Knowledge Formation and Dialogue Using the KRAKEN Toolset

Kathy Panton, Pierluigi Miraglia, Nancy Salay, Robert C. Kahlert, David Baxter, Roland Reagan
Cycorp, Inc.
3721 Executive Center Drive
Austin, Texas 78731
{panton,miraglia,nancy,rck,baxter,roland}@cyc.com

Abstract

The KRAKEN toolset is a comprehensive interface for knowledge acquisition that operates in conjunction with the Cyc knowledge base. The KRAKEN system is designed to allow subject-matter experts to make meaningful additions to an existing knowledge base, without the benefit of training in the areas of artificial intelligence, ontology development, or logical representation. Users interact with KRAKEN via a natural-language interface, which translates back and forth between English and the KB's logical representation language. A variety of specialized tools are available to guide users through the process of creating new concepts, stating facts about those concepts, and querying the knowledge base. KRAKEN has undergone two independent performance evaluations. In this paper we describe the general structure and several of the features of KRAKEN, focussing on key aspects of its functionality in light of the specific knowledge-formation and acquisition challenges they are intended to address.

Introduction

The goal of the KRAKEN effort is to develop a set of tools to support knowledge base expansion. In particular, the KRAKEN system is designed to allow subject-matter experts (SMEs) to make meaningful additions to the Cyc knowledge base, without the benefit of training in the areas of artificial intelligence, ontology development and analysis (Gangemi et al. 2001), or logical representation languages.

To fulfill these requirements, KRAKEN relies principally on natural language interactions with the user. The interface's metaphor is that of a conversation between a subject-matter expert and a non-expert; both use ordinary English, possibly containing specialized vocabulary. Consequently, two major challenges faced the project designers at the outset:

- To make the user's conversation with KRAKEN as smooth as possible, which meant development and enhancement of every aspect of Cyc-based NL processing;
- To drive the knowledge formation process in such a way that user-entered knowledge would be at a high level of logical and representational quality.

The second challenge is in effect a consequence of working in a large, comprehensive knowledge base, using a powerful logical language such as CycL. The magnitude of the KB raises non-trivial issues of navigation through the concept space, traversal of the taxonomic hierarchies, and so on.

It is worth emphasizing that both challenges are indeed familiar to experienced ontologists and knowledge engineers. We chose not to attempt to simplify the KE task by artificially restricting the knowledge base and expressive capabilities of Cyc, but to take on the very challenging problem of creating a powerful "assistant" that would allow the SME to work with all the resources made available by Cyc (Cohen et al. 1999).

This paper is organized as follows. First, we provide a brief introduction to the Cyc Knowledge Base. Next, we introduce the natural language processing components used in KRAKEN. We then give an overview of the KRAKEN system, along with an in-depth look at a few selected tools. An example of a user dialogue with the system is presented. Finally, we discuss our results to date, and our plans for future refinements of the system.

The Cyc Knowledge Base

The Cyc KB is currently the largest general knowledge base in the world, housing roughly 1.4 million hand-entered rules interrelating 100k+ concepts. Concepts are denoted in the KB with Cyc constants; these may be individuals, intensionally defined collections, extensionally defined collections, relations that obtain between terms, or functions which can be used to refer to many more individuals without having to reify a term for each (e.g. "gander" is represented in the KB by the nonatomic term *(MaleFn Goose)*). Currently, the KB has knowledge of a wide range of topics, from microbiology to pop music, and has an extensive knowledge infrastructure, including multiple treatments of causality and temporal and modal reasoning. Knowledge is clustered into context-specific domains (or "microtheories") with epistemic access determined by specialised predicates. Consequently, the system has an ability to differentiate between and accommodate logically conflicting bodies of knowledge, including hypothetical and counterfactual contexts.

Over the last fifteen years, our representational language, CycL, has evolved—as needed—into a highly expressive

one, comparable to higher-order logic. The hurdle of combinatorial explosion in inference is overcome in two ways:

1. By using its partitioning of knowledge into contexts so that inferences occur within one small subset of the overall KB; and
2. By having a set of special-purpose inference modules that recognize commonly-occurring special cases and handle them efficiently.

