# Capturing and Answering Questions Posed to a Knowledge-Based System

**Peter Clark[1], Shaw-Yi Chaw[2], Ken Barker[2], Vinay Chaudhri[3], Phil Harrison[1], James Fan[2], Bonnie John[4], Bruce Porter[2], Aaron Spaulding[3], John Thompson[1], Peter Z. Yeh[2]**

[1]Boeing Phantom Works, Seattle, WA 98124
[2]Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712
[3]SRI International, 330 Ravenswood Ave, Menlo Park, CA 94025
[4]HCI Institute, Carnegie-Mellon University, Pittsburgh, PA 15213

## ABSTRACT

As part of an ongoing project, Project Halo, our goal is to build a system capable of answering questions posed by novice users to a formal knowledge base. In our current context, the knowledge base covers selected topics in physics, chemistry, and biology, with AP (advanced high-school) level examination questions. The task is challenging because the original questions are linguistically complex and are often incomplete (assume unstated knowledge), and because the users do not have prior knowledge of the system's ontology. Our solution involves two parts: a controlled language interface, in which users reformulate the original natural language questions in a simplified version of English, and a novel problem solver that can elaborate initially inadequate logical interpretations of a question by using relevant pieces of knowledge in the knowledge base. An extensive evaluation of the work in 2006 showed that this approach is feasible and that complex, multisentence questions can be posed and answered successfully, thus illustrating novel ways of dealing with the knowledge capture impedance between users and a formal knowledge base, while also revealing challenges that still remain.

## Categories and Subject Descriptors

I.2.7 Natural Language Processing; I.2.8 Problem Solving, Control Methods, and Search

**General Terms:** Algorithms, Human Factors

**Keywords:** Controlled language, problem solving, question answering, knowledge-based systems

## INTRODUCTION

A key problem in capturing knowledge from people, in the context of knowledge-based systems, is bridging the gap between human language and formal (inference-supporting) knowledge. Human language is complex, broad, ambiguous, and often leaves information unstated (assumed); a formal knowledge base, on the other hand, requires precise and complete input. These differences create a profound gap between humans and machines, posing a major challenge for knowledge capture, and given that it is generally difficult for users to directly author formal representations, other approaches are needed to allow users to extend and query formal knowledge bases.

We have been exploring this challenge in the context of posing complex questions to a formal knowledge base (KB), with AP (advanced high-school) level examination questions. The KB covers selected topics in physics, chemistry, and biology, and the users do not have prior knowledge of the KB's ontology. Our solution involves two parts: a controlled language (CL) interface, in which users reformulate the original natural language questions in a simplified version of English, and a novel problem solver that can elaborate initially inadequate logical interpretations of a question by using relevant pieces of knowledge in the knowledge base. Our goal with controlled language is to find a "sweet spot" of language restriction that is both usable by people and understandable by machines. Our goal with the problem solver is to systematically elaborate logical question interpretations that reflect the original question but are still inadequate for question answering because of missing information, before passing them on to a deductive engine to solve for an answer. In this paper, we present this approach and describe the results of an extensive evaluation performed in 2006. The evaluation illustrates that the approach is viable, and thus that we have made some significant inroads into addressing these challenges, while also revealing some major challenges that still remain.

## USING A CONTROLLED LANGUAGE FOR POSING QUESTIONS

Posing questions to a knowledge-based system remains a challenging task in artificial intelligence (AI), in particular if the system is intended to answer a wide variety of questions, and if the end users are not fluent in formal logic. Typically, the designer is caught between using "fill in the blank" question templates (e.g., Clark et al. 2003), which severely restricts the scope of questions that can be posed, or attempting full natural language processing on questions, which remains a challenging task. In our work, we have aimed for a sweet spot between these two extremes, using a controlled language (a simplified version of English) called CPL (Computer-Processable Language) for posing questions, and feedback mechanisms to allow end users to confirm or modify their queries.

While there has been a substantial amount of prior work on CL processing, the majority of it has been devoted to making text easier for people to understand (e.g., Hoard et al. 1992; Mitamura et al. 2003), rather than, as is our goal here, to make text easier for computers to understand. Despite this, several ongoing projects with computer-processable languages have demonstrated their utility (e.g., ACE (Fuchs et al. 1998), PENG (Schwitter & Tilbrook 2004), GINO (Bernstein & Kaufmann 2006)), and the commercial success of some CLs (e.g., AECMA simplified English (AER 2005)) suggests that people can indeed learn to work with restricted English.

