Automated Transformation of SWRL Rules into Multiple-Choice Questions

Konstantinos Zoumpatianos [1] and **Andreas Papasalouros** [2] and **Konstantinos Kotis** [1]
[1] Department of Information and Communication Systems Engineering
[2] Department of Mathematics
University of the Aegean
Karlovassi, 83200 Samos, Greece

Abstract

Various strategies and techniques have been proposed for the generation of questions/answers tests in Intelligent Tutoring Systems by using OWL (Web Ontology Language) ontologies. Currently there have been no known methods to utilize SWRL rules for this task. This paper presents a system and a set of strategies that can be used in order to automatically generate multiple choice questions from SWRL rules. The aim of the proposed framework is to support further research in the area and to be a testbed for the development of more advanced assessment techniques.

Introduction

The use of structured knowledge in the form of domain-specific ontologies, rules and instances in order to automatically generate multiple-choice questions (MCQ) would enable Intelligent Tutoring Systems (ITS) to utilize pre-existing knowledge bases for the assessment of learners' knowledge and skills.

There have been various strategies and techniques proposed in literature that argued for the use of OWL (Web Ontology Language) ontologies for the generation of questions and answers (Papasalouros, Kanaris, and Kotis 2008; Kotis, Papasalouros, and Nikitakos 2009; Holohan et al. 2006; Papasalouros, Kotis, and Nikitakos 2010). Furthermore, the potential of extending these strategies for SWRL (Semantic Web Rule Language) rule specifications has been indicated by (Papasalouros, Kotis, and Nikitakos 2010). Nonetheless, there have been no known implementations proposed to this time capable to support the generation of questions and answers from SWRL rules. The SWRL is a proposal for a Semantic Web Rule Language based on the combination of OWL with RuleML, specifically OWL DL and OWL Lite with the Unary/Binary Datalog RuleML sub-language.

This paper presents a system that converts SWRL rules into natural language multiple-choice questions as well as techniques for generating the appropriate distracting answers. It has been designed and implemented as an extensible framework supporting new question and answer formation strategies. Currently, this system is able to form questions for data type properties (e.g. $price(x,5)$), individual type properties (e.g. $hasFriend(x,y)$) and class properties (e.g. $Engineer(x)$), as well as generate distracting answers for the two last.

Related work

Mitkov, Ha, and Karamanis (2006) describe a method for generating multiple choice questions from textbook material. In order to generate stems (questions) and the correct answer, text is parsed based on specific patterns. Then, Wordnet ontology is used in order to generate plausible distractors based on subsumption relationships between terms. Holohan et al. (2005) propose the use of subsumption relationships between classes, as well as class-instance relationships defined in domain ontologies in order to generate multiple choice questions, as part of personalized learning objects. The system they developed, OntoWare, was extended (Holohan et al. 2006) so as to generate tests for problem solving skills in the particular domain. Cubric and Tosic (2010) propose a method for generation of MCQ questions based on a combination of annotations on ontology elements and question templates. They propose strategies that utilize these annotations, besides the semantics of the ontological descriptions. By the use of templates they can generate questions that can assess learning objectives according to the Bloom Taxonomy. Papasalouros, Kanaris, and Kotis (2008) propose a set of strategies for generating questions exploiting the semantics of OWL language. These strategies only partially deal with the syntactic correctness of the produced questions, by using simple Natural Language Generation techniques. Cubric and Tosic (2010) have implemented the aforementioned strategies as a Protegé plug-in, also leveraging the problems of syntactical correctness. Soldatova and Mizoguchi (2007) describe an ontology-based method for test generation. Three ontologies are used for the following purposes: a test ontology specifying the items of each test, a student model and, finally, a set of rules for test creation. Focus is given to the determination of test difficulty for each particular student. Tests are automatically generated by using a domain model containing facts, events and terms(Soldatova and Mizoguchi 2003). While this method requires the adoption of a specific predefined ontology, our approach utilizes only the semantics of OWL and SWRL, regardless of a particular domain.

