Enabling Domain Experts to Convey Questions to a Machine: A Modified, Template-Based Approach

Peter Clark
Mathematics and
Computing Technology
Boeing Phantom Works
Seattle, WA 98124

Vinay Chaudhri
Sunil Mishra
Jerome Thomere
SRI International
Menlo Park, CA 94025

Ken Barker
Bruce Porter
Computer Science
Univ. Texas
Austin, TX 78712

ABSTRACT

In order for a knowledge capture system to be effective, it needs to not only acquire general domain knowledge from experts, but also capture the specific problem-solving scenarios and questions which those experts are interested in solving using that knowledge. For some tasks, this latter aspect of knowledge capture is straightforward. In other cases, in particular for systems aimed at a wide variety of tasks, the question-posing aspect of knowledge capture can be a challenge in its own right. In this paper, we present the approach we have developed to address this challenge, based on the creation of a catalog of domain-independent question types and the extension of question template methods with graphical tools. Our goal was that domain experts could directly convey complex questions to a machine, in a form which it could then reason with. We evaluated the resulting system over several weeks, and in this paper we report some important lessons learned from this evaluation, revealing several interesting strengths and weaknesses of the approach.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Graphical User Interfaces, Interaction Styles; I.2 [**Artificial Intelligence**]:

General Terms

Algorithms, Human Factors

Keywords

Question formulation, question answering

1. INTRODUCTION

In order for a knowledge capture system to be effective, it needs to not only acquire general domain knowledge from experts, but also capture the specific problem-solving scenarios and questions which those experts are interested in solving using that knowledge. For some tasks, this latter aspect of knowledge capture is straightforward, in particular when the system is essentially targeted at answering a single (possibly complex) question, e.g., "What drug regimen should this patient be given?". In these cases, the question-answering scenario is well-defined, and tools for communicating and querying those scenarios can be constructed straightforwardly (e.g., a spreadsheet to enter patient data in). In other cases, however, in particular when the space of questions cannot be fully anticipated, e.g., in a tutoring environment, the question-posing aspect of knowledge capture can be a challenge in its own right.

In this paper we present and evaluate the approach we have developed to deal with this task, arising out of a larger project to also acquire the formal domain knowledge itself from domain experts [3, 14]. This task is challenging because we wish the system to support answering a wide variety of questions, including longer questions which may include scenario descriptions, without requiring users to formulate their queries in predicate logic directly. Our solution is based on the creation of a catalog of domain-independent question types, each twinned with a question template and question-answering method. Unlike earlier catalogs of question types, e.g., [9, 6, 10], our questions are oriented around the modeling and reasoning paradigms which the underlying knowledge base (KB) supports, rather than the question word (who, what, where, etc.). We extend the question template approach to include graphical tools for scenario description, enabling more complex questions to be posed and a notion of a question context to be created, and allowing menu options to the user to be sensibly restricted.

Our focus in this paper is on the mechanisms by which a domain expert can communicate his/her questions to the system, rather than the details of the reasoning used within the KB itself to compute the answers. The significant contributions of this work are threefold: First, our catalog of domain-independent question templates, developed by extensive analysis of several hundred text-book style questions, provides a useful starting point (and/or point of comparison) for others

K-CAP'03, October 23–25, 2003, Sanibel Island, Florida, USA.

with similar goals. Second, by extending a "question template" approach with graphical tools, we are able to extend the scope of questions which can be asked. Third, we report several important lessons learned from an extensive evaluation, in which several surprising strengths and weaknesses of the approach were revealed.

2. APPROACHES TO CONVEYING QUESTIONS TO A MACHINE

AI applications vary tremendously in the scope of questions which they are intended to deal with, and the degree of formal reasoning required to answer them. Early expert systems, e.g., Mycin [12], were often specifically designed to answer just one or a few domain-specific question types (e.g., "What disease does this patient have?"), and as a result of this predictability, the entire system could be designed to allow the question to be posed and the answer to be computed.