Example 1 : Original Question:

*An object is thrown with a horizontal velocity of 20 m/s from a cliff that is 125 m above level ground. If air resistance is negligible, the time that it takes the object to fall to the ground from the cliff is most nearly (a) 3 s (b) 5 s (c) 6 s (d) 10 s (e) 15 s*

Reformulation in CPL:

*An object is thrown from the top of a cliff.*
*The initial horizontal velocity of the object is 20 m/s.*
*The initial vertical velocity of the object is 0 m/s.*
*The height of the cliff is 125 m.*
*The object falls from the top of the cliff to the ground.*
*What is the duration of the fall?*

Example 2 : Original Question:

*A cyclist must stop her bike in 10 m. She is traveling at a velocity of 17 m/s. The combined mass of the cyclist and bicycle is 80 kg. What is the force required to stop the bike in this distance?*

Reformulation in CPL:

*An object moves.*
*The mass of the object is 80 kg.*
*The initial velocity of the object is 17 m/s.*
*The final velocity of the object is 0 m/s.*
*The distance of the move is 10 m.*
*What is the force on the object?*

**Figure 1:** Two examples of questions reformulated in CPL.

## Design of the Controlled Language

Figure 1 shows two typical AP questions in the physics domain, and valid reformulations of them in our controlled language CPL. As can be seen in the second example, the onus is still on the user to spell out implicit commonsense knowledge (e.g., the person is riding the bicycle; if the person is traveling at 17 m/s then so is the bike; stopping the bike means stopping the bike+person "object") as well as simplifying the grammar, and even then the reformulation may be incomplete (in this case, the necessary knowledge to determine the force required to stop the bike+person is not specified); the subsequent section on the problem solver describes how such incompleteness is handled.

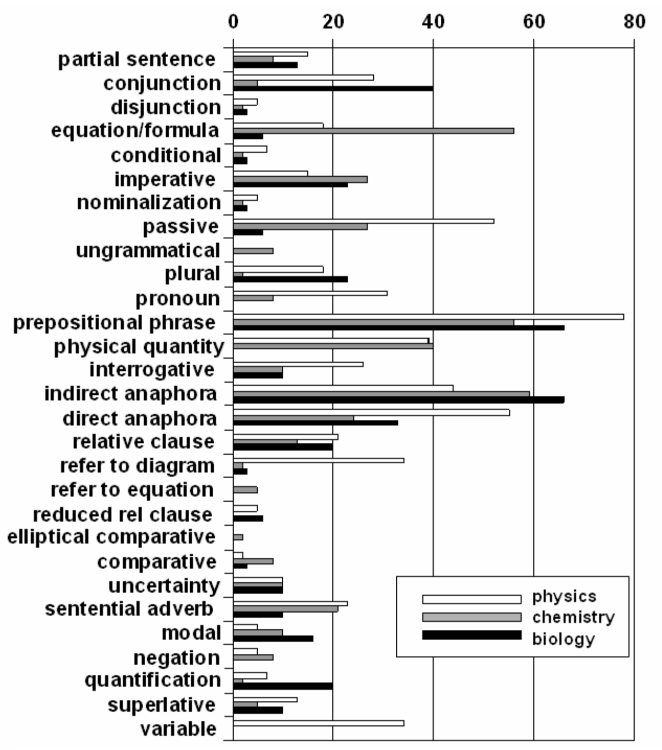

**Figure 2:** Occurrence (percent of AP question sentences) of different linguistic phenomena.

The language of science questions involves a variety of linguistic phenomena that need to be either handled automatically in CPL or easily reformulated by the user into CPL through training or advice. To identify these, we conducted an analysis of AP science questions (105 sentences from 36 questions), identifying 29 phenomena and their frequency of occurrence, shown in Figure 2. This provides a basis for some of the language features and reformulation advice that the system offers. The original version of CPL, prior to this project, already handled nominalizations, passives, plurals, prepositional phrases, relative clauses, direct anaphora, and a limited form of conjunction. It has been extended to also handle chemical equations (through a special chemical equation parser), interrogatives, indirect anaphora (described shortly), comparatives, variables, and physical quantities. For the remaining phenomena, if they are used, then CPL offers reformulation advice to the user on how to rephrase the sentence so it can be understood (described shortly).

## Computer-Processable Language (CPL)

A basic CPL sentence has the form

subject + verb + complements + adjuncts

where complements are obligatory elements required to complete the sentence, and adjuncts are optional modifiers. Users follow a set of guidelines while writing CPL. Some guidelines are stylistic recommendations to reduce ambiguity (and hence misinterpretation), while others are firm constraints on vocabulary and grammar. Some example guidelines are

- Keep sentences as short and simple as possible.
- Use just one clause per sentence.
- Assume the computer has no common sense. State the obvious in the question.
- Identify and describe the objects, events, and their properties involved in the question.
- Use "a" to introduce an item and "the" to refer back to it.
- Use "first" and "second" to distinguish two of the same kind of items.
- Use “There is a …” if needed to introduce an object.
- Do not mix groups and group members in a scenario.
- Avoid using pronouns, instead refer using a name (“the block”, “the table”).
- Begin a question with the words “What is”, “What are”, "How many", "How much", or “Is it true that”.
- Restate multiple-choice questions as simple questions.
- Ask for just one value in a single question.
- Set up a question by talking about one specific object.
- Use a question or statement in place of a command (imperative).
- Always include a unit of measure after a numerical value, or the word "units".

The full list of guidelines along with examples is given in the CPL user’s guide (Thompson and Clark 2006).

In addition, there are some grammatical and vocabulary restrictions, similarly designed to reduce ambiguity and guide the user away from difficult-to-interpret structures. For example, words of uncertainty (e.g., “probably”, “mostly”) are not allowed, not because they cannot be parsed but because their representation is outside the scope of the final logical language. In all these cases, when a violation occurs, CPL’s advice system (described shortly) reports the error and offers extensive rewriting suggestions with examples to help the user rephrase sentences.

