Enabling Experts to Build Knowledge Bases from Science Textbooks

Vinay K. Chaudhri[1], Bonnie E. John[2], Sunil Mishra[1], John Pacheco[1], Bruce Porter[3], Aaron Spaulding[1]

[1] SRI International, Menlo Park, CA, USA., [2] Carnegie Mellon University, Pittsburgh, PA, USA,
[3] University of Texas at Austin, Austin, TX, USA

ABSTRACT

The long-term goal of Project Halo is to build an application called Digital Aristotle that can answer questions on a wide variety of science topics and provide user- and domain-appropriate explanations. As a near-term goal, we are focusing on enabling subject matter experts (SMEs) to construct declarative knowledge bases (KBs) from 50 pages of a science textbook in the domains of Physics, Chemistry, and Biology in a way that the system can answer questions similar to those in an Advanced Placement (AP) exam in the respective discipline. The textbook knowledge is a mixture of textual information, mathematical equations, tables, diagrams, and domain-specific representations such as chemical reactions. In this paper, we explore the following question: Can we build a knowledge capture system to enable SMEs to construct KBs from the knowledge found in science textbooks and use the resulting KB for deductive question answering? We answer this question in the context of a system called AURA that supports knowledge capture from science textbooks.

Categories and Subject Descriptors

I.2.1 Applications and Expert Systems

General Terms

Algorithms, Performance, Human Factors, Languages

Keywords

Text Book Knowledge, Concept Maps, Equations, Tables

INTRODUCTION

Scientific information is growing faster than our ability to manage it. Furthermore, scientific results have cross-domain implications due to differing terminology and the volume of the knowledge. Traditional information retrieval systems are limited in their ability to support knowledge management because they cannot reason with the subject matter. Motivated by this, we have been developing a system, called AURA, with the goal of acquiring knowledge in a way that it could be used for answering questions by declarative inference. As an initial goal, we are focusing on enabling subject matter experts (SMEs) to author knowledge from approximately 50 pages each from college-level textbooks in Physics [1], Chemistry [2], and Biology [3]. There are two AP exams for Physics, and we have been focusing on Physics B.

We begin by giving an overview of our design approach and technical challenges we faced in developing AURA, and then consider in detail how we designed and implemented the knowledge capture interfaces. We present the result of an evaluation of those interfaces, and conclude with a summary and directions for future work.

K-CAP'07, October 28–31, 2007, Whistler, British Columbia, Canada.

AURA SYSTEM DESIGN AND OVERVIEW

We considered two classes of requirements in the design of AURA. The first set of requirements, identified using a *domain analysis,* was based on the content of the knowledge found in the three science textbooks. The second set of requirements, identified using a *SME need analysis,* was based on the fact that the users of the system are SMEs and not well versed in knowledge representation and reasoning (KR&R).

Domain Analysis

We started the domain analysis with a survey of the textbook knowledge in the three domains. The goals of the analysis were (a) to identify KR&R capabilities necessary to answer AP questions, (b) to assess the feasibility of implementing those capabilities, and (c) to estimate the coverage provided by the implemented capabilities.

We drew a sample of full-length tests from a recent offering of the AP exams in Physics, Chemistry, and Biology. We considered nine full-length tests in total, with three tests drawn from each domain. From the three tests we considered for each domain, for one of them, we did an in-depth analysis to identify the specific forms of KR&R necessary to answer questions. From the remaining two exams, we drew representative questions that had no similar question in the first exam that we had analyzed in depth. In total, we considered about 20 different kinds of KR&R types, but four of them contributed to answering

over 50% of the AP questions. We consider the four predominant KR&R types in detail.

Conceptual knowledge was the most frequent KR&R type across Chemistry and Biology. Examples of fragments of conceptual information include

In a Combination Reaction, two or more substances react to form exactly one product. [2]

A Cell is bounded by a membrane, called a plasma membrane. Within the membrane is a semi-fluid substance, cytosol, in which organelles are found. All cells contain chromosomes, carrying genes in the form of DNA. [3]

Representing conceptual information involves representing classes, subclasses, slots, slot constraints, and rules that assert general properties of all the instances of a class.

The second most frequent KR&R type involved equations. Almost every question on the AP Physics B exam involves using one or more mathematical equations. Many of the stoichiometry problems in Chemistry involve using mathematical equations. Chemistry also includes chemical reactions that are a domain-specific form of an equation. As an example equation, Newton's second law of motion can be succinctly captured using the equation *F= m * a*, that is, net force on an object (*F)* is equal to the object's mass (*m),* multiplied by its acceleration (*a).*

The third most frequent KR&R type was diagrams. In Physics, many problems are stated using diagrams. While in most of the cases, the same information can be stated without the diagrams, for a small percentage of cases, diagrams are essential. In Biology, there is an extensive use of diagrams to explain the information that is also introduced nontextually. In Chemistry, there is an extensive use of diagrams to visualize molecular geometry.