None of the above-mentioned methods utilizes SWRL

rules in question/answer generation. We conjecture that the expressiveness of these rules improves the quality of generated questions, compared with questions generated by OWL rules, since they can effectively test knowledge of relationships between concepts and individuals, thus assessing more deep levels of student understanding.

Three workshops have been dedicated to the problem of question generation solely from given texts or text corpora [1] where a number of solution approaches have been presented based on Natural Language Processing techniques. A widely used method is based on the application of rules for transforming selected clauses or sentences from declarative to interrogative form. These rules define certain syntactical tree patterns and the application of appropriate generation templates which apply morphological and syntactical transformations to the inupt clauses. In order to improve generated questions, certain forms of pre-processing are applied such as anaphora resolution, complex sentence splitting, key concept identification and name entity recognition. Besides syntax-based techniques, such as the above, approaches that utilized sentence semantics have been applied, such as Minimal Recursion Semantics (Yao and Zhang 2010) and Semantic Role Labeling (Pal et al. 2010). While our approach is not related to text processing, it can utilize NLP techniques such as the above for ontology and rules learning and population from text. These ontologies and rules can then be used, as an intermediate language, for question generation.

The Semantic Web Rule Language

SWRL rules are in the form $antecedent \implies consequent$, which is interpreted so that if the antecedent holds, then the consequent must also hold. Both the antecedent and the consequent parts are conjunctions of atoms. Atoms are unary or binary predicates, that is, class or property predicates, respectively. These predicates contain variables, individuals or data literals. Variables can be placed in both the antecedent and the consequent parts and they represent both individuals and data. A restriction called "safety", imposes that only variables present in the antecedent part can be placed in the consequent part. An abstract notation will be used for the rest of the paper to refer to individual variables as $I\text{-}variable(variable\_name)$ and data variables as $d\text{-}variable(variable\_name)$. Being an extension to the OWL-DL ontology description language, SWRL supports Classes, Sub-Classes, Class Equivalences, Disjoint Classes and Data Types as well as Data-valued Properties and Individual-valued Properties. Thus, all of those axioms and facts can be used to describe each variable present in a rule.

System description

The system consists of three layers. The first one is responsible for parsing rule files and converting them into an internal representation. Specifically a rule, which is originally written in the form of conjunctions of atoms, becomes a set of objects. One object for each variable, which in their turn contain all the atoms specific to this variable. The second layer is responsible for generating a textual representation for each variable. Those representations are then combined to form questions. The last layer is the one responsible for the generation of the appropriate distracting answers. It uses the semantics of OWL-DL to discover objects that are related to the correct one.

Data representation

For the sake of generality and convenience, an internal representation of rules has been developed. This not only allows the system to remain relatively input agnostic and thus extensible, but most importantly, it represents the main philosophy behind our approach. A set of atoms that form a rule are converted into a set of variables. Specifically, instead of describing a conjunction of property and class predicates, the variables themselves and the relations between them are described. The parser generates a set of rule objects, one for each rule in the file. These objects contain sets of variable objects, one for each variable present in the rule. As an example, for each rule in an SWRL document a new set $R = \{V_i, \ldots, V_n\}$ will be initialized, containing n variable objects $V_i = \{A_i, C_i\}$, where $A_i = \{P_{i_1}, \ldots, P_{i_k}\}$ and $C_i = \{Q_{i_1}, \ldots, Q_{i_l}\}$ are the sets of predicates, either class or properties, related to this specific variable. A_i contains the predicates that belong to the antecedent part of the rule and C_i contains the predicates found in the consequent part of the rule. Basically, it is the set of all the properties in a rule, where each variable V_i is the first argument of a predicate (e.g. $has(I\text{-}variable(V_i), something)$ or $Class(I\text{-}variable(V_i))$ but not $has(something, I\text{-}variable(V_i))$).

The properties $P_{i_k} \in \{DataTypePropery, ObjectProperty, Class\}$ can be either Data-valued Properties (properties for which the value is a data literal), Individual-valued Properties (properties for which the value is an instance) or Data-range Properties (properties for which the value is a class).

Each different property type contains a PropertyPredicate value (the name of the class, or the name of the relation) and one or more arguments.