For example, instead of writing

*Assume you are told that the best estimate of the mass of Planet X is 400 kg.*

a valid reformulation would be (short sentences, no imperatives)

*The mass of Planet X is 400 kg.*

Similarly, instead of writing

*A block starting from rest…*

a valid reformulation would be (identify the objects and their properties)

*The initial velocity of the block is 0 m/s.*

The choice of these restrictions is not based just on the difficulty of processing, but also on how easy it is for the user to work with them. For example, although many NLP systems perform pronoun resolution moderately well, it is easy for a user to simply refer by name and thus avoid possible resolution errors in the interpretation. The same is true for other phenomena, e.g., imperatives, disjunctions.

## CPL Interpreter

We now briefly describe the CPL interpreter itself; further details are provided in (Clark et al. 2005). Parsing is performed using SAPIR, a mature, bottom-up, broad coverage chart parser, also used commercially (Harrison & Maxwell 1986). During parsing, the system also generates a "logical form" (LF). The LF is a simplified and normalized tree structure with logic-type elements, generated by rules parallel to the grammar rules, that contains variables for noun phrases and additional expressions for other sentence constituents. Some disambiguation decisions are performed at this stage (e.g., structural, part of speech), while others are deferred (e.g., word senses, semantic roles), and there is no explicit quantifier scoping. An example LF is as follows, where DECL, S, VAR, and PP stand for Declarative, Sentence, Variable, and Prepositional Phrase:

```
;;; LF for "A ball falls to the ground."
(DECL ((VAR _X1 "a" "ball")
       (VAR _X2 "the" "ground"))
       (S (PRESENT) _X1 "fall" (PP "to" _X2)))
```

The LF is then used to generate ground logical assertions containing existentially quantified variables. First, a set of simple, syntactic rewrite rules is applied recursively to the LF to transform it into a set of ground, binary clauses of the form r(x,y), where each syntactic relation and preposition becomes a binary predicate:

```
syntactic-object(?f,?b)
“to”(?f,?g)
word-for(?b, [“ball”,noun])
word-for(?g, [“ground”,noun])
word-for(?f, [“fall”,verb])
```

Next, word sense disambiguation and semantic role assignment is performed. Because we are working in a restricted domain, word sense disambiguation is considerably easier than with a broad coverage application, as the choice of senses is constrained to the concepts in the KB. In addition, WordNet (Miller & Fellbaum 1998) is used to help expand the vocabulary of words recognized by CPL. To disambiguate a word, first its WordNet senses are found, and then CPL looks to see if any are mapped directly to a KB concept (each concept in the KB has a list of associated WordNet synsets). If one or more are found, the most likely sense is selected based on corpus frequency statistics. If not, CPL searches up WordNet’s hypernym (generalization) tree until a synset that is mapped to a KB concept is found. In this way, specific words like “bicycle” or “cliff” can be used by the user, although they are not explicitly in the KB, as they are mapped through this process to more abstract KB concepts (e.g., VEHICLE and PHYSICAL-OBJECT, respectively). For semantic role assignment, semantic roles are identified using a small rule base, e.g., for "of"(x,y), if x is an OBJECT and y is a MATERIAL, then "of"(x,y) maps to made-of(x,y). Here, the result is

```
;;; Logic for “A ball falls to the ground”
isa(?b,HOLLOW-BALL)
isa(?g,PHYSICAL-OBJECT)
isa(?f,FALL)
object(?f,?b)
destination(?f,?g)
```

Finally, other structural reorganizations are performed, including coreference resolution and handling coordination.

In addition, two modules for recognizing and correcting some semantic errors in the interpretation are applied, essentially correcting some types of “loose speak” (e.g., metonymy) by the user. These modules essentially look for constraint violations in the interpretation, e.g., on type constraints on a predicate’s arguments, and if found search for a minimal edit that will remove the violation using the KB for guidance (Yeh et al. 2003; Fan and Porter 2004). This helped reduce, but not eliminate, the impedance between the question-asking module’s output and the problem solver. The query itself is represented as variables to find values for, or a special directive to the problem solver indicating specific question types, e.g., determining the similarities or differences between entities.

## Interactive Question-Asking Process

While having a controlled language makes interpretation more reliable, it also introduces the challenge of having the users learn to use it. To facilitate this, the CPL interpreter is integrated with two other components, namely, an advice system that detects CPL errors and provides reformulation advice, and an interpretation display system that presents the system’s understanding of the question (both as English paraphrases and graphically) so the user can check that the system understood correctly. Figure 3 illustrates the overall process of using CPL in this environment. First, the user reformulates the original question in CPL. If there are CPL errors, the system pinpoints them and offers rewriting advice, and the user fixes the mistakes. If the user’s question is valid CPL, the system creates a logical interpretation, and presents that back to the user both in English paraphrases and as a graph. The user can then inspect the system’s understanding to make sure that it is correct, and if so, invokes the problem solver to attempt answering it.