The fourth most frequent KR&R type was information that appears in tables. Like diagrams, in most cases, the information stated in tables is repeated in text, but in some cases, the use of tables is unavoidable. For example, the table of Figure 1 shows the qualitative relationship between the strength of several acids and bases, but this information is not repeated anywhere in the text.

After identifying the four most frequent KR&R types, we assessed the feasibility of building a knowledge capture interface for each type. Even though we had considered three full-length AP exams in our analysis, the assessment process reported here was further biased by the fact that we had chosen only 50 pages from each of the three textbooks as the scope of the current phase of the project.

In a prior system called SHAKEN [4], we had done substantial work on capturing conceptual knowledge, and we were confident that we could build on that work to meet the first, and most frequent, KR&R type. There is limited prior work in integrating acquisition of mathematical equations in the context of a reasoning system [5], but we had prior experience in Physics problem solving [6] that gave us reasonable confidence that we could meet this requirement. The domain of graphical and spatial knowledge is difficult and challenging. For most examples we considered in the analysis, the knowledge captured in the diagrams could be stated using text. Based on these considerations, we decided not to support knowledge capture using diagrams in the current phase of the project. Finally, for answering questions, the information in tables that was not stated using text as in Figure 1 was very important, and did not appear too hard to implement.

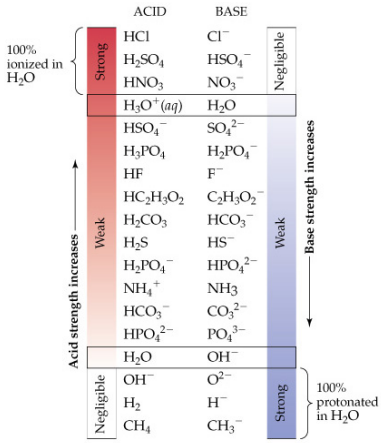

Figure 1. Relative strengths of some common conjugate acid-base pairs, listed opposite one another.

Based on this analysis, our initial design of AURA supports knowledge capture for conceptual knowledge, equations, and tables. We implemented a domain-specific interface for chemical reactions because reactions are fundamental for Chemistry. The data collected during the domain analysis supports the conjecture that these capabilities will allow us to answer approximately 50% of the AP questions in the three domains.

SME Need Analysis

Science SMEs are not trained to do KR&R. Our prior experience in building knowledge capture systems [7, 8] showed that (1) when SMEs are faced with the task of formulating knowledge, it is a challenge for them to determine where to begin (the *blank slate* problem), how much knowledge to encode, and how deep to go in their encoding; (2) knowledge formulation has a life cycle that includes initial formalization, testing and validation, revision, and further testing and question answering; and (3) since the SMEs are not well versed in KR&R we must place a high value on usability and the system should require minimal training. We therefore adopted a user-centered design approach to the development of AURA.

AURA System Architecture

The AURA system has three broad classes of functionality: knowledge formulation (KF), Question Formulation (QF), and Deductive Question Answering (QA). We show the system architecture in Figure 2.

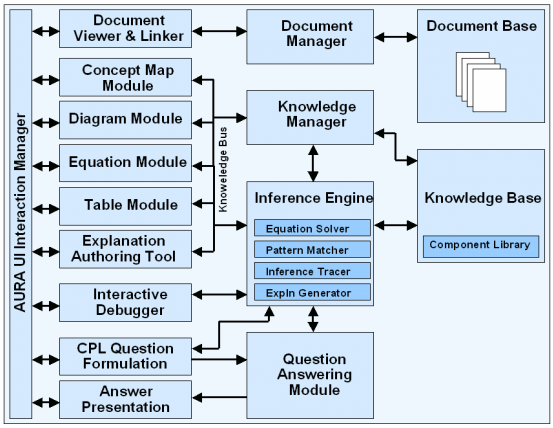

Figure 2. AURA system architecture.