$ObjectProperty = \{Arg_1, PropertyPredicate, Arg_2\}$
$DataTypeProperty = \{Arg_1, PropertyPredicate, Arg_2\}$
$Class\{Arg_1, PropertyPredicate$

Example For example, given the following rule:

$$Person(?x) \wedge Holds(?x, ?y) \wedge Diploma(?y) \wedge isIssuedBy(?y, ?z) \wedge EngineeringSchool(?z) \Rightarrow Engineer(?x).$$

A rule object $R_1 = \{V_x, V_y, V_z\}$ will be instantiated with three variable elements, one for each variable.

A graphic representation of the data structures generated can be seen in figure 1. Each variable object contains the set of properties that concern this specific variable.

[1] http://www.questiongeneration.org/

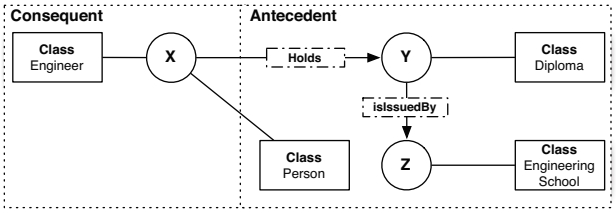

Figure 1: Data structure

$$
\begin{aligned}
V_x &= \{A_x, C_i\} \\
A_x &= \{Class(x, Person), IndividualProperty(x, Holds, y)\} \\
C_x &= \{Class(Engineer, x)\} \\
V_y &= \{A_y, C_y\} \\
A_y &= \{Class(y, Diploma), ObjectProperty(y, isIssuedBy, z)\} \\
V_z &= \{A_z, C_z\} \\
A_z &= \{Class(z, EngineeringSchool)\}
\end{aligned}
$$

At the same time there are formed relationships between the objects which are connected via an Individual-valued Property (figure 1). In this specific example, the object that represents I-variable(x) is connected to the object that represents I-variable(y) with the property *Holds* and this is connected to the object that represents I-variable(z) through the property *isIssuedBy*.

It comes natural that by implementing an algorithm able to express each variable object into natural language and then the relations between them we are able to describe the concept on the consequent part. In this example we would have the following sentence:

$$(a\ \textbf{Person}\ that\ \underline{holds}\ (a\ \textbf{Diploma}\ that\ \underline{is\ issued\ by}\ (an\ \textbf{Engineering School})^{I-variable(z)})^{I-variable(y)})^{I-variable(x)}$$

Variable verbalization strategies

The basic strategies implemented use a simple pattern and a set of sub-patterns to achieve this task. The basic pattern is based in a simple procedure which expresses the classes where each variable belongs to on the first part of the sentence, followed by the word "that" and then all the properties that connect this variable to other objects and values are expressed sequentially and concatenated appropriately (Algorithm 1). A possible extension of the aforementioned method in this stage, would be the use of WordNet to correctly decide the pronoun based on the class name(s) and not the use of the word "that" by default.

Each different property type can be expressed using a variety of strategies. In the default strategy the following simple methods are used. For Data Type Properties the property's predicate is verbalized with a set of standard string processing algorithms and added to the value itself. For Individual-valued Properties (Algorithm 1, **ExpressIndivProp** procedure) the predicate is verbalized and an **Express** procedure is recursively called in order to verbalize the related variable or call a procedure that verbalizes an instance. While with ReferenceTo the way the algorithm should refer to an element is decided. For example, an element which has already been expressed previously in the sentence must be referred to as "the element" and an element that has not been seen before must be called as "an element". One which has the same class as other elements, that have been already expressed before, must be called as "a second", "a third" and so on. Additionally, it should be referred as "the second", "the third", if it is already expressed earlier in the sentence. To achieve this, a stack has been implemented where all expressed elements are pushed and then used to decide the correct way of reference to an element.