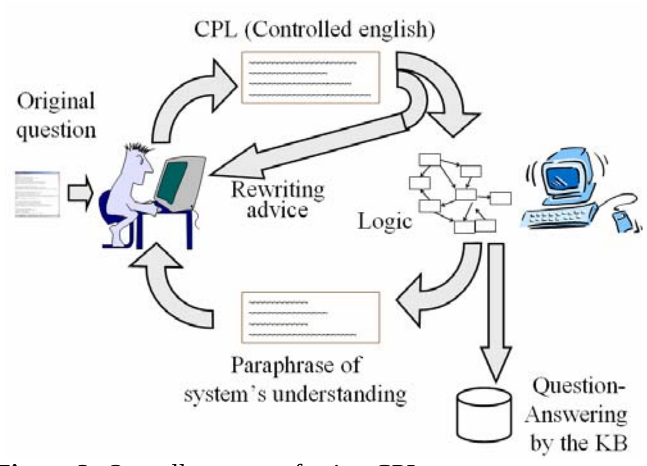
CPL (Controlled english)
Original question
Rewriting advice
Logic
Paraphrase of system’s understanding
Question-Answering by the KB

**Figure 3:** Overall process of using CPL.

### *Advice System*

The advice system currently contains 106 advice messages, triggered when the user violates a CPL guideline. Grammatical violations are detected when grammar rules outside the scope of CPL, but within the scope of full English, are used in the parse, and thus targeted rewriting advice can be given. In addition, vocabulary can indicate errors, e.g., using a banned word. For example, if the user forgets a unit of measure (e.g., the user enters “The initial velocity of the object is 17”) the system responds

> Always specify a unit for numbers (e.g., “10 m”, not “10”). Use the word “units” for dimensionless units.
> e.g., Instead of writing: *"The velocity is 0."*
> write *"The velocity is 0 m/s.."*

Note that the examples in the advice messages are canned text, not an automatic rewrite of the user's actual input (otherwise CPL could simply do the rewrite itself). In general, automatic rewording would be very challenging, especially with longer, more complex sentences.

### *Interpretation Display System*

When the system generates a logical interpretation of the user’s question, it shows its understanding back to the user in two ways:

1. As a set of **English paraphrases** of the logic, i.e., the question is “round-tripped” from the user’s CPL, to logic, to a machine-generated English paraphrase.
2. As a **graph**, where nodes represent objects or events in the question scenario (individuals), and arcs represent relationships (binary predicates). If errors are apparent here, the user has the option of directly editing the graph (limited to renaming nodes or arcs), or reformulating the original sentences. The left side of Figure 4 shows the graph produced for the second example in Figure 1 (stopping a moving object).

The goal is that the user can validate whether the system has understood the question correctly from this, and that both the graph and paraphrase can make errors obvious without requiring the user to become familiar with the internal representation. The graph is an interesting partial solution to this, as it reveals some of the underlying knowledge (predicates) while still being accessible to users. In debriefs after the evaluation, users reported that the graphical presentation was the most useful.

## PROBLEM SOLVER

While the CPL interface helps produce a valid, logical interpretation of the user's question, this is often not enough to produce an answer. Fundamentally, answering science questions requires not just "doing computation", but working out how facts in a question can be mapped onto an abstract model capable of deriving an answer. It is this ability, more than any other, that characterizes a good scientist. For example, in physics one may have learned models (a system of objects, parameters, relationships, and equations) about falling from rest, or two-object collision, or movement with friction. Then, given a new problem about a specific object moving in a specific way, the challenge is to map this problem onto the appropriate model such that an answer can be derived. If we are lucky, the question will provide exactly the right information so that the appropriate model can be applied through standard deductive reasoning (e.g., inheritance). However, this is rarely the case, and as a result a search is needed to find potentially applicable models, some making assumptions unstated in the question, to see if they can be used to answer the question. The prob-

