

# Developing an Intelligent Tutoring System Using Natural Language for Knowledge Representation

Sung-Young Jung<sup>1</sup> and Kurt VanLehn<sup>2</sup>

<sup>1</sup>Intelligent Systems Program, University of Pittsburgh, Pennsylvania, USA,  
syjung5@asu.edu

<sup>2</sup>School of Computing Informatics and Decision Systems Engineering, Arizona State University, Arizona, USA, Kurt.Vanlehn@asu.edu

**Abstract.** Authoring the domain knowledge of an intelligent tutoring system (ITS) is a well-known problem, and an often-mentioned approach is to use authors who are domain experts. Unfortunately, this approach requires that potential authors learn to write and debug knowledge written in a formal knowledge. This paper presents the design of an authoring system, 'Natural-K', in which domain authors including non-programmers are able to add background knowledge in natural language.

**Keywords:** Authoring System for ITS, Knowledge acquisition, Knowledge representation using natural language, Natural language understanding.

## 1 Introduction

Intelligent tutoring systems (ITS) require authors to add a several types of knowledge including problem statements, principles, commonsense knowledge, etc. A difficulty in developing such systems is in how to acquire such knowledge from authors who are usually not computer programmers. When authors have added knowledge, they must be able to follow problem solving detect errors, explore fixes and finally updating the knowledge. But this has not been possible so far without help of a knowledge engineer because knowledge has been represented in formal language.

For the ITS community, where tutoring systems often have a finite set of problems and authors often merely want to add more problems, the ability to represent problem statements in natural language is particularly appealing. ISSAC 1, MECHO 3, and pSAT 4 receive a problem statement written in natural language text and then transform into formal language to solve the given problem. Although these projects demonstrated that the problem statements read by students could also be read by the tutor, this capability was limited because the background knowledge required for translating the problem statements was encoded in formal language or computer program code. Only computer programmers were able to add such knowledge.

Several projects have explored the acquisition of inferential domain knowledge from natural language. The basic idea was to transform natural language into formal representations such as logic forms 7 or semantic networks 8, and the systems perform inferences to accomplish a problem solving goal. Domain knowledge includes a great deal of commonsense knowledge. Several universal commonsense knowledge bases are being developed (Opencyc 2). However, it is unlikely that the complete universal set of commonsense knowledge can be built soon.

The main idea presented in this paper is to have the system represent knowledge in natural language. Thus, authors including non-programmers can add knowledge, detect bugs, and fix them without help of a knowledge engineer.

## 2 A Problem Statement and Principles

The specific project is to replace the knowledge representation language of Pyrenees with natural language, then use Natural-K to re-author the Pyrenees knowledge base. Pyrenees is an intelligent tutoring system for equation-based problem solving 5 with two main types of domain knowledge: principles and problem statements (Fig. 1). The principle in the figure was modified using a natural language definition for each variable (ex: “*the direction of vector at time t*”).

(a)	Problem skateboarder: “A skateboarder rolls at 1.3 m/s up an inclined plane angled at 143 degrees. What is her vertical velocity?”
(b)	<pre>Pa_equation (along(projection(offaxis, Vector, T), Axis)), V_x=V*Trig) :-     match (Axis, 'xy_component', [xy_/XY]),     match(V_x, 'xy_component of vector_ at time t_', [xy_/XY, vector_/Vector, t_/T]),     match(V, 'the magnitude of vector_ at time t_', [vector_/Vector, t_/T]),     match(V_dir, 'the direction of vector_ at time t_', [vector_/Vector, t_/T]).</pre>

**Fig. 1: A problem example. (a) A problem statement. (b) A relevant principle.**

Often an author wants to add new domain problems using existing principles. Then, he needs to supply (1) a problem statement, and (2) commonsense knowledge (or inference rules), always in natural language to the system.

## 3 Mapping to Standard Natural Language

The initial problem statement will be translated into precise natural language until the resulting language exactly matches the variables in a principle, which are also written in natural language as Fig. 1-(b); the principle then applies and produces an equation. The precise natural language that appears in the principles is decided upon by the knowledge engineer and is called *standard natural language*. Many different non-standard phrases all get converted into the same standard sentence (for example, ‘A skateboarder rolls at 1.3 m/s’ and ‘The speed of the skateboarder is 1.3 m/s’ into ‘The magnitude of the velocity of a skateboarder is 1.3 m/s’).