Algorithm 1 Expressing variables and properties

```
procedure EXPRESS(Variable)
    VarClasses ← Variable.getClasses()
    VarProperties ← Variable.getProperties()
    expression ← ExpressClasses(VarClasses) + "that" + ExpressProperties(VarProperties)
    return expression
end procedure

procedure EXPRESSPROPERTIES(VarProps)
    e ← ""
    i ← 0
    while i < size_of(VarProps) do
        if is_answer(VarProps[i]) then
            continue                    ▷ this property is the answer!
        end if
        if is_data_type(VarProps[i]) then
            e ← concat(e, ExpressDataProp(VarProps[i]))
        else
            e ← concat(e, ExpressIndivProp(VarProps[i]))
        end if
    end while
    return e
end procedure

procedure EXPRESSINDIVPROP(prop)
    pred ← prop.getPredicate()
    arg ← prop.getSecondArgument()
    if isInstance(arg) then
        e ← Verbalize(pred) + ReferenceTo(arg) + ExpressInstance(arg)
    else
        e ← Verbalize(pred) + ReferenceTo(arg) + Express(arg)
    end if
    return e
end procedure
```

Question expression strategies

The system is implemented in such a way that it fully supports the easy addition of new strategies. As goes with the variable verbalization strategies, while there can be many different strategies available to express questions, only one for each type has been implemented.

In general, there are three basic question categories one can generate from SWRL rules.

Data-range questions Data-range property questions can mainly be of two types. Being easier to describe with an example, we are going to use the same one as in the previous sections. In the sentences bellow the underlined part refers to the antecedent while the not underlined one refer to the consequent part. Additionally, class names are marked with bold letters.

- An **engineer** is a **person** that holds a **diploma** issued by an **engineering school**

1. What is a(n) person that holds a diploma issued by an engineering school.
2. An engineer is a person that holds a _____ issued by an engineering school.

It is obvious that in the first type a question is asked, while in the second one some facts are simply removed and the test taker is asked to "fill in" the blanks. The prototype strategies focus only on the first method since the second one was considered to be more trivial to implement. Furthermore, there is another major difference between those two questions. In both of them Class questions are asked, but in the first one the name of the missing class is part of the consequent, while in the second one it belongs to the antecedent part. In the current prototype, only classes and properties that belong to the consequent part are asked from the quiz taker and only parts that belong to the antecedent part are expressed. This limitation can be bypassed with a small modification on the original rule. If some elements from the antecedent part are moved to the consequent part and the reverse the following rule can be produced.

$$Person(?x) \wedge Engineer(?x) \wedge Holds(?x,?y) \wedge isIssuedBy(?y,?z) \wedge EngineeringSchool(?z) \Rightarrow Diploma(?y).$$

Which in its turn can be simplified to exclude the class Person, since it is covered by its sub-class Engineer. Now the sentence becomes the following.

1. What is a(n) object issued by an engineering school, when an engineer holds this object?

Notice that the part of the sentence that follows the comma is introduced by a post-processing step that is going to be described later. The basic pattern that we use to express this kind of questions is simply the following: $Question =$ "$What\ is\ a(n)$" $+ express(variable)$.

Example

$$Person(x) \wedge Diploma(y) \wedge Holds(x,y) \wedge hasECTS(y,500) \Rightarrow Engineer(x)$$

The sentence above, would become: "What is a(n) (expression_of_x: person which holds a (expression_of_y: diploma which has ects 500))"

Individual-valued properties questions The algorithm for expressing questions based on individual properties is the most complex one and is described using the following pattern.

$$Question = "What\ is\ the\ relation\ between\ a(n)" + Express(prop.getFirstArgument()) + "and" + ReferenceTo(prop.getSecondArgument() + Express(prop.getSecondArgument()))$$

Example

$$Engineer(x) \wedge Holds(x,y) \wedge Transcript(y) \wedge hasDurationOfYears(y,5) \Longrightarrow hasEngineeringDiploma(x,y)$$

The above rule would be expressed as: "What is the relation between an (expression_of_x: engineer that holds a (expression_of_y: transcript which has duration of years 5) and this (expression_of_y: transcript that has a duration of years 5)) "

An apparent problem here, is the re-expression of element y. This is stopped by using a similar to the ReferenceTo function, which for already expressed elements, would restrict the expression of this element just to the class name.

