use by SAIC. Most of our software now runs on PCs except for the GKB WWW server that requires us to undertake a PC port for a C library for GIF generation. We also plan to streamline our distribution procedure for the PC version of our software. (A distribution procedure for the Sun/Unix platform is already in place.)

TECHNOLOGY TRANSFER TO PROJECT GENOA

DARPA's project Genoa, focused at helping crisis analysts, is a target application for the technology produced by the HPKB project. Since our HPKB work focused on the crisis management challenge problem, and we also participate in project Genoa, it was natural for us to explore avenues for possible technology transfer for our HPKB results. We have investigated the transfer of the knowledge server and the KB content developed during the HPKB project to project Genoa.

Transferring KB Content

In SRI's project Genoa, Scope Evidential Argumentation System (SEAS), the approach to supporting analysts engaged in crisis prediction is based upon the idea that for any given type of crisis, a description of how one argues for and against the inevitability of such a crisis can be codified. An organized set of questions is used to determine if a crisis of the given type is possible, that highlights the information that needs/remains to be collected to confirm or deny this hypothesis, and whose answers can be used to explain the rationale of these predictions when this argument is being briefed. In essence, such an argument identifies and structures the crisis indications and warning signs.

For any given crisis type, we anticipate there being several key questions that must be answered in the process of analyzing it. We expect there to be several, somewhat redundant, ways to go about answering any one of these questions. This will give rise to several supporting questions, whose answers are pooled to reliably assess the answer to the question that they support. This hierarchy of questions supporting questions may go a few levels deep before bottoming out in questions that must be directly assessed and answered.

A natural avenue for transferring the KB content and inference capability to Project Genoa was to identify places where the hierarchy of questions bottoms out and to see if some of those questions could be answered using our HPKB KB. When a Genoa question bottoms out, an external discovery tool can be invoked. The external discovery tool can be an HPKB inference procedure or a pointer to a portion of the KB that may be useful in answering the Genoa question.

As an example, consider the analysis of external factors of nation states from project Genoa. Of several questions where the analysis bottoms out, one of the questions is as follows.

WMD/ADVANCED WEAPONS: Is an adversary trying to acquire or is in the process of deploying WMD or advanced weapons?

Consider the following:
WMD proliferation activities or WMD program developments
Nuclear tests or test preparations

