

# A Rule Editing Tool with Support for Non-Programmers in an Ontology-Based Intelligent Tutoring System

Eric Wang, Sung Ah Kim, and Yong Se Kim

CREative Design & Intelligent Tutoring Systems (CREDITS) Research Center  
Sungkyunkwan University  
Suwon 440-746 Korea

wang@me.skku.ac.kr, sakim@skku.edu, yskim@me.skku.ac.kr

## Abstract

We are developing an ontology-based intelligent tutoring system, in which domain, pedagogical, and tutoring knowledge is represented as ontologies. Inference rules are a key representation of pedagogical knowledge for automated evaluation. However, the use of existing rule-based languages requires programming skill. Rule editing can be made more widely accessible to non-programmers through the development of smarter tool support. Through analysis of inference rules used in an existing intelligent tutoring system, we identify common idioms within rules that are simple to express in natural language, but which vary widely in their complexity of implementation in the Jess rule language. We have developed a prototype rule editing tool in which these idioms are provided as keywords, with automatic translation to complete rules, which simplifies the rule editing process.

## 1 Introduction

We have previously developed a multimedia application called Visual Reasoning Tutor (VRT) [Hubbard *et al.*, 1996], which uses the *missing view problem* as a mechanism to develop the visual reasoning abilities of design and engineering students, as shown in Figure 1.

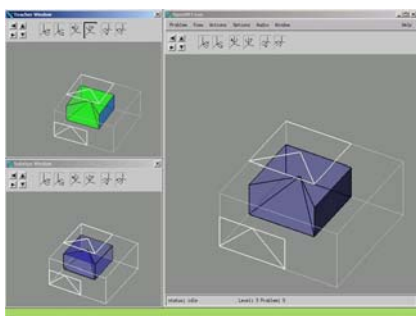


Figure 1. Visual Reasoning Tutor

From 2001–2003, we have developed a successor system called Intelligent VRT (IVRT), which embeds VRT within an intelligent tutoring system framework. IVRT uses

manually-encoded learning contents consisting of *skills, lessons, and problems*, and a simple learner model that records student skill scores and activity history.

## 2 Ontology-Based Intelligent Tutoring System

Ontologies support knowledge sharing and reuse, by both humans as well as computers [Gruber, 1993]. We have migrated to an ontology-based approach to intelligent tutoring systems, shown in Figure 2. Domain knowledge (learning contents), learning process knowledge, tutoring knowledge, and learner information are formalized as ontologies. Pedagogical knowledge is also represented as inference rules, which are executed at run-time by a separate inference engine. We use the Protégé ontology editor [Protégé, 2004] with OWL plugin, and Jess [Friedmann-Hill, 2003] as the rule inferring engine, with XSLT conversion from OWL to Jess.

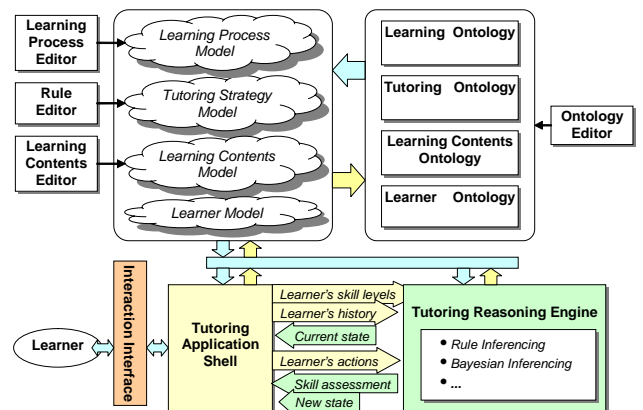


Figure 2. System architecture of ontology-based intelligent tutoring system

## 3 Pedagogical Rules in IVRT

IVRT's teaching strategy is to show a subset of lessons and problems based on the learner's current skill scores. Items that the learner has already mastered, and items for which the learner is not yet ready, are not shown. As the learner solves problems satisfactorily, the learner's skill scores

increase, which causes new lessons and problems to become visible, and previous ones to be hidden.