More recently, there has been significant advances in question-answering technology in the information retrieval world, e.g., [7, 5]. The focus of that work has primarily been on answering questions by finding the answer in pre-written text, rather than through inference. Typically questions are posed in natural language, and answered by first using word search techniques to identify candidate sentences/paragraphs which may contain the answer, followed by more extensive natural language processing (NLP) on those candidates to seek the answer word(s) in a sentence. This approach works well provided that the answer is explicitly stated in the text, and that the question is short, generic (i.e., not about some novel scenario), and requires just a one/few word answer. Because the answer method is based on information retrieval, rather than inference, it is not necessary to generate a full formal logic rendition of the question from its natural language form.

However, to go beyond this to questions requiring inference, new challenges are raised. While theorem-proving/knowledge-based systems can perform such inference, the challenge of interest here is to allow end users to convey questions to such systems without those users having to learn the underlying logic language. One approach is to allow users to enter questions in natural language and then convert them to logic automatically, but the unconstrained nature of language make this approach challenging. The only significant successes of this approach have been in querying databases, e.g., Microsoft English Query [11], where the scope of questions to be handled, the subject domain, and the method of answering queries are well constrained. Even then, the lack of constraint on the user can be problematic [2].

An alternative is the use of templates, where users can "fill in the blanks" of pre-written questions. This has been successful in earlier projects, e.g., DARPA's HPKB project [4], but the templates have been domain-specific (e.g., "What risks/rewards would <country/group> face/expect in taking hostages citizens of <country>?"), and thus suffers from the opposite problems of NLP, namely being too constrained (although active menu techniques, e.g., NLMenu [13], can start to ease this problem). Our challenge is to achieve something more general purpose, without losing the ease of translating to machine-sensible form which templates provide.

3. CONVEYING QUESTIONS

To highlight the challenges of using "question templates" for communicating questions to a machine, consider the following question, taken from the test suite we have been working with [8]:

- I-7-208 "When during RNA translation is the movement of a tRNA molecule from the A- to the P-site of a ribosome thought to occur?"

Creating a template which will accommodate this question presents two challenges:

- This question's structure is idiosyncractic
- The number of "menu options" to offer the user for each parameter is impractically large

A naive rendition of this question as a template might be: "When during <A> is the <B> of <C> from <D> to the <E> of a <F> thought to <G>?", but clearly this is not a sensible approach, as the system will need impractically many such templates for broad coverage. We could reduce the number of template parameters by allowing them to include complex objects, e.g., replacing "... to the <E> of a <F>..." with "...to the <EF>..." and allow the parameter <EF> to include "the P- site of a ribosome", but the result will be fewer menus with exponentially (and impractically) more options to select from. We address these concerns in two ways:

1. We assume the question is being posed in some *scenario of interest*, which can be formally represented. This scenario can then be used to bound the range of objects, including complex objects, which are potentially of interest to the user, and hence potential values of template parameters. As a result, a simpler template becomes feasible to use.

2. We require the user to re-formulate, or *coerce*, his/her specific question into a smaller, fixed set of question types known to the system. The challenge here is to then design that smaller set of templates which adequately cover the space of problems which the user is likely to deal with.

3.1 1. The Scenario of Interest

A question can be viewed as containing two parts: A *scenario specification*, namely the thing the question is about, and the actual query itself, posed to that scenario. In our analysis of textbook questions, multi-sentence questions are mostly scenario specification ("Imagine that the..."), with a shorter, specific query at the end ("Given this, what would ...?").

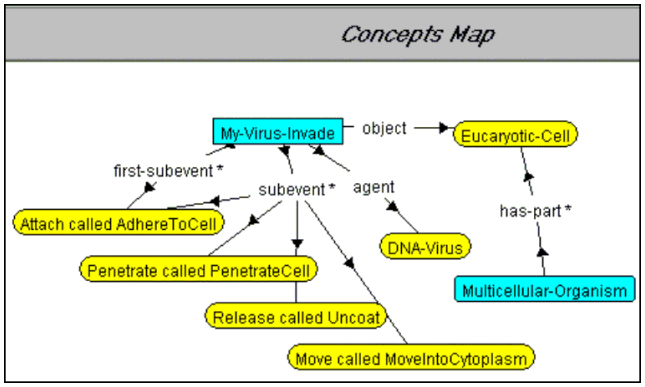

Figure 1: The user specifies a question scenario by giving it a name (here, My-Virus-Invade), and then graphically assembling the various involved objects in the appropriate configuration (here a DNA virus invading the cell of a multicellular oganism). Menus and browsing tools constrain the user to use terms and relations from the ontology of the system's knowledge base.