To partially address the blank state problem, the KF subsystem of AURA supports a document-rooted interface. We load the electronic versions of each of three books, and they provide the scope of the knowledge to be encoded. We also provide tools that help a user get started by cross-referencing the document. These tools and facilities are represented using the *Document Viewer & Linker, Document Manager,* and *Document Base* modules in Figure 2. The *Concept Map* module in the KF subsystem deals with conceptual knowledge, and provides modules tailored for equations, diagrams, and tables—the KR&R types identified by the domain analysis. We have not yet implemented the interface for diagrams in the current system. The *Explanation Authoring Tool* enables a user to author explanation knowledge that is used in explaining the answer. To support the full knowledge entry life cycle, AURA provides an interactive debugger that can help detect errors in the KB. The capabilities such as knowledge revision and testing are embedded in the *Knowledge Manager.* In the current paper, we describe our approach for conceptual knowledge, equations, and diagrams. Acquisition of explanation authoring knowledge, and the knowledge debugger, is the subject of our current work, and will be described in future papers.

The QF subsystem uses a simplified form of English to allow a user to formulate questions. Once a question has been entered, the user can edit the system's interpretation of the question. The QA subsystem of AURA uses an object-oriented KR&R system called the Knowledge Machine—in short KM [9]. The inference engine in KM is enhanced using specialized reasoning modules such as for solving systems of equations found in Physics and Chemistry, and a pattern matcher that can relate the interpreted questions to the KB. A description of the QF and the QA capabilities is not included in this paper and is available elsewhere [10].

Interaction Design Approach

To address the SME need for usability, the development team adopted a user-centered design approach that included many techniques throughout the development process. We began with a walkthrough of the system architecture, examining it for its ability to support general usability concerns [11] so that development on the back end could proceed while the user interface (UI) was being investigated, designed, and tested. We then conducted Contextual Inquiries (CIs) [12] with science graduate students to better understand the training, knowledge, preferences, and work practices of our intended SMEs. Iterative UI design then proceeded, using competitive analysis, ideation, paper prototypes, interactive prototypes built in Flash, training materials development, and user testing at every stage. As we discuss various knowledge capture interfaces in greater detail, we will illustrate how the user-centered design approach influenced our solutions.

Knowledge Capture Interfaces

In describing the knowledge capture interfaces for AURA, we begin with the document-rooted interface, and then describe the interfaces for the three KR&R types: conceptual knowledge, equations, and tables.

Document-Rooted Interface

We designed the document-rooted interface to partially address the *blank slate* problem, and the problem of defining the context and the scope of the knowledge to be authored. With the document-rooted interface, we situate the knowledge formulation process in the context of a source document—here electronic versions of three science textbooks [1-3]. As the user selects sections from the book to encode, and underlines the words of interest, in the same way as if reading a book, AURA semantically maps the selected words in the source text into concepts that already exist in the KB, and presents them to the user on a palette. The search is semantic in the sense that AURA maps English words from the text to the concepts in the KB via WordNet synsets [13].

Another important aspect of addressing the blank slate problem is that we draw the existing concepts in the KB from a domain-independent KB called the Component Library [14]. The Component Library is built by knowledge engineers (KEs) and contains domain-independent classes such as *Attach, Penetrate, Physical Object*; predefined sets of relations such as *agent, object, location*; and property values to help represent units and scales such as *size, color*. These classes and relations provide a starting point to the SMEs in the knowledge formulation. In Figure 3, we illustrate the use of the document-rooted interface. The palette, shown in the bottom half of Figure 3, provides the user with a starting point for KF, and the SME does not have to start from scratch.

As the user formulates knowledge, AURA associates the formulated knowledge with the sections in the source textbook that were used as the starting point. Such cross-referencing provides a way to navigate the KB using the table of contents of the electronic book. To identify the

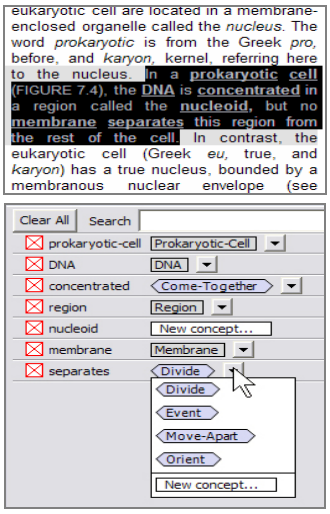

Figure 3. Top: a portion from the biology text, with selected words that the SME wants to encode. Bottom: the palette with the selected words and corresponding concepts from AURA's KB. *Nucleoid* did not match to any concepts, so AURA suggests that the user create a new concept. Some words match multiple concepts. Here we see the drop-down showing all the concepts *separates* matches.

knowledge formalized with a section of the book, a SME can use the table of contents of the book to retrieve a particular section, and then look at the KB concepts that are formalized from that section.