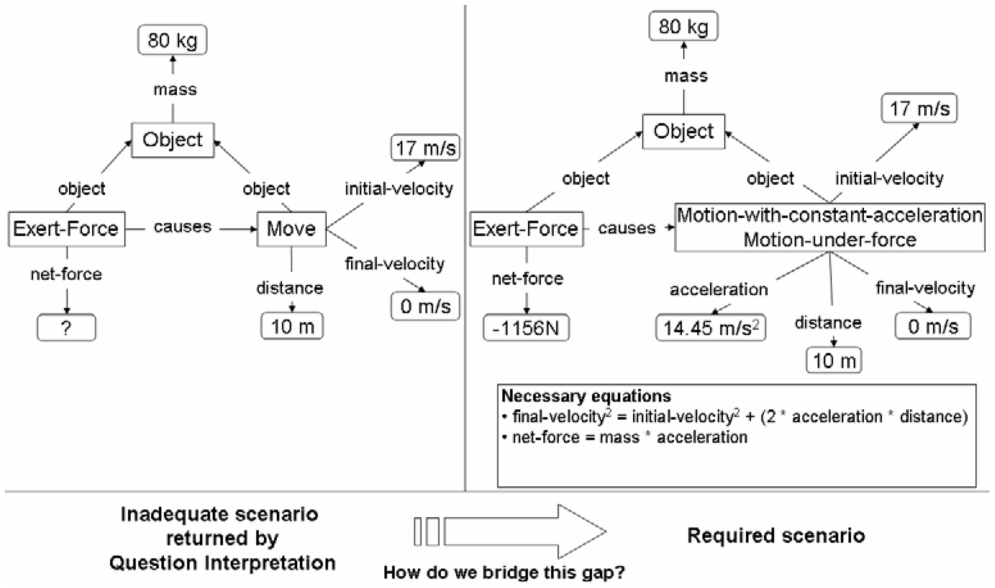

**Figure 4:** The initial question interpretation (left side) does not contain enough information to solve the problem (missing information about how the object accelerates). To address this, the problem solver searches for models that can answer the question, including by making assumptions, and elaborates the question scenario so those models can be applied. In this case a model that assumes constant acceleration is found to be able to answer the question, and so is used.

lem solver aims to do exactly this, matching the formalized question against models in the KB (i.e., axiomatized concepts denoting minitheories, such as (in physics) FALL-FROM-REST with equations relating the time, distance, and acceleration of the falling object), and thus enable questions that otherwise could not be solved to be answered.

## Example

To illustrate how this works, consider the second question shown earlier in Figure 1, whose logical interpretation is sketched in the left side of Figure 4, asking how much force is required to stop a moving object. The logical interpretation, the input to the problem solver, consists of two parts. First, the *scenario* contains the setup of the question, represented with instances of EXERT-FORCE, PHYSICAL-OBJECT, and MOVE. Second, the *query* identifies the variable of interest – in this case, the net-force of the EXERT-FORCE event – shown as the node with a question mark.

Solving this question requires applying relevant equations to determine the acceleration of the MOVE and the net-force exerted to cause it. Axioms about the general concepts (e.g., MOVE) in the scenario on the left in Figure 4 do not contain the necessary information (e.g., equations) for calculating the net-force required to move the object. As a result, the reasoner is unable to compute an answer to the query. In fact, these equations are found in the axioms about two other concepts (models) in the knowledge base not mentioned in the scenario, namely, MOTION-WITH-CONSTANT-ACCELERATION and MOTION-UNDER-FORCE. Figure 4 shows the differences between the inadequate scenario returned by question interpretation and the scenario necessary for answering the question. We thus would like the problem solver to search for such models in the KB to elaborate the inadequate scenario so that an answer can be deduced.

### State-Space Search

The problem solver systematically explores the search space (building a graph of the space) until it finds an elaborated scenario that can be used by the reasoner to return a solution for the question. Figure 5 shows the search graph created in solving the example question.

### States

Each state in the state space graph represents a scenario. The initial state represents the original scenario returned by question interpretation. All other scenarios in other states are elaborations of the original scenario, i.e., the scenario with additional axioms applied to it. In Figure 5, State 1, which is the initial state, contains the original interpretation of the example question and all other states in the graph are elaborations of it.

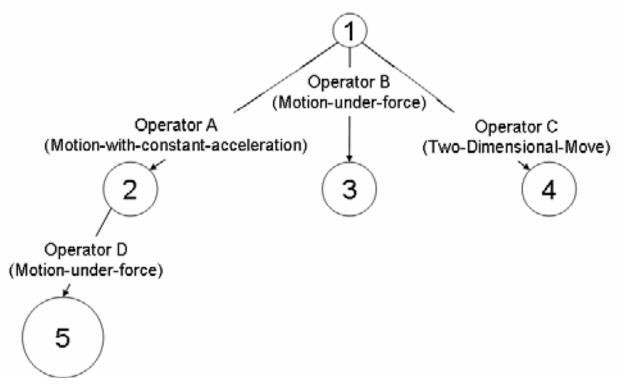

**Figure 5:** Problem solving search graph example

### Operators

Operators elaborate one scenario (a state) to produce another one. Every scenario can be elaborated by potentially many concepts in the knowledge base. Operators are created from the set of concepts that can be applied to elaborate the scenario. The set of applicable concepts is selected from the knowledge base using a flexible semantic matcher (Yeh et al. 2003). The matcher works by comparing the facts in the scenario with the axioms about each concept to find the "biggest overlap"; formally, both the scenario facts and concept axioms are internally represented as graphs (similar to conceptual graphs). The matcher uses knowl-