Rather than creating templates to accommodate the entire question, including the scenario specification, we have developed an approach where the user uses a graphical tool to enter the scenario, and then selects a question from a set of question templates posed to that scenario. By factoring out the scenario information from the query, the range of templates required is significantly reduced, and can be feasibly enumerated.

A scenario is a set of instances (individuals) and ground facts about them, with one particular instance being the "root" of the scenario. To specify the scenario graphically, the user creates a graph on the screen in which nodes denote (instances of) the objects involved, and arcs denote the relationships involved. The interface tools ensure that the object types and relations are selected from the KB's ontology, and hence that the assertions entered are meaningful to the computer. For example, if the scenario concerned the behavior of a DNA virus invading the cell of a multicellular organism, the expert would first specify that question scenario graphically by selecting and connecting objects and graphs denoting a DNA virus, invasion of a cell, multicellular organisms, etc. An illustration of this interface is shown in Figure 1. (Menus and browsing tools constrain the user to use terms and relations from the ontology of the system's knowledge base). Behind the interface, the resulting graph is translated into a set of ground assertions which can then be reasoned about by the system. A query is then posed to a scenario by selecting one of a number of question templates, and selecting values for its parameters using pull-down menus. Queries about general concepts are handled by creating a "scenario" containing a single instance of that general concept, and then posing a question to that instance.

Given the scenario, we can bound the possible values for parameters to just objects contained within that scenario. The notion of "contained within" goes beyond the objects the user explicitly placed in his/her graph, to include additional objects which can be inferred by the KB to be in the scenario also. This set of objects is defined by a simple, recursive procedure: Starting with the scenario's root instance, if it is an event then query the KB for that event's actors (agent, object, location, instrument, etc.) and its subevents; if it is a physical object, then query the KB for its main physical properties (parts, location, is-part-of, etc.); otherwise do nothing. The objects returned are thus a mixture of the ones in the original graph, and additional objects inferred using rules in the KB. For example, the system will infer that the cell in Figure 1 contains a nucleus, and thus include the cell's nucleus in the returned list of scenario objects, even though it is not explicitly in the original graph. This procedure is applied recursively to all the objects thus found. This process is guaranteed to terminate because (from constraints in the KB) both the "subevent" and "parts" hierarchies are necessarily finite and non-cyclic, i.e., an event/object will never be the subevent/subpart of itself. (The scenario graph itself, of course, may include cycles). The objects thus found are potential parameter values for the question templates. Note that these objects may be "complex objects", i.e., not just instances of a class, but instances in a particular relationships to other instances (e.g., "the sigma factor attached to the polymerase"; "the P- binding site of the ribosome"; and in Figure 1 "the virus invading the cell"). In this way, possible values for question parameters can be derived from the context in which the question is being asked. As a result, the options presented to the user are generally both manageable in number, and meaningful in the context of the question. For cases where a generic class name (rather than a scenario participant) is required in the template, an ontology browser is launched for the user to select from.

3.2 2. Question Reformulation

As the user is restricted to a small set of question types, he/she may need to perform some work recasting his/her original question in terms of those types. In some cases, this reformulation is a minor rewording, but in others, the gap can be larger. For example, the earlier question (I-7-208) would be recast to fit template q17: "During RNA translation, when does the tRNA molecule move to the P-site of the ribosome?".

4. A CATALOG OF QUESTION TYPES

Each question template is twinned with an answer procedure specifying the information required in the answer, the order in which it should be presented, and how it should be formatted. To create the catalog of question types and their associated answer procedures, we performed an extensive analysis of 339 questions about cell biology, drawn from two text books and a set provided by the company IET Inc. It became clear during this analysis that the natural grouping of these questions was not by the question word (who/what/why), but by the underlying "modeling paradigm" (i.e., style of representation and reasoning) which seemed appropriate for answering the question. That is, for each particular question,