The KRAKEN Natural Language Processing System

A crucial design feature of the KRAKEN system is its natural language interface. Primarily, the NLI was intended to allow users to interact with the system using simple English statements and queries. It was required to parse sentences and questions, and to generate English paraphrases for CycL statements. Furthermore, the interface had to be capable of immediately processing new lexical information corresponding to knowledge being entered by a user. For example, if a user introduced a new concept, such as *Clostridium-BotulinumToxin*, into the knowledge base, the system should prompt the user to supply natural language terms (such as "botulinum toxin" or "botulism toxin") that could refer to the new concept.

These design considerations, along with the need for quick parsing and generation, led us to create a system that combines features of principled, theory-driven approaches as well as more practical, less theoretical approaches. While we have, for example, decided against using an HPSG parser for this application, we still recognize the need for utilizing detailed lexical and syntactic knowledge. The KRAKEN natural language processing system consists of several subcomponents, including the lexicon, generation system, parsing system, and the iCycL (intermediate CycL) representation language, each of which is described briefly below.

The lexicon (Burns and Davis 1999) contains syntactic, semantic, and pragmatic information for about 27,000 English root words. Inflectional morphology is handled by a separate code component. Each root word is represented as a term in the KB, with assertions providing information about the word's part of speech, subcategorization patterns, and semantics. Semantic information in the lexicon provides a mapping between word senses and corresponding KB concepts or formulae.

The natural language generation system produces a word-, phrase-, or sentence-level paraphrase of KB concepts, rules, and queries. The NLG system relies on information contained in the lexicon, and is driven by generation templates stored in the KB. These templates are not solely string-based; they contain linguistic data that allows, for example, correct grammatical agreement to be generated. The NLG system is capable of providing two levels of paraphrase, depending on the demands of the application. One type of generated text is terse but potentially ambiguous, and the other is precise but potentially wordy and stilted. Through the KRAKEN Dictionary Assistant Tool, users can add parsing and generation templates for new terms as soon as they are introduced into the knowledge base.

Depth of parsing from natural language to CycL can range from very shallow (i.e. concept mapping) to deep (i.e. text understanding). For KRAKEN, deep interpretation is a requirement. However, earlier generations of parsing tools developed by our team, including an HPSG-based parser, proved much too slow to be used with KRAKEN. In order to balance demands of speed vs. depth, a hybrid top-down/bottom-up system was developed. This involves a template-matching parser for sentence-level parses, along with a chart parser and semantic templates for key sub-constituents (noun phrases and verb phrases).

Our natural language understanding system parses input strings into fully-formed semantic formulas. Design criteria for the parsing system included that it (1) be fast; (2) produce parses of adequate semantic detail; (3) ask the user for clarification only in cases where the system could not itself resolve ambiguities; and (4) support parsing into underspecified formulas, and then rely on some of the other KRAKEN tools, such as the Reformulator, to determine the best semantic translation.

The Text Processor controls the application of the various parsing subcomponents, using a heuristic best-first search mechanism that has information about the individual parsers, including their applicability to coarse syntactic categories and cost. This information is used to perform a syntax-driven search over the parse space, applying relevant parsers to the sub-constituents until all are resolved, or until the parsing options have been exhausted. The parsers at the disposal of the Text Processor are the Template parser, the Noun Compound parser, and the Phrase Structure parser.

The Template parser is essentially a string-matching mechanism driven by a set of templates compiled into an efficient internal format. These templates, like those used for generation, employ a simple format so that users can add templates as they are entering new knowledge into the system. The template parser is relatively fast, but is of limited flexibility. It tabulates semantic constraints during a parse, but does not attempt to verify them; that task is passed along to the next processing layer.

The Noun Compound parser uses a set of semantic templates combined with a generic chart-parsing approach to construct representations for noun compounds such as "anthrax vaccine stockpile". Unlike other parsing components, it makes heavy use of the Cyc ontology, and can therefore resolve many ambiguities that are impossible to handle on a purely syntactic level (e.g. "Mozart symphonies" vs. "Mozart expert").

The Phrase Structure parser takes a similar bottom-up approach to constructing parses. After completing a syntactic parse, it uses semantic constraints gleaned from the KB to perform pruning and to build the semantic representation. Specialized sub-parsers are used to parse noun phrases and verb phrases; resulting constituent parses are combined to produce a complete semantic translation.

In order for parsing to be successful in the current application, some decisions about semantic meaning needed to be deferred. In particular, radically vague or underspecified words such as "is" or "contains", which can map onto many distinct relations in the KB, introduce ambiguities which are

not handled well by producing all possible interpretations in parallel. To deal with such cases, strings are parsed into an intermediate CycL (iCycL) layer that conflates relevant ambiguities into a single parse, by using very general predicates such as is-Underspecified. Another level of processing reformulates iCycL representations into final, more specific CycL representations, often with the user's help.