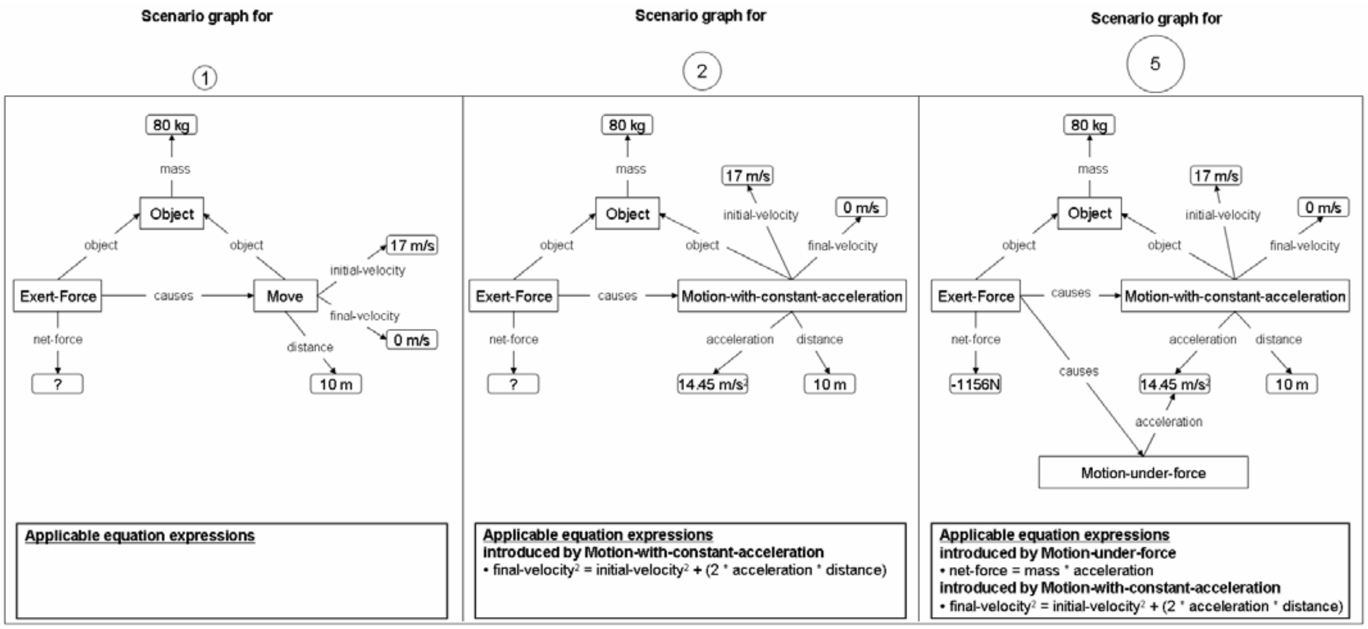
Scenario graph for
1
Scenario graph for
2
Scenario graph for
5
80 kg
mass
Object
object
object
Exert-Force
causes
Move
initial-velocity
17 m/s
final-velocity
0 m/s
net-force
distance
?
10 m
Motion-with-constant-acceleration
acceleration
14.45 m/s2
-1156N
Motion-under-force
Applicable equation expressions
Introduced by Motion-with-constant-acceleration
• final-velocity2 = initial-velocity2 + (2 * acceleration * distance)
Introduced by Motion-under-force
• net-force = mass * acceleration

**Figure 6:** Scenario graph examples showing states 1, 2, and 5 from the search graph of Figure 5. State 1 is the initial state containing the original scenario graph for the question. State 2 is the scenario graph that results from elaborating the scenario graph in State 1 using the MOTION-WITH-CONSTANT-ACCELERATION concept in the knowledge base being queried. This elaboration introduced an equation that calculates the acceleration of the MOVE event to be 14.45 m/s$^2$.

edge (about how their concepts and relations subsume each other) to find the largest connected subgraph in one representation that is isomorphic to a subgraph in the other. This matcher then uses a library of about 200 generic transformation rules to shift the representations to improve the match. This improvement might enable other subgraphs to match isomorphically, which in turn might enable more transformation rules to apply, and so on until the match improves no further.

A successor state containing an elaborated scenario is created when an operator is applied to a state. This elaborated scenario includes new knowledge introduced by axioms about the concept from which the operator was created. In Figure 6, the scenario graph for state 2 in Figure 5 is created when Operator A is applied to State 1.

### *Control Strategy*

The search graph is expanded in a breadth-first manner. The application of operators is based on the degree of match between the scenario in the state and the selected concept. In Figure 5, Operator A has a higher priority than Operator B because the concept MOTION-WITH-CONSTANT-ACCELERATION has a higher degree of match to the scenario in State 1 than the concept MOTION-UNDER-FORCE.

### *Goal Test and the Goal State*

The goal test determines if the scenario of a state is capable of returning a solution. In Figure 6, State 2 fails the goal test because it does not answer the question – its scenario does not return a value for the net-force of the EXERT-FORCE event. The scenario in State 5 satisfies the goal test as its scenario returns a numeric value, i.e., -1156 Newtons, as the solution.

### *Result*

In this way, the problem solver helps bridge the gap between the initial formalization of the user's question, and the internal science models (axioms) encoded in the KB, arguably the biggest challenge to overcome for the whole question capture and answering process. In the next section we evaluate the degree to which the full question asking and answering pipeline has been successful.

## EVALUATION

### Methodology

The question-answering system, along with a separate knowledge acquisition system used to build KBs (not described here, see Chaudhri et al. 2007), was extensively evaluated during May and June 2006. Experts first built KBs about physics, chemistry, and biology (outside the scope of this paper). Then, of interest here, a different set of users attempted to pose AP-level questions using CPL to the KBs and received answers. For this task, there were two users in each domain (physics, chemistry, biology), four being undergraduates and two being graduate students. They each received 6 hours of training in using CPL, followed by a week (each) using the question-asking system.