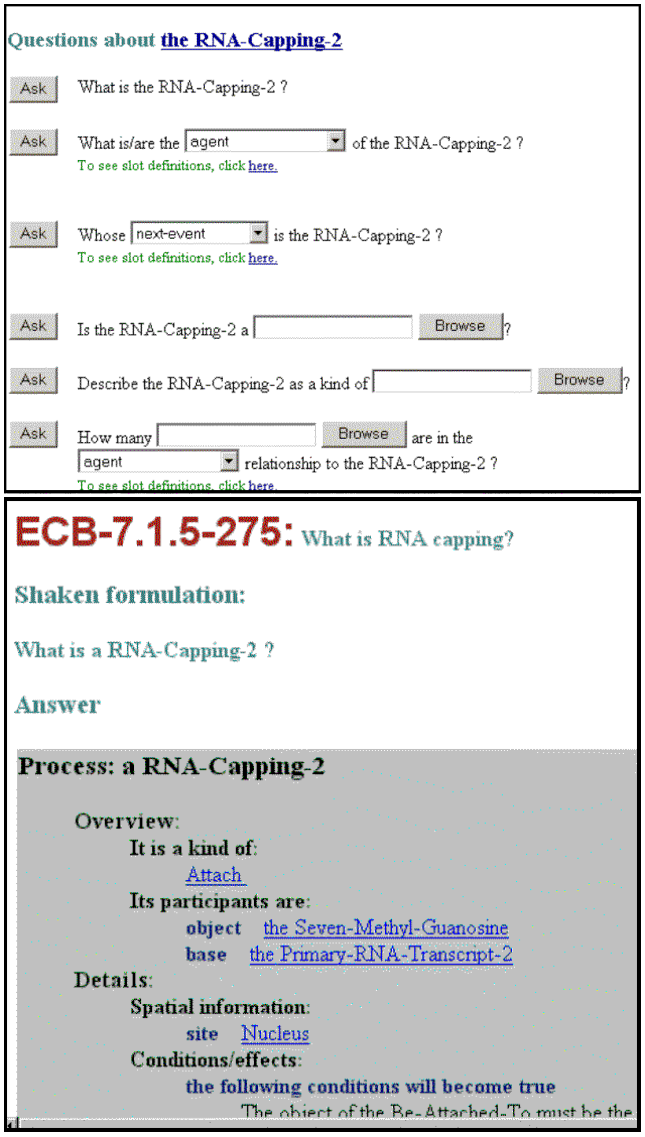

Figure 2: Two screendumps of the question-answering interface, the first showing some of the question templates and the second showing the machine-generated answer to the question "What is RNA capping?".

we could identify that *if* a particular style of representation was used (and the appropriate domain knowledge was encoded in it) *then* the question would be answerable. We identified four such paradigms, and for each created a set of templates reflecting the question types which it could (in principle) answer. Examples are given below, and the full catalog is given in Appendix A:

1. Deductive Reasoning :

Many simple "fact-finding" questions can be answered by fact lookup and/or simple deductive reasoning on those facts. Some examples of such questions (re-expressed using our templates) are:

- I-7-25 "What are the functions of RNA?"
- A-7-8D-a "Is a ribosome a cytoplasmic organelle?"
- A-7-8D-b "How many membranes are in the parts relationship to the ribosome?" (i.e., "How many membranes does a ribosome have?")

2. Discrete Event Simulation :

Some questions require reasoning about change in the world. One paradigm for doing this is discrete event simulation, in which a process is modeled as a sequence of discrete events, and simulation involves "executing" these events to observe how the world changes. We support this modeling paradigm using situation calculus and STRIPS action rules. Some questions answerable using this modeling paradigm are (again expressed using our templates):

- I-7-47 "What happens to the DNA during RNA transcription?"
- A-7-11A "During protein synthesis, after codon bonding, what is the type of the bond between the mRNA codon and tRNA anticodon?"

3. Qualitative Reasoning :

Another way of modeling change in the world is through qualitative influence diagrams, specifying how parameters influence each other. This paradigm accommodates knowledge of continuous change, and supports questions such as:

- I-7-136 "In a cell, what factors affect the rate of protein production?"
- I-7-106 "In RNA transcription, what factors might cause the transcription rate to increase?"

4. Analogical/Comparative Reasoning :

A fourth style of question-answering involves reasoning about similarities/differences between objects. As this is substantially different from the logic-based reasoning categories, we place it in a separate category. An example question from this category is:

- I-7.1.5-288 "What is the difference between procaryotic mRNA and eucaryotic mRNA?"