The association between the document and the formalized knowledge can also be used during explanation generation. The current system does not leverage this possibility, but we plan to investigate this in our future work.

Capturing Conceptual Knowledge

Our CIs suggested that it will not be feasible to expect the users to learn a formal KR language such as KM. Therefore, we needed to come up with a notation that is easy to understand, and yet facilitates acquisition of multiple forms of knowledge.

We settled on a graphical notation called Concept Maps or CMAPs. We chose CMAPs because they are effective as a visual tool for formalizing and representing a range of knowledge types, and have even been shown to be used effectively in education [15]. User tests with paper prototypes confirmed that CMAPs will indeed be a viable approach for SMEs to capture conceptual knowledge in the three science domains.

In AURA, the users define new concepts by first positioning a concept in the existing taxonomy, and then creating a CMAP for it by choosing existing concepts from the palette and connecting them to each other using basic graph manipulation operations such as add, connect, and equate nodes. In Figure 4, we show a graph that captures the concept of *Eukaryotic-Cell*. Each node in this graph, shown using a black border, represents an individual, and each labeled edge between two nodes, with the label shown in a white rectangle with no border, a relationship between them.

We logically interpret the graph as follows: for every instance of *Eukaryotic-Cell*, there exists an instance of *Nucleus, Chromosome*, and *Plasma Membrane*, such that each is in *has-part* relation to *Eukaryotic-Cell* and the *Chromosome* is inside the *Nucleus* [4].

```
(forall ?c
    (=> (instance-of ?c Eucaryotic-Cell)
        (exists ?x ?y ?z
         (and
            (instance-of ?x Nucleus)
            (instance-of ?y Chromosome)
            (instance-of ?z Plasma-Membrane)
            (has-part ?c ?x) (has-part ?c ?y)
            (has-part ?c ?z) (is-inside ?y ?x)))))
```

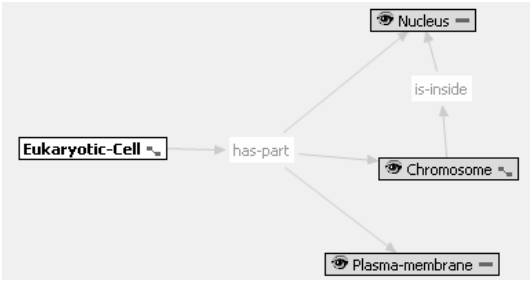

Figure 4. Graphical representation of a Eukaryotic cell.

We have extended this basic notation in several different ways to capture additional forms of knowledge: (1) constraints, such as a eukaryotic cell contains exactly one nucleus; (2) conditionals, e.g., if an ionic compound is insoluble in water, it will appear on both sides of an ionic equation, and (3) values, e.g., an aqueous solution of a strong acid has a qualitative *PH value* of *acidic*. We have omitted a description of these interfaces here because of space limitations.

While we leveraged our prior work with the SHAKEN system in capturing conceptual knowledge, the specific user interaction of the system was enhanced substantially through user-centered design. Once we had tested the basic design of the system on paper, we built several Flash prototypes. These prototypes were not connected to the KM KB, and were directed at gathering information about user interaction.

We found that the users tended to use a left-to-right layout for showing graphs as opposed to the top-to-bottom layout that was used in the SHAKEN system. The left-to-right layout also made it easier to do a cleaner presentation of the graph if the node labels became too long. In a top-to-bottom layout, the long label names caused a greater disruption in the presentation of the graph. We fine-tuned the controls for performing operations, such as connect, expand, and contract, so that they could be applied directly to the nodes instead of visiting a menu as it

was necessary in the SHAKEN system [4]. For example, in Figure 4 a node may be hidden simply by clicking on the eye symbol. A partial tree symbol, as shown on *Eukaryotic-Cell* indicates that all of its descendants are not visible on the screen. A rectangular symbol next to *Plasma-Membrane* indicates that none of its descendants has been shown. These controls were a result of significant experimentation with the flash prototype of CMAPs.

Capturing Mathematical Equations

The CIs showed that the users had no preference for a particular tool for creating equations (save by hand). In their current work, the only time they used any kind of equation editor was when preparing a technical paper. They used LaTeX [16] or MathType [17], depending on the submission requirements of the paper. One user had familiarity with Mathematica [18], but this was for a specific course he had taken and he had not used it since then.