### Overall Accuracy

Table 1 shows the accuracy measures obtained for the overall system's question-answering performance. Note that a correct answer requires not only a good interpretation of the user's question and good reasoning, but also that the user had given enough information to solve the problem in the first place (e.g., making commonsense facts explicit), and that the KB itself had the knowledge to answer the question. Thus these figures only represent a lower bound on the performance of the QA system.

The most important conclusion from this result is that this approach to question asking is viable; in particular, many of the questions (especially in physics) were multiple sentences, involving a scenario description and a query posed to that scenario. The fact that a controlled language approach was able to interpret at least some questions of this complexity, and the problem solver able to map them onto appropriate science models, is a significant achievement. However, there were also many cases where questions were not answered, as we now discuss.

| Domain | Number of questions | Percentage correct | | |
|---|---|---|---|---|
| | | User1 | User2 | **Average** |
| Biology | 146 | 52% | 24% | **38%** |
| Chemistry | 86 | 42% | 33% | **37.5%** |
| Physics | 131 | 16% | 22% | **19%** |

**Table 1:**Overall correctness scores for question answering. Six users (two in each science) each posed questions to six KBs (two in each science, each built independently).

## Credit and Blame Analysis

To examine which failures were due to the question-answering system itself, and which were due to other causes, we performed a blame assignment study by examining the CPL, system's reasoning, and KBs in detail for a random sample of cases where the system produced an incorrect (or no) answer. We found that

- In about one-third of the cases, the knowledge required to answer the question was simply missing in the KB, i.e., the question was inherently unanswerable.
- In about one-third of the cases, while the CPL question had been interpreted correctly, the user had omitted some essential information (e.g., not made a common-sense fact explicit) in the original formulation, so that again the question was inherently unanswerable given that the KB did not contain that knowledge.
- In about one-third of the cases, the CPL interpreter had misinterpreted the CPL English (and the misinterpretation not been recognized and corrected by the user) or the problem solver had failed to generate a solution.

Thus, at least from this sample, about one-third of the failures in the overall scores were due to failure of the question-answering machinery itself. The full analysis is provided in (Chaudhri et al. 2006). This shows that, while successful, there is still considerable room for improvement. In particular, it shows the need for further improving reformulation advice, and further improving the interpretation display so that it is more obvious to the user when the system has made a mistake.

## Reformulation Actions by the User

Although users were able to create a correct CPL formulation of the questions in many cases, they rarely "got it right" the first time. A metric we have found useful is "mean tries to completion" (MTC), namely, the number of attempts a user had at formulating a question before either getting a satisfactory answer from the computer (during the subsequent question-answering step) or giving up. Table 2 shows this statistic computed from a random sample of questions in each science. This table shows that typically users took about six attempts to get an answer (or give up) in physics and chemistry. In biology, the MTC was a lot less, primarily because questions were typically about definitions and properties (e.g., "What is an X?"), in sharp contrast to the often complex "story" questions posed in physics, and the algebraic questions posed in chemistry.

| | **MTC (mean # of tries/qn)** |
|---|---|
| Physics | **6.3 tries/qn** |
| Chemistry | **6.6 tries/qn** |
| Biology | **1.5 tries/qn** |

**Table 2:** Users typically made two to six attempts to reformulate a question before success or giving up.

Again care needs to be taken to interpret this data, as some of the reformulation attempts were not due to failure of the CPL interpreter itself but due to failure of the subsequent question-answering process (e.g., missing knowledge). To explore this further, we identified 13 types of reformulation actions that users performed when re-attempting a question. Figure 7 summarizes the frequency of these actions, as seen on a random sample of questions. From this analysis, and from the traces, the users appeared to have little trouble working within CPL's grammar restrictions (only a few actions were in response to a CPL violation). Rather, the major challenge the users had was finding the right wording that enabled the system to answer the question. In general, there are only certain wordings that translate to logical forms that trigger the right problem-solving process. In many cases the users appeared to be performing trial-and-error guessing until they hit a wording that worked, or they

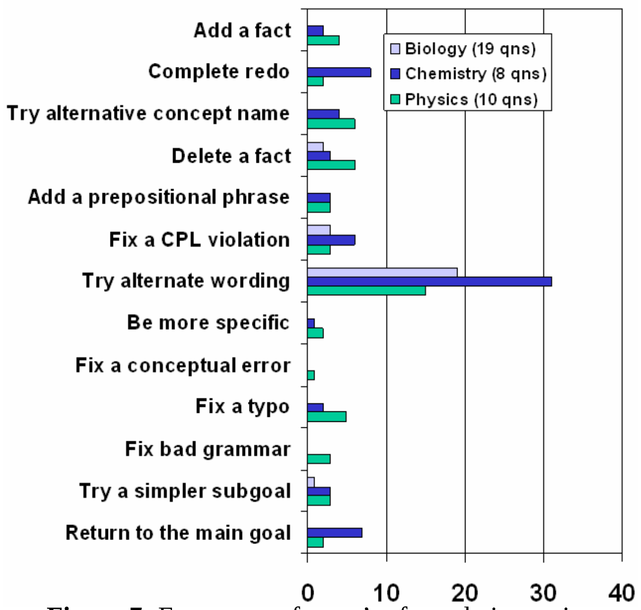

**Figure 7:** Frequency of users' reformulation actions.

gave up. For example, chemistry question E6 asks whether a compound is insoluble. The users tried multiple ways of talking about solubility ("soluble", "dissolve", "solution", "insoluble") until finally hitting on "solubility" as a property for which the system was able to give a value. While these earlier attempts all translate into valid logical expressions, typically the KB was not able to answer either because of missing knowledge (e.g., the relation between a dissolve action and solubility) or limitations of the problem solver.