The complete catalog of question types is given in Appendix A. We do not claim it gives full coverage of all possible questions; rather, it appeared to give adequate coverage of the large, varied set of biology text-book questions that we examined.

5. EVALUATION AND DISCUSSION

To evaluate the system (including other aspects not described in this paper), we conducted an extensive trial with 4 users over a 4 week period. Our goals concerning the question-posing technology were to assess both its coverage and usability by domain experts. The four users were experts in biology (three graduate students, one undergraduate) rather than computing, and they were each given the task of, over the 4 week period, formally encoding an 11-page subsection of a biology textbook using tools described elsewhere

User	Qns attempted (out of 70)	Av. correctness (on qns attempted)	(on all qns)
1	51	2.29	1.67
2	70	2.44	2.44
3	44	2.11	1.32
4	61	2.07	1.80
average	**56.5**	**2.23**	**1.81**

Table 1: Users were able to pose most questions and receive adequate answers (scored on a 0-3 scale).

User	Uses of Question Template:									
	q1	q2	q3	q4	q5	q6	q11	q15	q19	q29
1	18	20	5	1	0	2	8	0	0	2
2	34	45	10	0	1	0	12	0	4	2
3	30	29	0	0	0	0	0	0	1	2
4	18	35	2	1	5	3	5	1	5	6

Table 2: Frequency of template use. Templates q7, q9, q12, q13, q14, q16, q17, and q18 were also available at the time of the trials but never used. Sometimes users used multiple templates to answer a single question.

[14], and then testing their representations using a set of 70 test questions. The test questions were set by an independent contractor (IET Inc) and without knowledge of our templates. They were expressed in English, and were similar in style to the examples in the previous Section. They were approximately high-school level difficulty, and were mainly "reading comprehension" type questions, although a few required more complex inference and simulation. The system's answers were scored by a fifth biology expert on a 0-3 scale (0 = completely incorrect, 1 = mostly incorrect, 2 = mostly correct, 3 = completely correct). At the time of the trial, we had implemented 18 of the 29 templates and answer procedures (see Table 2). The four qualitative reasoning templates were not implemented and thus not part of the evaluation.

Most significantly, all four users were able to use the interface to pose most (80%, see Table 1) of the test questions to the system, and the procedures associated with the question templates generally returned acceptable answers. Of the few unasked questions, only some were unasked because the user could not work out how to express them in terms of the templates; others were unasked because the user could not work out how to encode the domain knowledge required, or ran out of time. On the 0-3 scale listed above, the answers scored well, averaging 2.23 (2 = mostly correct) on the questions actually posed to the system. These results are important achievements for this approach, and suggest that this is a viable way for users to pose questions and get good, inference-derived answers, without having to formulate their queries directly in logic.

Despite this, there are several significant challenges which became apparent during the evaluation. First, although in principle (from our analysis) many questions can be rephrased to fit a template, in practice this turned out to be a challenging task for the users, and they often resorted to answering specific questions (e.g., ECB-7.1.5-181 "What happens to an RNA primary transcript before it leaves the nucleus?") using the general descriptive template q1 (e.g., "What is RNA transcription?"). Table 2 shows the frequency of use of each template, and indicates that the simpler, general templates were heavily used, including heavy use of the general template q1 "What is <object>?". The question reformulation task is indeed difficult, and is not just a linguistic rewording task; rather, it also requires viewing the scenario in terms of one of the KB's modeling paradigms, e.g., thinking about RNA transcription as a sequence of discrete events connecting "snapshots" (situations) of the process. This task was easier for us to do during our analysis of questions than for the users to do during the trials, probably reflecting the advantage of our own background in knowledge representation. Additional training for users in how to maximize usage of the templates would help.

A second observation from the trials is the need for the use of some kind of "viewpoint" mechanism. A biological process can be viewed in different ways, by ignoring or paying attention to certain events, and by viewing objects and events at different levels of detail, and often question-answering requires selecting the appropriate viewpoint to use. However, our underlying knowledge base technology assumes a single, universal representation of the world, and currently has no viewpoint mechanism built in. As a result, users sometimes created multiple representations for the same concept (using slightly different concept names) to encode different viewpoints, and hence enable the KB to generate viewpoint-specific answers. A more principled approach to this, e.g., [1], would be desirable.