We performed a competitive analysis of the equation editors mentioned during the CIs. Three of the systems (LaTeX [16], MathType [17], and Equation Illustrator V [19]) were equation typesetting tools, and did not include any mathematical parsing of the equations. The systems that could parse equations (Scientific Notebook [20] and Graphing Calculator [21]) provided a much more complex feature set than needed by AURA. Two other applications (Mathematica [18] and Maple [22]) had excellent expressivity and ability to perform complex mathematical operations, but were rich programming environments, and too complex to integrate with AURA. While the analysis of existing tools showed that none could be easily integrated into AURA, they had features we felt were useful: WYSIWYG entry of equations, and automatic formatting.

Since the CIs had shown that the users preferred a natural representation of the equations, we designed an equation editor in which the users could directly enter an equation. The analysis of existing tools uncovered common key sequences across some of the applications, such as a caret (^) to indicate an exponent. We incorporated these standards to leverage any previous experience that the users may have had with equation editors. As the user types in an equation, the system echoes back the equation by showing its formatted version. We wanted to integrate the expressiveness of equations into the CMAP interface, while adding a minimal amount of complexity. We did this by expanding the link and node metaphor to attach meaning to the equation variables. The users enter an equation, using any variables that they prefer. The system recognizes the variables and highlights them until they have been mapped to a node in the graph. To complete this mapping, the user simply drags a link from the appropriate node to the variable. This needs to be done only once per variable; other occurrences of that variable within the same CMAP will be automatically understood by the system. Once a node is assigned to a variable, the node is labeled with the variable, and connected to the equation with a line (see Figure 5).

Logically, the equation in Figure 5 is associated with *Move* as a slot value, and inherits to its subclasses and instances. Whenever there is a question that involves querying some slot value for *Move,* for example, a query to determine the value of slot *force,* the reasoner collects all the equations associated with *Move,* computes the known values for the variables that appear in those equations, *acceleration* and *mass* in this case, and exports them to an external equation solver. The equation solver performs the computation and, if successful, returns the result [10].

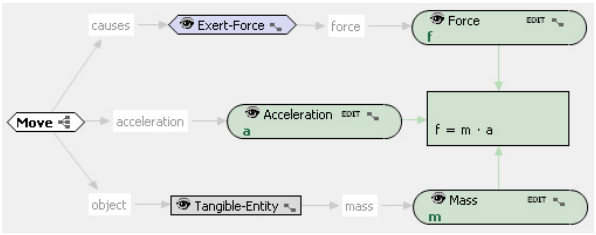

Figure 5. Example concept map with equations.

Early user tests with the implementation of the equation editor showed that the auto-formatting and error checking caused major problems. The initial implementation rigorously parsed the user input to ensure that entered equations were mathematically sound. However, users were continuously frustrated by the lack of flexibility permitted by this parser. Users wanted to include partial equations for later elaboration, and to create incomplete or conflicting versions of the equations as a note to themselves. In short, they wanted the flexibility of paper. Based on this feedback, we loosened the error checking to accommodate unexpected entry patterns. Tests on the final version showed that revising equations was much easier, and the users appreciated the flexibility in entering equations.

Capturing Tabular Knowledge

We undertook a survey of the tables found in the three textbooks. In general, tables can have an arbitrary nesting structure. However, the tables in the textbooks we were working with made infrequent use of nesting. Therefore, we did not consider the nesting in our design, and assumed that wherever nesting appears in the table, we can train the SME to flatten the table. We identified three major types of tables depending on the kind of logical representation they will generate if we were to encode the knowledge appearing in them.

The first type of table, an *object-level* table, results in one or more classes or individuals in the KB. For example, the Chemistry textbook contained a table with two columns: one containing examples of combination reactions, and the other containing an example of decomposition reaction. The representation of this table in the KB will have the definition of two classes (combination reaction and decomposition reaction) and a representation of each of the example reactions.

The second type of table, a *value-level table*, results in slot values for instances of classes. For example, for the table shown in Figure 6, each entry in the table results in an assertion in the KB specifying the value of the solubility slot for instances of a class of compounds. We represent each exception as an additional row in the table.

TABLE 4.1 Solubility Guidelines for Common Ionic Compounds in Water

Soluble Ionic Compounds		Important Exceptions
Compounds containing	NO_3^-	None
	$C_2H_3O_2^-$	None
	Cl^-	Compounds of Ag^+, Hg_2^{2+}, and Pb^{2+}
	Br^-	Compounds of Ag^+, Hg_2^{2+}, and Pb^{2+}
	I^-	Compounds of Ag^+, Hg_2^{2+}, and Pb^{2+}
	SO_4^{2-}	Compounds of Sr^{2+}, Ba^{2+}, Hg_2^{2+}, and Pb^{2+}
Insoluble Ionic Compounds		**Important Exceptions**
Compounds containing	S^{2-}	Compounds of NH_4^+, the alkali metal cations, and Ca^{2+}, Sr^{2+}, and Ba^{2+}
	CO_3^{2-}	Compounds of NH_4^+ and the alkali metal cations
	PO_4^{3-}	Compounds of NH_4^+ and the alkali metal cations
	OH^-	Compounds of the alkali metal cations, and Ca^{2+}, Sr^{2+}, and Ba^{2+}

Figure 6. Solubility table from a chemistry book.