In addition to handling underspecification, the iCycL layer is also well-suited for other types of semantic processing, such as interpretation of quantification and negation, and type-shifting. The interpretation of quantifiers, for example, consists in a transformation from iCycL expressions into CycL logical forms performed by a dedicated "reformulation" component. Although CycL representations are modelled on first-order logic, the language itself allows the definition of higher-order constants. We exploit this capability to represent a wide range of NL quantifiers (*most, many, few, no*, etc.) formally as generalized quantifiers, i.e., as higher-order relations between collections.

Overview of KRAKEN

The KRAKEN system consists of an integrated set of tools for adding to the Cyc KB, accessed via an HTML interface. The tools already deployed include:

- Creators, selectors, and modifiers for all of the categories distinguished by the Cyc system (concepts, statements and rules, predicates) and the categories identified by analysis of the KE process (scenarios and queries).
- Tools for determining the quality and consistency of the statements made to the knowledge base, i.e. checkers for contradiction and redundancy, precision manipulators for improving generality or specificity, and tools for providing feedback on the quality of the KE done in terms of rule and query critiquers.
- Tools that leverage existing knowledge to elicit new knowledge, such as the Concept Differentiator, the Analogy Developer, and, for ensuring breadth, the Salient Descriptor.
- Lexifiers, such as the Dictionary Assistant, that allow users to enter natural-language words and phrases for concepts they are describing.

The UIA: the User Interface to KRAKEN

The User Interaction Agenda (UIA) interface is a message-based HTML system that uses the HTML REFRESH capabilities of the browser to simulate real-time updating. It consists of four areas on the user's screen:

1. A menu of tools that is organized according to the recommended steps of the KE process (i.e. browsing what is there, creating what is missing, testing it via rules, finalizing the information, and debugging).
2. A type-in box that sends the entered text or query to KRAKEN's NL processing system, to interpret the text as either a command for an action, a question to be run as a query, or a sentence to be parsed as a new statement of fact.
3. An interactive center pane that is used as the render space for the currently active tool. Most of the user and system interaction takes place here, from the selection of initial topics, to clarification dialogues in which KRAKEN asks the user to select among possible interpretations of an ambiguous utterance.
4. An agenda summary pane, where the essential steps to completing an interaction are displayed. These steps are color-coded to indicate whether the action in question is currently possible, or is blocked.

Specialized KRAKEN Knowledge Entry Tools

The process of entering or modifying knowledge in Cyc typically consists of several steps:

1. Creating basic concepts: types (collections), individuals, predicates, and so on.
2. Classifying concepts at the appropriate level of generality, by placing them in proper taxonomic relationships to existing concepts.
3. Identifying relations and slots applicable to the new concepts.
4. Formulating rules establishing the proper use (and thus expressing the meaning) of such concepts in reasoning.

Normally these tasks are performed by knowledge engineers and ontologists. The intended user of KRAKEN, however, is untrained in these areas. For this reason, KRAKEN includes a series of tools and assistants designed to ensure the quality of the knowledge acquired. These tools operate essentially by eliciting new knowledge from the user, or querying the user about optional additions (automatically generated) to the KB.

Given space limitations, we can describe here only a few among the tools that are currently implemented:

Precision Suggestor

Whenever a new concept is created, it must find its place in the taxonomic structure of the existing KB. Considering the size of the Cyc ontological hierarchy, this is of course much more easily said than done.

The Precision Suggestor helps the user place new concepts at the appropriate level of generality/specificity in the ontology. After the user has entered a concept and formulated a basic or "initial" fact about it (e.g., *Rudy Giuliani is a person* or *Botulin is a kind of toxin*), the Suggestor identifies a suitable number of possible generalizations and specializations that might be suggested for the concept entered. It then queries the user whether any of the suggestions would lead to a more accurate statement than the one originally entered. These suggestions are heuristically determined from the current state of the KB. The selection can be tuned by Cycorp ontologists using specific "KE facilitation" flags in the KB.

Salient Descriptor

A more complex task is to ensure both *breadth* and *depth* of knowledge representation. This requirement might be expressed by the question: once a concept is added, what is

the minimal set of features (properties, facts, rules, . . .) that should be specified about it?