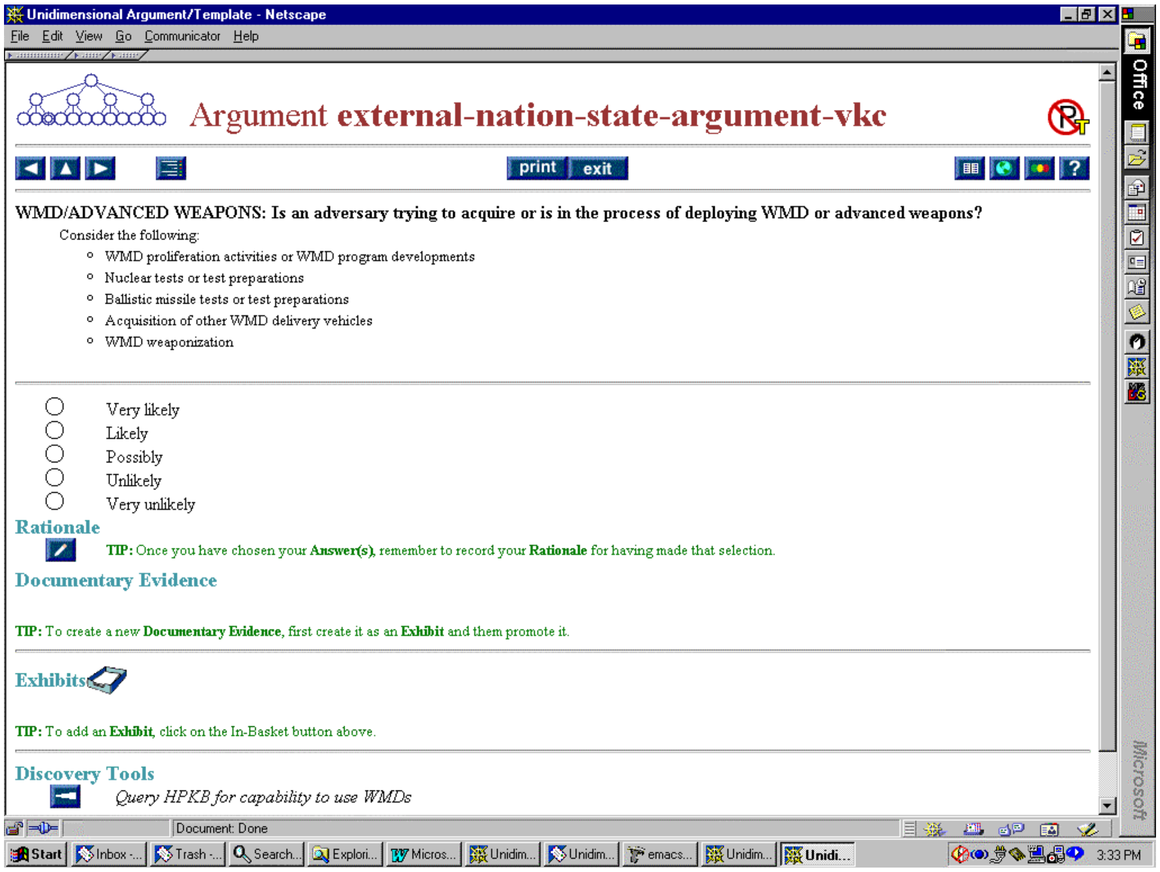
Ballistic missile tests or test preparations
Acquisition of other WMD delivery vehicles
WMD weaponization
WMD deployment
Advanced conventional weapon acquisition activities
Acquisition or enhancement of force projection capabilities
Deployment of advanced conventional weapons

A closely matching question from the HPKB is

Can Country1 sponsor terrorism with weapons of mass destruction inside Country2's borders?

An inference procedure to answer this question can be invoked as a discovery tool. For example, Figure 12 shows the Genoa screen where the question bottoms out. The pointer to the HPKB decision procedure is embedded at the bottom of the page. When a user clicks on it, a form-based interface allows the specification of country to be analyzed, and the result is returned as an HTML form.

We have incorporated about half a dozen discovery tools in the analysis of the external factors of a nation state. These discovery tools perform inference tasks such as military capability analysis and terrorist group information.



Argument **external-nation-state-argument-vkc**

WMD/ADVANCED WEAPONS: Is an adversary trying to acquire or is in the process of deploying WMD or advanced weapons?

Consider the following:
- WMD proliferation activities or WMD program developments
- Nuclear tests or test preparations
- Ballistic missile tests or test preparations
- Acquisition of other WMD delivery vehicles
- WMD weaponization

○ Very likely
○ Likely
○ Possibly
○ Unlikely
○ Very unlikely

Rationale

TIP: Once you have chosen your **Answer(s)**, remember to record your **Rationale** for having made that selection.

Documentary Evidence

TIP: To create a new **Documentary Evidence**, first create it as an **Exhibit** and them promote it.

Exhibits

TIP: To add an **Exhibit**, click on the In-Basket button above.

Discovery Tools

Query HPKB for capability to use WMDs

Figure 13. An example of embedding a discovery tool in an argument

The project Genoa KB also uses an agent's ontology and a situation ontology. During the HPKB project, both ontologies were developed. We hope to transfer these ontologies to project Genoa in our future work.