The third type of table, *qualitative ordering*, specifies qualitative order among the values of a specific slot. For example, in Figure 1, we illustrated a table from a Chemistry book that shows the relative ordering of the strengths of various acids and bases. In this table, the acid strength increases from bottom to top, and the base strength increases from top to bottom.

Both object-level and value-level tables provide a way of viewing knowledge that may be spread across the KB. The value-level table of Figure 6 collects the solubility values of different compounds in one place. These values could be captured through the CMAP interface even though a user would need to visit each compound (e.g., nitrate, acetate). The qualitative ordering table, however, is unique in that the order of acid and base strength is not enumerated in the text.

There are at least two approaches to capture the knowledge about qualitative orders. The first approach is to extend the equation editor so that a user can enter qualitative equations. The second approach is to design a table interface in which a user can specify a qualitative order. With the objective of keeping the interface as close as possible to the natural form in which the knowledge appears in a textbook, we preferred to design a table-based interface to capture knowledge about qualitative orders.

Based on this analysis, our design for tables was primarily aimed at acquiring knowledge that involves specifying a qualitative order in a table. In the knowledge capture interface for a qualitative ordering table (described below), if no qualitative ordering is specified, the table reduces to a value-level table.

There are automated techniques for extracting data from tables [23, 24], but they work well only when there is a large number of tables with regular structure—an assumption that breaks down in textbooks.

Let us first consider some representational aspects of the table in Figure 1. The first column gives the qualitative value of the strength of an acid (e.g., *HCl* is a strong acid), and the last column gives the qualitative value of the strength of the base (e.g., Cl^- has negligible base strength). The second and third columns in Figure 1 specify the conjugate acid and base relationship between an acid and base. We can view the table of Figure 1 as a combination of three traditional tables, the first comprising the first two columns and asserting the qualitative value of the strength of an acid, the second comprising the second and third columns and asserting the conjugate base pair relationship, and the third comprising the third and fourth columns and asserting the strength of a base. With this analysis, we can represent this table as three separate tables.

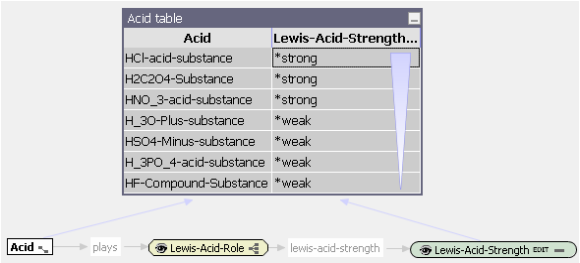

Figure 7. Capturing the first two columns of the table in Figure 1.

We illustrate in Figure 7 our approach to capture the first two columns of the table in Figure 1. At the bottom of Figure 7, we show a fragment of a CMAP for an acid. We allow a user to add a table to this CMAP. The first column of the table is always linked to the root node of the CMAP, and can be populated only by the subclasses of the class that is the type of the root node. The user can add additional columns to the table and must relate them to the nodes on the graph. In this case, the user relates the second column to the property *lewis-acid-strength*. The user specifies the qualitative values of the acid strength as **strong* or **weak*, and then adds a qualitative order suggesting that the values increase from bottom to top. As an example of the logical axioms captured from this table, consider the axiom generated by the first row.

```
(forall ?x
    (=> (instance-of ?x HCl-acid-substance)
        (exists ?y ?z
         (and
            (instance-of ?y Lewis-Acid-Role)
            (instance-of ?z Lewis-Acid-strength)
            (plays ?x ?y)
            (lewis-acid-strength ?x ?z)
            (magnitude-scalar ?z *strong))))
```

The qualitative order in the table results in an axiom of the following form:

```
(and
```

(instance-of Property-Order-1 Property-Order)
(property-slot Property-Order-1 lewis-acid-strength)
(elements Property-Order-1 '(
HCl-acid-Substance H2C2O4-Substance
HNO3-acid-substance H_3O-Plus-substance
HSO_4-Minus-substance H_3PO_4-acid-substance
HF-Compound-Substance))

We represent the qualitative order as an individual instance in the KB, and specify the slot the order applies to, and the classes for which the slot values of their instances satisfy the specified ordering. The reasoner then uses the qualitative order to perform inferences that involve determining relative strengths of acids or bases.