Put in different terms, a robust (sufficiently broad and deep) representation should include the *salient* features of the concept. The problem of course is that salience is heavily context-dependent. The Salient Descriptor is the KRAKEN component that queries the user with suggestions about additional knowledge; it identifies such knowledge as salient on the basis of special "facilitation" rules in the KB. One such rule, for instance, might be that if X is a type of biological organ, then it makes sense to specify the biological function of X; or that if X is a particular eating event, it makes sense to specify what food was consumed in it, or who (what animal) took part in it, and so on.

On this basis, KRAKEN can query the user with additional clauses and facts that are *prima facie* salient with respect to the concept(s) the user has entered. Note, furthermore, that rules such as the ones above are quite general, because they ultimately derive from conceptual analysis. One side benefit of Cyc is that it contains the accumulated results of years of ontological analysis of ordinary concepts. In effect, the Salient Descriptor leverages this accumulated patrimony in order to foster better knowledge formation practices on the part of subject-matter experts.

Process Descriptor

The representation of *processes* as complexes of events with multiple participants playing various roles receives special focus in many domains of knowledge representation. It has figured prominently in the microbiology and biochemistry domains in which KRAKEN has been tested by SMEs. On the other hand, the representation of processes as reified "objects" in an ontology is a complex task, generally presupposing a certain degree of logical sophistication on the part of the knowledge enterer (consider, for instance, the problems of constraining and preserving the identity of participants throughout the unfolding of a complex event, well described in (Aitken 2001a)).

Cyc has a very rich vocabulary for describing processes as complex events or "scripts." A script is an event constituted by a temporally ordered sequence of subevents. Both scripts and their individual subevents are related to things and individuals playing some role in them (agent, patient, etc.) by "actor slots." Additional relations specify the temporal ordering of subevents in a script. Furthermore, the expressive power of CycL is used by casting much of the relevant vocabulary at the type level. Thus, instead of asserting that an individual event of type *Paying* follows an individual event of type *OrderingFood* in a typical instance of the script *GoingOutToEat* (at least in the context of *UnitedStatesSocialLifeMt*) by a complex rule, we can state succinctly

(startsAfterEndOfInScript Paying OrderingFood).

The type-level description, however, requires that specific constraints be added about the identity of the participants: it's easy to assert, in type-level vocabulary, that the agent in the *Paying* subevent type, as well as the agent in *OrderingFood* is a person, but less obvious how to express that the same agent is involved in both the paying and the ordering event (in this example, actually, the precise constraint is even more complex than this: most times there are multiple agents ordering food in a particular restaurant visit, but not all of them need be the payers). In any case, it seems clear that the KRAKEN user would need to be guided through the proper steps.

This guidance is provided by a separate process description tool. It allows the user to construct the description of a process, specifying the participants by roles (at the type level, we indicate what is true of an entire class of complex or simple events), constraining their identity through the main process and its subevents, and articulating the temporal structure of the process, namely its subevents (by type) and their relative ordering within the main process.

Example of an Interaction with KRAKEN

The user begins an interaction with KRAKEN by providing his name to the system, and selecting a topic to converse about. Suppose that a user wants to teach the system about a new disease. The user has multiple means for determining whether a concept is already known to Cyc. For one, he can use the Concept Browser to navigate the ontology. For another, he can type a question such as "What do you know about Endemic Relapsing Fever?" into the interaction box. Having determined that Cyc does not yet know anything about this disease, he types into the interaction box:

Endemic Relapsing Fever is an infectious disease.

KRAKEN parses the input sentence, but does not recognize the name of the new disease. Thus, only a partial CycL representation can be created at this point. KRAKEN responds:

I do not know what *Endemic Relapsing Fever* means in this context.

[Describe now] [Search KB] [Describe later] [Continue] [Forget it]

The user selects the [Describe now] button. KRAKEN asks the user what "Endemic Relapsing Fever" is a more specific kind of, and the user responds with "infection". KRAKEN also supplies a box in which the user can enter a concept similar to Endemic Relapsing Fever, but which is already known to Cyc. The user opts to do so, entering "Rocky Mountain Spotted Fever" as a similar concept.

Since the Noun Phrase parser finds two potentially relevant meanings for "infection", KRAKEN asks for clarification of its meaning: whether it refers to an ailment, or to the process of contaminating something. The user selects the first interpretation.