## The Problem Solver

The goal of the problem solver is to match the interpreted question with a solution method (model) capable of answering it, including cases where there may be unstated assumptions in the question, thus performing elaboration (but not correction) of the interpreted question. To evaluate whether the problem solver is helping, we performed a separate study on the two physics KBs (KB #1 and KB #2) created for the intermediate evaluation, plus a third physics KB created by a trained knowledge engineer (KE). To test them, we used the question formulations from the earlier evaluation (combining the two users' sets), using those formulations that produced a correct answer on at least one of these three KBs (67 formulations). We investigated whether switching off the elaboration mechanism in the problem solver reduced the correctness score on this set. The results are shown in Table 3, and they demonstrate the problem solver's usefulness in broadening the range of questions that can be answered by the system. (Chaw et al 2007) provides a detailed description of the problem solver.

| | KB #1 | KB #2 | KE |
|---|---|---|---|
| Without elaboration | 9.0% | 17.9% | 0% |
| With elaboration | 61.2% | 56.7% | 61.2% |

**Table 3:** Percentage accuracy without and with the heuristic elaborations performed by the problem solver, on questions that at least one of the three KBs can answer.

## CONCLUSIONS

Capturing and answering questions posed to a knowledge base, a special case of formal knowledge capture from people, is challenging because it requires bridging the chasm between human language and formal knowledge. In this work we have developed a solution using a controlled language embedded in an interactive interface, coupled with a problem solver capable of aligning formalized questions with solution methods, even when the formalized question is itself incomplete (e.g., missing assumptions) for producing an answer directly. In our experiments, users were able to successfully pose and get correct answers to a large number of questions, including complex, multisentence questions beyond the scope of previous approaches. This result is significant as it demonstrates that our approach is viable, and thus may have applications beyond question answering to other knowledge capture environments (e.g., the Semantic web). However, our evaluation also shows several challenges. In particular, it still requires several attempts and some cognitive effort on the user's part to pose questions successfully. In cases where the system is unable to answer, the user is often left guessing about how to reformulate the question for another attempt. Despite this, our evaluation reveals that the approach is workable, and we are thus optimistic about this approach and believe that a combination of controlled language and flexible reasoning has a large role to play for human-machine interfaces of the future.

## Acknowledgments

We are grateful to Vulcan Inc, which funded this work as part of Project Halo.

## REFERENCES

Aerospace and Defence Industries Association of Europe (2005). *ASD Simplified Technical English.* Specification ASD-STE100.

Bernstein, A., Kaufmann, E. (2006). GINO – A Guided Input Natural Language Ontology Editor. In *ISWC'06*.

Chaudhri, V. et al. (2006). *AURA: Automated User-Centered Reasoning and Acquisition System.* Technical Report, SRI International.

Chaudhri, V. et al. (2007). Acquiring Knowledge from Science Text Books (submitted).

Chaw, S. Y., Porter, P., Barker, K., Yeh, P. (2007). *Towards an Ontology-Independent Problem-Solver.* Technical Report, AI-Lab, University of Texas at Austin.

Clark, P., Chaudhri, V. et al. (2003). Enabling Domain Experts to Convey Questions to a Machine: A Modified, Template-Based Approach. In *KCap'03,* 13-19, ACM Press.

Clark, P., Harrison, P., Jenkins, T., Thompson, T., Wojcik, R. (2005). Acquiring and Using World Knowledge Using a Restricted Subset of English. In *Proc FLAIRS'05*.

Fan, J., Porter, B. (2004) Interpreting Loosely Encoded Questions. In *AAAI'04*.

Fuchs, N. E., Schwertel, U., Schwitter, R. (1998). Attempto Controlled English. In *Proc. LOPSTR'98.*

Harrison, P., Maxwell, M. (1986). A New Implementation of GPSG. In *Proc. 6th Canadian Conf on AI,* 78-83.

Hoard, J., Wojcik, R., Holzhauser, K. (1992). An Automated Grammar and Style Checker for Writers of Simplified English. In *Computers and Writing,* 278-296.

Miller, G., Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database.* The MIT Press.

Mitamura, T., Baker, K., Nyberg, E., Svoboda, D. (2003). Diagnostics for Interactive Controlled Language Checking. In *Proc. CLAW'03.*

Schwitter, R., Tilbrook, M. (2004). *PENG: Processable ENGlish.* Technical report, Macquarie University, Australia.

Thompson, J., Clark, P. (2006). *Guide for CPL Users, Version 7.5.* Technical Report, The Boeing Company.

Yeh, P., Porter B., Barker K. (2003). Using Transformations to Improve Semantic Matching. In *Proc. KCap'03*