This high-level strategy is implemented using inference rules. Terms used in these rules are defined in our learning contents ontology for the IVRT domain, as follows:

- A skill has an *activeness* property, which is true or false, and zero or more *required* skills, arranged in a hierarchical structure.
- A lesson or problem has one or more *associated* skills, and has a *visibility* property, which is either true (it is shown to the learner) or false (hidden).
- A global property of *skill satisfaction* is defined by a system predicate, which takes a skill and returns true or false. Each teacher can customize this test.

Selected rules are shown below. These rules are given in a quasi-formal manner using natural language, which was to ease discussion of the rules without requiring Jess expertise.

#### *Rules to Activate Skills*

1. For each skill: if it has no required skills, activate it.
- 2a. For each skill: if **any** required skill is not satisfied, deactivate this skill.
- 2b. For each skill: if **all** required skills are satisfied, activate this skill.

#### *Rules to Show and Hide Lessons*

- 3a (Default strategy) For each lesson: if **all** associated skills are active, show it.
- 3b (Default strategy) For each lesson: if **any** associated skill is inactive, hide it.
- 4 (Special strategy) For each lesson: if **any** associated skill is active, show it.

#### *Rules to Show and Hide Problems*

- 5a For each problem: if **all** associated skills are active, show it.
- 5b For each problem: if **any** associated skill is inactive, hide it.

### 3.1 Implementation 1: Rules in Jess

A full implementation of IVRT's strategy is straightforward in a standard rule language such as Jess, totaling about 20 Jess rules. However, this requires a Jess programmer's skill. This tends to exclude any users who are not skilled Jess programmers. We assume most teachers lack sufficient programming skill to rely entirely on this approach.

### 3.2 Implementation 2: SWRL Ontology using Protégé OWL Interface

We have modeled a subset of SWRL rule syntax [Horrocks *et al.*, 2004] as an OWL ontology in Protégé, by defining SWRL terms as classes, and SWRL grammar rules as properties of these classes. Then we were able to use Protégé OWL's user interface to construct instances of SWRL rules, i.e. as a rudimentary rule editor. We integrated this interface with Jess using XSLT conversion, so that SWRL rule instances edited in Protégé are immediately updated in the Jess run-time environment.

This approach was successful insofar as it gave us a rule editing capability. However, it faced two severe drawbacks:

1. *Verbosity*. A SWRL rule naturally has a hierarchical structure. To instantiate such a rule as an instance of an ontology required instantiating every element and subexpression separately, in a bottom-up manner. This was a tedious process, even for trivial rules.
2. *Programming skill*. To create rules that would work properly after conversion to Jess still required expertise in Jess. Hence, we judged this approach to be no simpler than programming in Jess directly.

## 4 A Rule Editing Tool with Support for Non-Programmers

We have identified as a desideratum within our ontology-based intelligent tutoring system environment to make rule editing *accessible to non-programmers*. That is, it should provide intelligent support to hide or reduce the complexity of programming.

### 4.1 Identification of Common Idioms

We considered the actual rules used in IVRT, and also plausible rules within a typical teaching strategy, i.e. which could reasonably be expected to be reused by many teachers across many domains. When these rules are written at a fairly abstract level, using natural language, certain idioms emerged. These idioms correspond to everyday concepts in natural language, on which most people can agree at a non-technical level.

Two common idioms used throughout IVRT's rules are "if any" and "if all", as highlighted in bold font in Section 3. When implemented in Jess rules, they require substantially different techniques, due to Jess's own characteristics.

### 4.2 Mapping of Idioms to Rule Fragments

For each idiom, we define a mapping to a Jess rule fragment, which is a portion of a Jess rule. A rule fragment could be as simple as a single keyword<sup>1</sup> in the language. More generally, it consists of a block structure within a rule, and it may introduce variables, or even multiple rules.

We have identified the following idiom-to-fragment mappings. For each mapping, we show the idiom in quasi-formal natural language in italics, followed by its Jess rule fragment. Unimportant details of Jess rule syntax are shown in gray text.

- *for each s in a set S*  
(defrule R1 (S ?s) => ...)

"*for each*": This idiom expresses a simple iteration over a set *S* of facts. As this is a fundamental operation in any rule language, Jess performs this iteration implicitly, without

---

<sup>1</sup> This correspondence reflects the fact that the design of a programming language itself involves a choice among a range of possible programming idioms, and the idioms chosen will thereafter be trivial to use within that language, by design.

requiring any language keyword. Hence, it suffices to just specify the set itself, using one pattern (S ?s), where ?s denotes a Jess variable.