Transferring Knowledge Server Technology

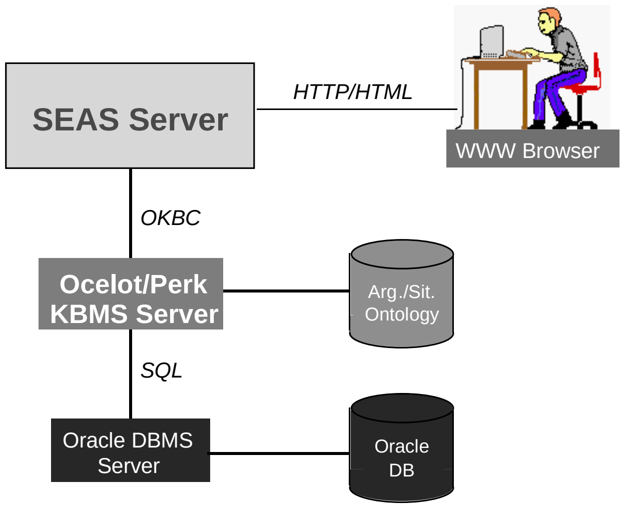


Figure 14. SEAS server relies on Ocelot as its knowledge server. SEAS accesses Ocelot using OKBC.

The SEAS server uses the knowledge server Ocelot and the OKBC for storing and retrieving its KB. Both these systems were developed under the HPKB project, and were transferred to the SEAS server without any additional work (see Figure 14). The SEAS server also needed a KB synchronization tool because one copy of the SEAS server is operational at DARPA's Technology Integration Center and another copy at SRI. These copies are sometimes updated in parallel. The SEAS server also requires a model for multiuser access control. We undertook an analysis of the requirements of the SEAS server and developed the following scheme to support multiuser access.

Three possible modes for accessing the SEAS server are

1. Using the web-based SEAS environment. The client is a WWW browser such as Netscape.

2. Using the Network OKBC via an application program. The client can be a C, Lisp, or Java program.

3. Keeping the knowledge base in Oracle, and replicating the SEAS server for each of the users.

The current architecture for the SEAS server follows the first model. The second model is not very practical and not encouraged. The Ecocyc server was built on the third model. The third model is unlikely to be adopted for the SEAS server, as we do not want any client-side software requirements.

The software from the third access model is applicable to the first model. For example, we can still store the knowledge base in Oracle, and the SEAS server then acts like a client to the Oracle server. The SEAS server, however, is not the end-user client. The end-user client is still Netscape and connects to the SEAS server.

For the third model, we have previously developed a technique for multiuser access that is primarily geared toward long transactions. It is based on the assumption that while updates are made to a KB, the updates should be isolated from the other concurrent users until the user is ready to commit those changes, and any inconsistent changes are then identified at the commit time.

For the SEAS server, the application characteristics are as follows.

- The transactions are short.
- There is a well-defined access-control policy:
  - An argument template, once published, cannot be updated. (This turns out to be irrelevant, because it means only that some objects in the KB are read-only).
  - Every argument and template has an owner, and can be updated only by the owner. (This also turns out to be irrelevant, because some arguments may have multiple owners.)
  - Some slot values are inferred as part of a chain of inference. Inconsistency in such slot values is not a concern because it can be fixed by simply rerunning the inference.
- Weak isolation such as “last person wins” is acceptable.
- No schema changes can occur, but new objects can be created.

Support of multiuser access in such an environment has two alternatives.

- Adapt the existing software.
- Devise a new scheme to take advantage of the application characteristics.

To adapt the existing software, we will need to give each user a separate copy of the KB. The KB and all the associated changes will be stored to Oracle. A user’s will not be visible to other users until the user presses the save button. Changes will be visible to current users if they refresh the state of their KBs. Given the short transactions and the access control policies, the conflicts should be rare, and the user should never have to see them. The advantages of this approach are that (1) we can use our existing software, (2) we can keep track of changes made by each user, and (3) it integrates gracefully with the third mode of accessing the server. The disadvantages are that (1) we are forced to have

Oracle in the architecture, (2) changes are not visible until they are saved, and (3) the conflict checking adds an extra layer of complexity.