Mapping non-standard natural language to standard natural language is done by inference rules. Rules are entered by authors as two pieces of natural language. The left side of the rule corresponds to a condition, and the right side of the rule corresponds to new knowledge produced.

‘A skateboarder rolls at 1.3 m/s up a plane’  
*implies* ‘the magnitude of the velocity of the skateboarder is 1.3 m/s’.

The strings entered by authors are represented inside the authoring system as dependency graphs after parsing 7. The parser does syntactic normalization for passive and active sentences. The translation process starts with a set of dependency graphs that represent the problem statement. Inference rules run in a forward chaining manner, augmenting the set with more dependency graphs until no new one can be added. Then, the variables of the principles are matched against the set of dependency graphs and produce equations.

As the amount of added rules increases, the system can automatically produce generalized rules. For instance, after entering the rule above and this new one:

*'A sled glides at 0.2 mph up a hill'*  
*implies 'The magnitude of the velocity of the sled is 0.2 mph'.*

the authoring system would produce a generalized rule with *semantically constrained variables* (ex: *object\_*, *moves\_*, etc). The generalization is driven by WordNet's ontology 6. If the system finds a hypernym (a superclass word), such as "object" which is the least common ancestor of "skateboarder" and "sled", then the system produces a generalized rule with the semantically constrained variable, *object\_*. In this way, the system generalizes the existing rule so that it will subsume both the old rule and the newly entered one.

*'An object\_moves\_ at num\_unit\_ up a hill\_ '*  
*implies 'the magnitude of the velocity of the object\_ is num\_unit\_ '.*

#### 4. Discussion and Further Works

An important issue in using natural language is that there can be a large number of different expressions that have the same meaning. This is called *the paraphrasing problem* 9. An important thing that should be noted is that it is not a serious issue in an authoring system for ITS. Authoring tasks in ITS is task-specific. It is enough for an author to add only the knowledge required to solve a given problem. Thus, the required set of knowledge to recognize the paraphrases is fixed and determined by the given problem statement. In other words, the author doesn't have to make the system recognize all paraphrases not seen in the given sentence. For this reason, the issue of paraphrases is not a serious problem except the case of fully automatic text processing without help of human authors.

So far, the authoring system succeeded in adding 15 physics problem statements correctly; 98 inference rules were added for them. It showed that the average number of rules per a problem was around seven (=98/15) which is small enough for an author to add. The remaining work is to implement rule generalization, integrate with Pyrenee, and perform empirical evaluation of the system.

#### References

1. Novak, G. S. Jr., Representations of Knowledge in a Program for Solving Physics Problems', *IJCAI-77*, Cambridge, MA, pp. 286-291 (1977).
2. Opencyc.org, OpenCyc Tutorial, <http://www.cyc.com/cyc/opencyc/overview>.
3. Bundy, A., et. al., Solving mechanics problems using meta-level inference, *IJCAI-79*, pp. 1017-1027 (1979).
4. Ritter, S., et. al., Authoring Content in the PAT Algebra Tutor, *Journal of Interactive Media in Education*, (9), pp. 1-30 (1998).
5. VanLehn, K., et. al. Implicit versus explicit learning of strategies in a non-procedural cognitive skill. *The Intern'l Conf' on Intelligent Tutoring Systems*, pp. 521-530 (2004).
6. Harabagiu, S. M., Miller, G. A., Moldovan, D. I., WordNet 2 - A Morphologically and Semantically Enhanced Resource, *SIGLEX 1999* (1999)
7. Jurafsky, D., Martin, J. H. *Speech and Language Processing*, Prentice Hall (2000)
8. Sowa, J. F., Current Issues in Semantic Networks, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Morgan Kaufmann (1990).
9. Ifene, A., *Textual Entailment*. Iași, Romania: PhD Thesis, Computer Science, University of Iasi (2009).