- *for each s in set S that satisfies property P*  
(defrule R2 (S ?s) (P ?s) => ...)

To restrict this iteration to a subset of S that satisfies an additional property P, we simply add a second pattern for P.

- *if any s in set S satisfies property Q*  
(defrule R3 (S ?s) (exists (Q ?s)) => ...)

“if any”: This idiom differs from “for each” in that we don’t need to visit each element that satisfies the property. Instead, we halt the iteration as soon as any one succeeds. This idiom maps to the Jess keyword **exists**.

- *if all s in set S satisfy property Q*  
(defrule R4  
; Let all other rules go first before checking the ‘not’  
(declare (salience -1))  
; Guard (to ensure volatility, and to exclude empty set)  
(and (S ?s1) (Q ?s1))  
; All  
(not (and  
(S ?s2) (~Q ?s2) ; ~Q is the negation of property Q  
)  
)  
=> ...)

“if all”: This idiom is also an iteration over a set. However, by its nature, it *must* visit every element of the set. Jess does not implicitly handle this operation. We apply De Morgan’s law to convert the *all* idiom to *not any*, which Jess does provide as primitives. Hence, this idiom expands to a block structure, using the Jess **not** and **and** keywords.

Jess’s **not** keyword introduces three complications.

1. *Volatility.* The pattern immediately preceding a **not** must be volatile: the rule will be re-checked only when that pattern changes. We handle this by adding a guard clause before the *all* block.
2. *Empty set.* **not and** gives a false positive for an empty set. The guard clause also prevents this.
3. *Temporal dependency.* **not** assumes that all other rules have reached a quiescent state (i.e. are no longer changing any facts). This requires temporal ordering among rules, which maps to a Jess **salience** declaration.

### 4.3 Simplified Syntax with Idioms as Keywords

We have defined a simplified rule syntax, in which the common idioms appear as keywords. This supports a non-programmer who thinks at the level of the natural-language idioms, by hiding their implementation details. As this rule format is essentially text-based, any text editor would be sufficient in theory. For added convenience, we have developed a graphical front-end using Java Swing, shown in Figure 3, which allows selection of the idiom keywords from menus.

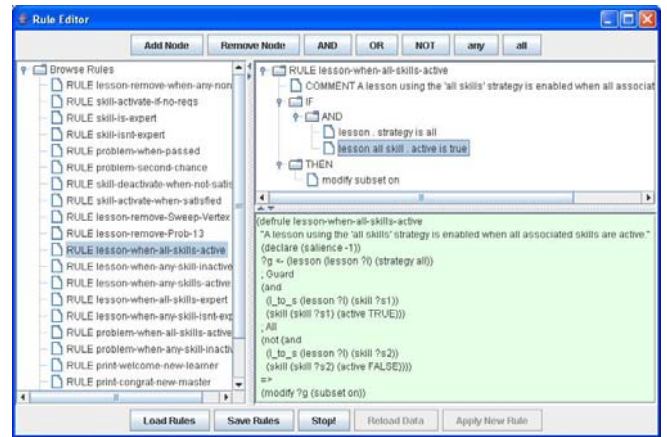


Figure 3. Rule editor interface with conversion to Jess

Rules written in this simplified syntax are mapped into Jess rule fragments, using a standard recursive descent parsing approach. The fragments are then merged into complete Jess rules, and are immediately evaluated, which updates Jess’s run-time environment.

## 5 Pedagogical Knowledge Editing in ITS Development

We are integrating the rule editing capability into a distributed, persistent ITS development framework. In this framework, the ontologies serve as repositories for many learning domains, tutoring strategies, and bodies of pedagogical knowledge, accumulated over time and across many individual teachers and courses. Rule editing is then a subtask of the more general operation of *pedagogical knowledge editing*. Each individual teacher composes her own tutoring strategy model from the tutoring ontology and a library of previously-developed inference rules, which exploits knowledge reuse. A teacher can customize her model by defining her own pedagogical knowledge as new inference rules. We support local extensions to a particular model, as well as extension of the ontologies themselves by using an ontology editor.

The ontologies and tutoring strategy models are accessible over the web, using standard web services. For distributed rule editing, we are exploring the further enhancement of the rule editing tool as an embedded Java applet, or a separate web-enabled application.