An alternative solution would be not giving each user a different copy of the KB. Then we would need to provide access control for (1) shared global variables and data structures, and (2) data in the KB.

To support access control over shared data structures, we will need to assure that whenever a global variable is accessed, it will have to be accessed using a lock. Data-structure protection will have to be assured for several low-level Ocelot functions: *put-raw-values, add-raw-value, remove-raw-value, remove-raw-slot-values,* and *add-frame-to-kb.* The raw value functions will require a short duration lock on a frame, and *add-frame-to-kb* will require a short duration lock on the KB object.

To support access control for data in the KB, we need to consider insert/delete operations on frames and add/delete/put operations on slot values, and we must identify the necessary access control. Insert/delete operations on a frame will require locking the frame before the start of an operation. While a frame is locked, it can be read, but no one else can insert/delete it. This works fine because of the application characteristics listed earlier. For the add/delete/put of slot values, the slot value must be locked. While a value is locked, it may be read, but no one else can add/delete/put it. Since adding/deleting a frame may require adding/deleting values, the locks on slot values can be granted only if the frame is not locked. Similarly, a frame can be locked only if none of the slot values is being updated. If transactions require adding/deleting multiple frames/values, all locks must be acquired before start of the operation to avoid deadlocks.

The advantages of this approach are that (1) it does not depend on and require Oracle, (2) the changes are immediately visible, and (3) it requires minimal extra code. The disadvantages are that (1) users may be annoyed with dirty reads, (2) it is specific to the characteristics of the SEAS server, and (3) it does not gracefully integrate with the A3 mode of access. We have implemented the second approach for the SEAS server.

SUMMARY

Traditional KB development efforts start from scratch, and the cost of creating new KB content remains a serious bottleneck. Addressing this limitation requires two kinds of technique: one must have a large body of knowledge to start from, and one must have techniques to exploit that knowledge. This project produced several results for the second technique, that is, taking an existing body of knowledge and reusing it. Our approach to reuse had the following steps: translation, comprehension, slicing, merging, and reformulation. Translation and comprehension techniques required an engineering effort. The application of existing tools, slicing, and reformulation required the development of new algorithms and the identification of representation tradeoffs. By slicing an existing KB, one need use only that portion of a KB which is needed and satisfactory for a new application. Our experimental results with the HPKB-UL and the SAIC merged ontology showed that such slicing is indeed possible in practice. We reformulated the representation from the HPKB-UL to make it suitable for our inference tools. We made several measurements to show the level of reuse. The level of reuse, among many other factors, depends on the match between the already-available content and the needs of a new application.

We investigated several techniques for developing new KB content. We made significant use of axiom templates to create KB content. We demonstrated the effectiveness of the axiom template by testing it on several challenge questions. Preliminary investigations into principles of taxonomy design, alternative KB modularization, and compositionality in representation were undertaken.

We developed representations for several commonsense notions. We adapted the representation of temporal knowledge from the HPKB-UL and incorporated a temporal reasoner into our theorem prover. We formalized qualitative influences between actions and interests and used them to perform inferences on such actions that underlie an interest and the interests that may lead to an action. The qualitative relationships between actions were also used to represent benefits and risks, which were then used to assess the viability of an action. We also developed a model for reasoning about *escalation of conflict* in a scenario. The model was based on linear orders that were composed using lexicographic ordering. We represented the requirements for the capability for performing terrorist and military attacks. We also represented a few common cases of asymmetry in military capability of agents.

We developed a KIF/OKBC interface to our theorem prover SNARK. This interface accepts input in a subset of KIF and recognizes some of the relation names from the OKBC knowledge model. We developed a facility to print explanations for the proofs produced by SNARK. We developed tools for translating, loading, and saving ontologies encoded in a subset of KIF. The ontologies were loaded into Ocelot, which is a frame representation system developed at SRI. We extended our collaboration system to deal with schema changes and to synchronize the divergent copies of a KB. Instrumentation was developed to compute statistics on KB size, reuse, and axiom creation time.