It is possible to draw equivalent qualitative inferences if the quantitative values of the Lewis acid strength for each of the acids were known. Implementing such reasoning is the subject of future work. Our domain analysis had shown that the qualitative information is used often in the science textbooks, and in some cases, no quantitative values are available. Thus, the tabular knowledge capture interface considered here is indispensable.

INTERMEDIATE EVALUATION

The question that we want to answer with the evaluation of the AURA system is: Can we build a knowledge capture system to enable domain experts to construct KBs from the knowledge found in science textbooks and use the resulting KB for deductive question answering?

The paper prototype studies that we reported earlier were designed to get preliminary insights into the designs. The goal of the intermediate evaluation was to assess how AURA would fair under longer-term use.

We conducted a test of the system with two knowledge formulation SMEs in each of the three domains. The SMEs were graduate students in their respective fields at a local university. They underwent training for 40 hours on how to use the system, and then were left alone for 80 hours to formulate knowledge and test it by asking the questions. We chose the current training and experience requirements based on what we considered achievable for a near-term evaluation. We designed the training as a mixture of lecture on the system's capabilities, hands on exercises, and independent exercises. One of our long-term goals is to significantly reduce the training time.

We asked each SME to encode knowledge from the selected syllabus from the respective domain. As the SMEs encoded knowledge, they tested the knowledge by posing test questions. We provided the SMEs with a suite of test questions that they could use for this purpose.

During the evaluation period, the Chemistry and the Biology SMEs authored 150 to 200 concepts, and the Physics SMEs authored 6 to 12 concepts using the CMAP interface. The number of concepts authored in Physics was much lower because much of the Physics knowledge is in the form of equations, and can be represented in a small number of concepts. On average, it took the SMEs 22 minutes to author each concept. This measurement was done based on the time that the SMEs spent actually editing a concept. Any time used on testing or other activities was not included. The Physics SMEs completed their work much sooner than the total of 80 allocated hours.

The SMEs made extensive use of the semantic search in formulating the concepts. They did not make as extensive use of selecting the text from the document and invoking the semantic search. Instead, they typed the search terms directly in the dialog box.

As expected, the equation editor was most relevant in the Physics domain, where nearly all questions required AURA to reason with equations to provide an answer. The Physics SMEs added approximately 25 equations with an average equation creation time of 2 minutes.

Despite its success, lack of features in the equation editor was apparent. We were aware of the need for some features from the domain analysis, but had not implemented by the time of the intermediate evaluation. For example, a Physics equation such as $v=½\cdot a.t$ can be applicable to multiple dimensions. We can write the same equation as $v_y=½\cdot a_y t$, $v_x=½\cdot a_x t$ for y and x dimensions, respectively. The current system does not support a facility to capture such knowledge.

The SMEs authored six tables for Chemistry. As a specific example, they authored the table from Figure 6 by simply flattening it. They created two separate concepts—one representing soluble ionic compounds, and the other representing insoluble ionic compounds. To each of those concepts, they added a table listing the compounds that were respectively soluble or insoluble. For each exception, they created a new row. By not supporting nesting, the design was much easier to use and understand, but at the cost of being more verbose than the table of Figure 6.

We tracked all the help requests from the SMEs. Over a total of 480 hours of KF time, there were 67 help requests, giving an average of one request every 7.2 hours. Approximately half of the requests could be resolved by pointing the SMEs to the relevant sections of the training material. While this number is not non-zero, it does suggest that SMEs were able to work unassisted for a significant length of time without encountering any need for help. During the debrief sessions, the SMEs commented that they could have been equally effective with much less training than 40 hours. These data provide positive evidence in support of the individual KF methods.

To ensure the efficacy of the authored knowledge, the KEs tested the knowledge formulated by SMEs on several test questions that were not known ahead of time. In all three domains, the knowledge authored using CMAPs was extensively used in answering questions. In Chemistry, the system answered questions about the solubility values entered through the table editor. Every question in Physics involves the use of equations. The answers to the

test questions received a correctness score of 21% in Physics, 40% in Chemistry, and 51% in Biology.

In summary, the ability of SMEs to use AURA to author knowledge, with little need for help, and the correctness scores give us evidence that the design we have presented here is on the right track. Clearly, the evaluation is preliminary, but with this evaluation we have collected specific action items for improving the system, after which we will make a more large scale evaluation.