The Precision Suggestor tool is then automatically invoked, to ensure that the new knowledge is presented at the appropriate level of specificity. KRAKEN asks:

In the sentence
Endemic Relapsing Fever is a kind of infection
could you replace the phrase *infection* with any of the following?

bacterial infection
viral infection

fever infection
fungal infection
parasitic infection
inhalational infection
[...]

The user, knowing that Endemic Relapsing Fever is a febrile illness that is caused by a bacterium, selects "bacterial infection" and "fever infection".

Having been supplied with a basic definition of "Endemic Relapsing Fever", KRAKEN can fully parse the user's original statement into a CycL formula. KRAKEN now invokes the Analogy Developer tool. This tool selects relevant statements about Rocky Mountain Spotted Fever, which the user claimed was similar to Endemic Relapsing Fever, and allows the user to modify those statements to fit the concept he is describing. KRAKEN scans the KB for facts about Rocky Mountain Spotted Fever. For each relevant CycL assertion, an English paraphrase is generated. These sentences are then presented to the user:

Which of the following things I know about the disease Rocky Mountain Spotted Fever are also true of Endemic Relapsing Fever?

Only R. ricketsii causes cases of Endemic Relapsing Fever

People typically acquire Endemic Relapsing Fever through biting by a tick

The risk of mortality for people afflicted with Endemic Relapsing Fever is 0.2

The incubation period for Endemic Relapsing Fever in people is usually between 3 and 14 days

There is no risk of Endemic Relapsing Fever spreading directly from people to people

The user edits the first statement to refer to Borellia recurrentis, the bacterium that causes the fever. The second statement is true as is; Endemic Relapsing Fever is indeed tick-borne. The user does not know what the mortality rate and incubation periods for this infection are, so he does not select the next two statements. Finally, the last statement is correct as is. The user then submits this edited page of assertions for processing. The Salient Descriptor tool is then activated, prompting the user to give more facts about the new disease. It states:

Yes/No Question

Is the following sentence true?
Endemic Relapsing Fever has distinct stages.

The user selects "Yes", and the Salient Descriptor continues:

Here is one way in which you are encouraged to refine Endemic Relapsing Fever :

Please enter something to fill in the blank(s) in the sentence
Endemic Relapsing Fever primarily afflicts _____.
with something that is a kind of organism part.

The user enters "nervous system". Another question is then posed by KRAKEN:

Yes/No Question

Is the following sentence true?
The different stages of an instantiation of Endemic Relapsing Fever do not involve different bacteria.

The user selects Yes, since only B. recurrentis is implicated in this disease. The Salient Decriptor then prompts:

Please enter something to fill in the blank(s) in the sentence
In Endemic Relapsing Fever, fever is likely to have a onset rate.
with something that is a rate of change attribute.

Examples:
gradual
moderate
sudden

The user selects "sudden". KRAKEN processes this choice, and the initial teaching session regarding Endemic Relapsing Fever comes to an end. At this point, the user can ask KRAKEN to summarize what it has learned, by typing "What do you know about Endemic Relapsing Fever?" into the interaction box. KRAKEN collects its new knowledge about the disease, and generates English paraphrases:

Here's everything I've been told about Endemic Relapsing Fever :

[+] Endemic Relapsing Fever is a type of infection.
[+] Endemic Relapsing Fever is an infectious disease.
[+] Endemic Relapsing Fever has distinct stages.
[+] A case of Endemic Relapsing Fever is a kind of fever infection.
[+] A case of Endemic Relapsing Fever is a kind of bacterial infection.
[+] Only B. recurrentis causes cases of Endemic Relapsing Fever.
[+] Ticks are an infection vector of Endemic Relapsing Fever.
[+] Endemic Relapsing Fever is similar to the disease Rocky Mountain Spotted Fever.
[+] Endemic Relapsing Fever primarily afflicts nervous systems.
[+] The different stages of an instantiation of Endemic Relapsing Fever do not involve different bacteria.
[+] In Endemic Relapsing Fever, fever is likely to have a sudden onset rate.
[+] People typically acquire Endemic Relapsing Fever through biting by a tick.
[+] There is no risk of Endemic Relapsing Fever spreading directly from people to people.