## 6 Rule Editing for Contents Presentation Design

We are developing an intelligent learning environment targeting heritage education, using ontology-based learner modeling to customize and refine the learning interaction [Kim *et al.*, 2004]. To support this, we have developed a learner ontology based on [Chen & Mizoguchi, 1999]’s approach. A learner is modeled with profile information containing personal data, comprehensive assessment of learner’s capabilities, dynamic assessment of learner’s

current mood and knowledge, low-level activity records for every learner action and system activity, and processed data obtained from the activity records. In addition, our learner ontology incorporates multiple sets of learner preferences, including Myers-Briggs Type Indicators, Felder & Silverman's Index of Learning Styles [Felder, 2002], and Chen & Mizoguchi's learning preferences. Learner models are inferred from the learners' records of interaction with the system, using data mining.

Felder & Silverman's Index of Learning Styles classifies a learner along 4 axes, with two extremes per axis: (S)ensory–I(n)tuitive, (V)isual–(A)uditory, A(c)tive–(R)eflective, and Se(q)uential–(G)lobal. We abbreviate each value to 1 letter, denoted by the parentheses. Each of the 16 combinations defines a set of learning preferences, which determines the best way in which learning contents should be presented to those students.

For a teacher, this becomes a task of *contents presentation design*. Specific learning content objects are annotated with properties defining the learning styles in which they are to be used. Using the rule editor, the teacher then defines the contents presentation knowledge as inference rules, which take a student's Felder & Silverman learning style as input, and selectively enables and disables learning content objects, and also adjusts their sizes, colors, positions, etc. Examples of the customized contents presentation for two combinations NARG and SVCQ are shown in Figure 4.



Figure 4. Contents Presentation Design using Felder & Silverman's Index of Learning Styles

A future extension of this work is to deduce these inference rules automatically from visual exemplars of the learning contents presentation, which further simplifies the teacher's task.

### 7 Conclusion

Inference rules are a significant form of knowledge representation for pedagogical knowledge within an ontology-based ITS, which accurately records a teacher's intent, while supporting automatic execution. We have identified a desideratum to make pedagogical rule editing accessible to non-technical users. To support this, we have identified common rule idioms at the natural language level, and developed a mapping technique from these idioms into complete Jess rules. This supports a simplified rule syntax in which the idioms appear as keywords, hiding the complexities of their implementation.

### References

[Chen & Mizoguchi, 1999] Chen, W., and Mizoguchi, R. Communication Content Ontology for Learner Model Agent in Multi-agent Architecture. *Proc. 7th Int'l. Conf. on Computers in Education*, pp. 95–102, Chiba, Japan.

[Felder, 2002] Felder, R. Learning and teaching styles in engineering education. *Engineering Education*, **78**(7), pp. 674–681.

[Friedmann-Hill, 2003] Friedmann-Hill, E. *Jess in Action*. Manning, Greenwich.

[Gruber, 1993] Gruber, T. R. Toward principles for the design of ontologies used for knowledge sharing. In N. Guarino and R. Poli, (Eds.), *Int'l. Workshop on Formal Ontology*, Padova, Italy, 1993. Revised version also published in *Int'l. Journal of Human-Computer Studies*, **43**(5-6), pp. 907–928, Nov./Dec. 1995.

[Horrocks et al., 2004] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosz, B., and Dean, M. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. *W3C Member Submission*, [www.w3.org/Submission/2004/SUBM-SWRL-20040521/](http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/), May 2004.

[Hubbard et al., 1996] Hubbard, C., Mengshoel, O. J., Moon, C., and Kim, Y. S. Multimedia Instructional Software for Visual Reasoning: Visual Reasoning Tutor (VRT). *Proc. Int'l. Conf. on Multimedia Computing and System*, pp. 261–268, Hiroshima, Japan, Jun. 1996.

[Kim et al., 2004] Kim, Y. S., Kim, S. A., Wang, E., Park, B. J., Jeon, K. J., and Cho, Y. J. An Intelligent Learning Environment for Heritage Alive. To appear in *Proc. Conf. of KSPE* (special session on intelligent HCI), Busan, Korea, Oct. 2004 [in Korean].

[Protégé, 2004] [www-protege.stanford.edu](http://www-protege.stanford.edu).