SUMMARY

We have reported on the current status of a knowledge capture system, AURA, in the context of a long-term initiative to build an application called Digital Aristotle that can answer questions on a wide variety of science topics and provide user- and domain-appropriate explanations. Our work, however, is not complete, and we are just beginning the next phase of the effort to refine the system. The first class of enhancements to the knowledge formulation capability of the system will be within the current framework: capturing knowledge that involves performing algorithmic computation and detailed representation of processes, addition of a knowledge debugging and validation facility, the ability to propagate changes when knowledge is revised, and a facility to author knowledge necessary for explanation generation. The second class of enhancements will extend the current architecture to investigate the possibility of interfacing AURA to a Semantic Wiki [25] to test if some of the knowledge needed for answering questions (e.g., ground facts, taxonomy) could be acquired from contributors given little or no training.

ACKNOWLEDGMENT

The work reported here was funded by Vulcan Inc. We thank the members of the AURA development team at UT Austin, and Boeing for their contributions to the effort.

REFERENCES

[1] Giancoli, D.C., *Physics Principles with Applications.* 2004: Benjamin Cummings.

[2] Brown, T.L., et al., *Chemistry: The Central Science. 2003,* New Jersey: Prentice Hall.

[3] Campbell, N.A. and J. Reece, *Biology,* Sixth Edition. 2001: Benjamin Cummings.

[4] Clark, P., et al., Knowledge Entry as the Graphical Assembly of Components, in *Proc. 1st Int. Conf. on Knowledge Capture (K-Cap'01). 2001*. p. 22-29.

[5] Forbus, K., et al., CyclePad: An articulate virtual laboratory for engineering thermodynamics. *Artificial Intelligence,* 1999. 114: p. 297-347.

[6] Novak, G. and J.K. Hyung, Representation of Models for Expert Problem Solving in Physics. *IEEE Transactions on Knowledge and Data Engineering,* 1991. 3(1): p. 48-54.

[7] Barker, K., et al., A Knowledge Acquisition Tool for Course of Action Analysis, in *Innovative Applications of Artificial Intelligence Conference. 2003*. p. 34-50.

[8] Blythe, J., et al., An Integrated Environment for Knowledge Acquisition, in *Int. Conf. on Intelligent User Interfaces.* 2001. p. 13-20.

[9] Clark, P. and B. Porter, KM – *The Knowledge Machine: Users Manual.* 1999 [The system code and documentation are available at http://www.cs.utexas.edu/users/mfkb/km.html.]

[10] Clark, P., et al., Capturing and Answering Questions Posed to A Knowledge-Based System. 4*th Int. Conf. on Knowledge Capture (K-Cap'07). 2007*.

[11] Bass, L. and B. John, Linking Usability to Software Architecture Patterns through General Scenarios. *Journal of Systems and Software.* 66(3): p. 11.

[12] Beyer, H. and K. Holtzblatt, Contextual Design. *Interactions*, 1999. 6(1): p. 32-42.

[13] Miller, G.A., et al., Five Papers on WordNet. 1993.

[14] Barker, K. B., et al., A Library of Generic Concepts for Composing Knowledge Bases, in *Proc. 1st Int. Conf. on Knowledge Capture (K-Cap'01).* 2001.

[15] Canas, A.J., et al., Using Concept Maps with Technology to Enhance Cooperative Learning in Latin *America. Science Teacher,* 2001.

[16] Latex. [Available from: http://www.latex-project.org.]

[17] MathType, [Available from http://www.dessci.com/en/products/mathtype/.]

[18] Mathematica, [Available from http://www.wolfram.com/products/mathematica/.]

[19] Illustrator, [Available from http://www.equation-illustrator.com/.]

[20] Notebook, [Available from http://www.mackichan.com/products/snb.html.]

[21] Graphing Calculator, [Available from http://www.pacifict.com/Products.html.]

[22] Maple. [Available from http://www.maplesoft.com.]

[23] Pivk, A., et al., Transforming Arbitrary Tables into F-Logic using TARTAR. *Data and Knowledge Engineering,* 2006. Accepted for publication.

[24] Lerman, K., C. Knoblock, and S. Minton, Automatic Data Extraction from Lists and Tables in Web Resources. in *Workshop on Automatic Text Extraction and Mining Workshop.* 2001. Seattle.

[25] Völkel, M., et al., Semantic Wikipedia, in *15th International World Wide Web Conference.* 2006: Edinburgh, Scotland.