A third issue that arose is that our templates all assume that the user has already created/selected the scenario to pose a question to, but in practice sometimes identifying the appropriate scenario is actually part of the question. For example, the question ECB-7.1.3-94 "What kinds of final products result from mRNA?" is not posed to a particular scenario, but rather requires *finding* all scenarios involving mRNA and analyzing those. Our templates will not accommodate this question. We are currently adding new templates for "scenario searching" like this.

Fourth, some of the answer procedures, although producing competent answers, could be improved. In particular, they are lacking an ability to distinguish "significant" or "relevant" facts. For example, the procedure for template q17 "When does <X> occur?" returns the events immediately before and after X, even if those events are not particularly important or significant with respect to X. A better answer would refer to events which cause and are caused by X. Similarly, analogical question-answering (q29) is currently not

able to distinguish incidental facts (e.g., name, position in space) from functionally significant facts.

Finally, although our templates appear to give us good coverage, they do not give universal coverage and there are certain question topics and question types outside their scope. They are not able to accommodate questions dealing with uncertainty or likelihood, and currently do not support analysis of the causal structure of events (to answer "Why...?" questions). In addition, questions requiring more sophisticated problem-solving strategies, e.g., diagnosis or abduction, and questions dealing with advanced concepts not already in the KB (e.g., complementarity of nucleotides), cannot be handled. Some examples include:

- I-7-3 "What are the building blocks of proteins?"
- I-6-48 "What structural property determines the polarity of a DNA strand?"

Finally, questions requiring specification of "impossible objects" could not be posed ("Is <object> possible?"), as the interface would not allow such objects to be entered in the first place (they would produce constraint violations at knowledge entry time). One could perhaps view detection of such constraint violations as a kind of answer in itself.

6. SUMMARY AND CONCLUSIONS

Our goals in this work are to enable users, not trained in AI, to communicate a wide variety of questions to a knowledge base, in a way allowing machine inference to then be applied to derive answers. We have described a novel extension of template-based question-answering in which the question scenario is entered graphically, and as a result a restricted set of domain-independent question types can be identified. Appendix A gives the catalog of question types we have developed, the result of extensive analysis of many text-book questions. This catalog itself provides a starting point for others with similar goals, and hence is also a significant contribution of this paper.

The evaluation of this work suggests that the basic approach is viable: domain experts were able to pose and receive acceptable, inferred answers to a broad range of questions, without having to encode those questions in logic. However, the evaluation also revealed several significant areas for future work, in particular in developing methods to help experts exploit the range of question templates available.

Acknowledgements

This material is based upon work supported by the Space and Naval Warfare Systems Center - San Diego (DARPA's Rapid Knowledge Formation project) under Contract No. N66001-00-C-8018. We are grateful to all the other members of our team who have also contributed to this work.

7. REFERENCES

[1] L. Acker and B. Porter. Extracting viewpoints from knowledge bases. In *AAAI'94*, pages 547–552. AAAI Press, 1994.

[2] L. Androutsopoulos, G. D. Ritchie, and P. Thanisch. Natural language interfaces to databases: An introduction. *Journal of Language Engineering*, 1(1):29–81, 1995.

[3] P. Clark, J. Thompson, K. Barker, B. Porter, V. Chaudhri, A. Rodriguez, J. Thomere, S. Mishra, Y. Gil, P. Hayes, and T. Reichherzer. Knowledge entry as the graphical assembly of components. In *Proc 1st Int Conf on Knowledge Capture (K-Cap'01)*, pages 22–29. ACM, 2001.

[4] P. Cohen, R. Schrag, E. Jones, A. Pease, A. Lin, B. Starr, D. Easter, D. Gunning, and M. Burke. The DARPA high performance knowledge bases project. *AI Magazine*, 19(4):25–49, 1998.

[5] D. Elworthy. Question answering using a large NLP system. In *Proc TREC-9 Conference*, pages 355–360. NIST, 2001.

[6] A. C. Graesser and S. P. Franklin. QUEST: A cognitive model of question answering. *Discourse Processes*, 13:279–303, 1990.

[7] E. Hovy, L. Gerber, U. Jermjakob, M. Junk, and C.-Y. Lin. Question answering in webclopedia. In *Proc TREC-9 Conference*, pages 655–664. NIST, 2001.