Results and Future Plans

KRAKEN's performance was evaluated in the Summer of 2001 in an independent experiment (results for which will be published elsewhere). A team of SMEs unfamiliar with Cyc used the system to enter new knowledge in the domain

of cell biology. The system's responses were rated on a 0-3 point scale in terms of correctness, quality of representation, and quality of explanation. KRAKEN's average score in the cell biology domain was 1.91 (with scores as high as 2.54 on some sections of the experiment). These results, especially since they were produced with a very early version of the system, are encouraging.

A further evaluation of KRAKEN in January, 2002 showed that knowledge entry rates of the SMEs had increased significantly, compared to the evaluation six months earlier. Improvements in tools at this stage allowed users to more rapidly create knowledge, and led users through more complex entry sequences. Most importantly, perhaps, the SMEs had caught up to the logic experts in terms of quality of representation.

From a qualitative standpoint, the KB segments produced by the SMEs using KRAKEN warranted positive marks in many areas. In some respects, the KRAKEN products compared favorably to standard practice by Cycorp knowledge engineers. Assertions were mostly introduced at the proper level of generality and specificity; logical and conceptual soundness seemed generally respected. Overall, these seem to have been the most successful aspects:

- Taxonomic relations: by and large, concepts are defined at the right level of generality and the relations of generalization and specialization between collections are well-chosen and informative.
- Articulation of events and processes in subevents, and specification of participants' roles in same (users seemed quite capable of navigating Cyc's vast stock of role-representing relations).
- Ability to insert "exception" statements (e.g.: not every transcription has a nucleolytic proofreading as subevent), by using "natural" assertions involving ordinary quantification.

Aspects that seemed less successfully developed:

- Guiding the user through stating the temporal ordering of subevents in complex events (i.e., "scripts").
- Use of complex predicates, especially to describe similarities and differences between concepts and entities; and creation of new predicates.

We believe that developments both in KRAKEN design and further engineering of the ontology should soon improve performance in these problem areas.

In the coming months, along with improvements to tools like the Process Descriptor and Analogy Developer, larger and more ambitious versions of the User Modeller will be designed. We would like to be able to use context to realize when a term can be used profitably in conversation, to determine saliency of a term description given the context, and to organize some of the answers returned by the system.

The bigger issues in the background remain challenging. In order to achieve optimal performance in User Modelling, the KRAKEN system would have to introduce an abstraction layer between itself and the SME, just as a doctor takes the layman's description of an illness and translates it opaquely into a scientific medical description without ever confronting the patient with the medical terminology. Much ontological engineering research and work will have to be done to reach this long-term goal, but there are some gains which can be made in the shorter term.

A number of refinements to the NL system are planned for the coming months. For generation, the system will move from a focus on generating isolated sentences and questions to rendering multi-sentence-level text. We will be incorporating indexicals, pronouns, and other intersentential anaphors in order to create natural-sounding interactions. Future work on the parsers will emphasize increasing their speed, and on extending coverage to include more complex structures, such as adjectival and adverbial phrases.

Acknowledgements

KRAKEN is being built as part of DARPA's Rapid Knowledge Formation (RKF) project (DARPA, 2000). The authors are indebted to the work and contributions of the Knowledge Formation and Dialogue group at Cycorp: Michael Witbrock, Dave Schneider, Keith Goolsbey, Jon Curtis, John Jantos, Matt Smith, Matthew Olken, Doug Foxvog, Fred Hoyt, Jennifer Sullivan, Kim Loika, Bjørn Aldag, Stefano Bertolo, Peter Wagner, Ben Rode and Michael Wakoff.

References

Aitken, S. 2001. Participants, Conditions and Identity in Scripts. Technical Report, Artificial Intelligence Applications Institute, University of Edinburgh.

Burns, K. J., and Davis, A. R. 1999. Building and maintaining a semantically adequate lexicon using Cyc. In Viegas, E. ed. *Breadth and depth of semantic lexicons*. Dordrecht: Kluwer.

Cohen, P., Chaudri, V., Pease, A., and Schrag, R. 1999. Does prior knowledge facilitate the development of knowledge-based systems? *Proceedings of the AAAI-99,* pp. 221-226. Menlo Park, CA.

DARPA. The Rapid Knowledge Formation Project (main website). http://reliant.teknowledge.com/RKF/, 2000.

Gangemi, A., Guarino, N., Masolo, C., and Oltramari, A. 2001. Understanding top-level ontological distinctions. To appear in *Proceedings of IJCAI 2001 Workshop on Ontologies and Information Sharing.*