The technology developed during the project was transferred to Project Genoa, one of the target applications. The deductive tasks involving capabilities can be invoked as discovery tools from the SEAS server. We also expect that the ontologies for representing events and agents will be transferred to project Genoa in the near future. The Genoa server is using the knowledge server Ocelot and the multiuser access capabilities developed during this project.

Acknowledgements

We thank Nina Zumel, Chalie Ortiz, Ryan Blitstein and Myriam Ribiere for their contributions to the project. Nina contributed to the reorganization of the CMCP-98 knowledge base. Charlie helped formalize actions in the PQ ontology. Ryan developed the editing interface for the WWW version of the GKB-Editor. Myriam contributed to the solution of the CMCP-99 and taxonomy design. We will also like to thank Barbara Starr, Benjamin Good and David Easter at SAIC for their enthusiasm and support for our work.

REFERENCES

Chaudhri, V. K., A. Farquhar, et al. (1998). OKBC: A Programmatic Foundation for Knowledge Base Interoperability. Proceedings of the AAAI-98. Madison, WI.

Chaudhri, V. K., J. F. Thomere, et al. (2000). Using Prior Knowledge: Problems and Solutions. Proceedings of the AAAI-2000, Austin, TX.

Clark, P. and B. Porter (1997). Building Concept Representations from Components. National Conference on Artificial Intelligence.

Cohen, P., V. K. Chaudhri, et al. (1999). Does Prior Knowledge Facilitate the Development of Knowledge-based Systems. Proceedings of the AAAI-99**:** 221-226.

Forbus, K. (1984). "Qualitative Process Theory." Artificial Intelligence Journal **24**: 85-168.

Genesereth, M. R. and R. E. Fikes (1992). "Knowledge Interchange Format, Version 3.0 Reference Manual." (Logic-92-1).

Jermano, J. and J. Picarelli (1998). International System Framework. http://www.iet.com/Projects/HPKB/Intl_System.doc.

Karp, P. D., V. K. Chaudhri, et al. (1999). "A Collaborative Environment for Authoring Large Knowledge Bases." Journal of Intelligent Information Systems **13**: 155-94.

Knight, K. and S. Luk (August 1994). Building a Large-Scale Knowledge Base for Machine Translation. Proceedings of the National Conference on Artificial Intelligence. Seattle, WA.

Lenat, D. B. and R. V. Guha (1990). "Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project." : 336.

Lester, J. C. and B. W. Porter (1997). "Developing and Empirically Evaluating Robust Explanation Generators: The Knight Experiments." Computational Linguistics **23**(1): 65-101.

Levin, B. (1993). English Verb Classes and Alterations, University of Chicago Press.

Pease, A., V. K. Chaudhri, et al. (2000). Practical Knowledge Representation and DARPA's High Performance Knowledge Bases Project. Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning, Brekcenridge, Colorado.

Rickel, J. and B. Porter (1999). "Automated Model Composition." Artificial Intelligence Journal.

Schrag, R. (1998). HPKB Year 1 End-to-End Challenge Problem Specification. Information Extraction and Transport Corporation and Pacific Sierra Research. http://www.iet.com/Projects/HPKB/Y1Eval/Spec.

Schrag, R. (1999). HPKB Year 2 End-to-end Challenge Problem Specification. Information Extraction and Transport Corporation and Pacific Sierra Research. http://www.iet.com/Projects/HPKB/Y2/Y2-CM-CP.doc.

Stickel, M., R. Waldinger, et al. (June 1994). Deductive Composition of Astronomical Software from Subroutine Libraries. Proceedings of the Twelfth International Conference on Automated Deduction (CADE-12)**:** 341--355.