[8] IET Inc. The RKF test question suite. http://www.iet.com/Projects/RKF/, 2001.

[9] W. Lehnert. *The Process of Question Answering*. Erlbaum, NJ, 1978.

[10] K. R. McKeown. *Text Generation*. Cambridge Univ. Press, UK, 1985.

[11] Microsoft. *Microsoft SQL Server 7.0 Data Warehousing*. Microsoft Press, 1999. (Chapter 11).

[12] E. H. Shortliffe. *Computer-Based Medical Consultations: MYCIN*. American Elsevier, NY, 1976.

[13] H. Tennant, K. Ross, R. Saenz, C. Thompson, and J. Miller. Menu-based natural language understanding. In *Proc. SIGCHI Conference on Human Factors in Computing systems*, pages 154–160, 1983.

[14] J. Thomere, K. Barker, V. Chaudhri, P. Clark, M. Eriksen, S. Mishra, B. Porter, and A. Rodriguez. A web-based ontology browsing and editing system. In *IAAI'02*, pages 927–934, 2002.

Appendix A: The Catalog of Question Templates and Answer Procedures

ID	TEMPLATE	ANSWER PROCEDURE (outline)
Deductive Reasoning		
q1	What is <object>?	For events: present <object>'s class, participants (agent, object, ...), purpose, and subevents. For physical objects: present <object>'s class, parts, connections, location.
q2	What is/are the <relation> of <object>?	Query KB for objects in that relationship
q3	Whose <relation> is <object>?	Query KB for objects in that relationship
q4	Is <object> a <class>?	Query KB for class membership
q5	Describe <object> as a kind of <class>	Compute properties of <object> applicable to <class>
q6	How many <classes> are in the <relation> relationship to <object>?	Query and count objects in that relationship
q7	Does <relation> of <object> have <value>?	Query for specific object in that relationship
q8	Is <quantity1> <greater/less> than <quantity2>?	Query and compare
q9	What is the relation between <object1> and <object2>?	Search a set of possible relations for connection
q10	How many types of <class> are there?	Count direct subclasses of <class>
Discrete Event Simulation		
q11	What happens during <event>?	Present the subevents of event ("During <event>, First <subevent1>, then ...")
q12	What happens to <object> during <event>?	Run simulation to find <object>'s parts, connections, location at each step
q13	During <event>, what happens to <object> <before/after> <subevent>?	Same, except only report steps before/after named event
q14	How does the <relation> of <object> change during <event>?	Run simulation and present how <relation> of <object> changes
q15	During <event>, after <event>, what is the <relation> of <object>?	Run simulation to compute that relation after the stated event
q16	During <event>, does <object> <event-type> <object>?	Search subevents of <event> for a matching subevent.
q17	During <event>, when does <object> <event-type> <object>?	Find matching subevent, and present its previous, next, and parent ("before", "after", and "during") events.
q18	During <event>, how does <object> <event-type> <object>?	Find matching subevent, and present its own subevents ("First <subsubevent1>, then ...")
q19	What is the role of <object> in <event>?	Find all subevents where <object> is a participant
q20	What events are enabled by <object> during <event>?	Trace the dependency chain between effects of subevents involving <object>, and preconditions for subsequent events
q21	What kinds of final products result from <object> during <event>?	Trace dependency chain to find subsequent created objects
q22	Given <situation>, what might the consequences be?	Create situation, run simulation and report results
q23	In <situation>, if <event> occurred, what might the consequences be?	Create situation, execute event, run simulation and report result
q24	In <situation>, if <assertion> were <true/false>, what might the consequences be?	Create and modify situation, run simulation and report results
Qualitative Reasoning		
q25	In <object/event>, what factors affect <parameter>?	Search influence diagram for parameters upstream of <parameter>
q26	In <object/event>, what factors might cause <parameter> to <increase/decrease>?	Search for parameters upstream of <parameter> with desired directional influence
q27	In <object/event>, does <parameter> affect <parameter>?	Search for connection in influence diagram
q28	In <object/event>, what are the effects of <parameter> on <object>?	Search for connection from <parameter> to parameters of <object>
Analogical/Comparitive Reasoning		
q29	What is the <similarity/difference> between <object1> and <object2>?	Compute and compare fixed list of properties of objects