Data-valued properties questions Expressing Data-valued properties can be a daunting task, so a simple and hopefully meaningful strategy has been used. It is described in the following pattern.

$Question = "What\ is\ the" + Verbalize(prop.getPredicate()) + "value\ for" + Express(prop.getFirstArgument())$

Example

$$Engineer(x) \wedge Diploma(y) \wedge Holds(x,y) \Rightarrow ECTS(y,500)$$

The rule above would become "What is the **ects** value for (expression_of_y: a **diploma**) "

An apparent problem here, is that the *engineer* is not described at all. This is solved by this strategy using a post-processing step mentioned below, its job is to expresses all unexpressed elements.

Post processing step It is apparent that some times elements are not described at all. This is because they are not linked directly to the target element but rather in an inverse fashion. This is demonstrated in Figure 2, where while in both figures we ask the quiz taker to describe an element of the consequent part, in Figure 2b the variable object Y, does not link to X but is linked by it. For those elements that are left out, they are expressed at the end of the sentence using the following pattern.

$PostProcess(question) = Question + "when" + ExpressAllUnexpressedElements()$

By using this technique the problem apparent in a previous example is solved, as the question is now converted to: "What is the ects value for (a diploma) when (an engineer holds this diploma)?"

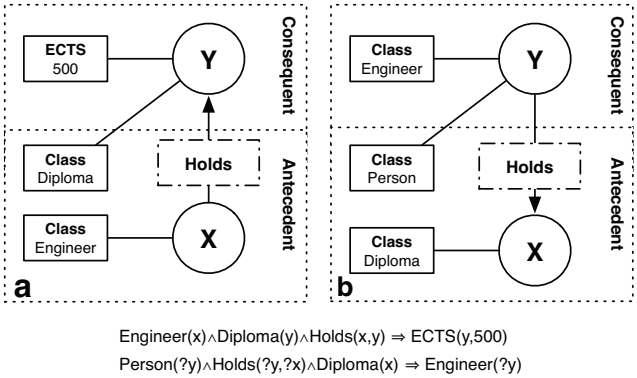

Figure 2: Backward linking

Answer generation algorithms

Answers can be generated using a variety of strategies, most of which are based on the OWL semantics. Some example strategies that have been implemented in the prototype are presented here.

- For Data-range questions, the sub-classes of the correct class's super-class are displayed
- For Individual-valued Property questions two strategies have been developed.
  - All the sub-properties of the correct property's super-property are presented.
  - All the properties with the same domain and range are presented.
- For Data-valued properties no strategies have been developed due to their increased complexity, but some future strategies could be the following.
  - For each different data type property present distracting values.
    - Based on a mathematical series which contains this value (numerical data types).
    - Using WordNet for textual data.
    - Randomly display values that belong to this DataType and exist in the ontology file.

Implementation

A framework that supports the ideas presented in this paper has been developed in Java 1.6 using the Jena framework as a portable library. The addition of new question and answer generation strategies is fully supported by the implementation of the appropriate interfaces. Additionally, there have been efforts to make the library remain as agnostic as possible in regards to the format of the input files. Thus, input file parsers can also be enhanced by implementing the appropriate interfaces and incorporating them into the system. Finally, a graphical user interface was added that presents the multiple choice questions to the test taker as an interactive quiz and enables the user to choose the correct answer. At the end of the test, the user is informed about the total score of correct answers he/she achieved.

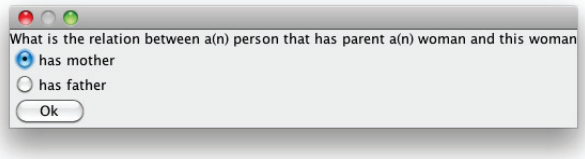

Figure 3: Graphical user interface

Evaluation

The system has been tested with three SWRL documents. The metrics of these documents are described in Table 1.

Table 1: Ontologies used for evaluation

Domain	Classes	Properties	Individuals	Rules
Musical Instruments	14	8	0	1
Family	65	28	29	14
Oil Pollution	89	31	33	5

In Table 2 the total number of generated questions are displayed. Because of the way the presented algorithm works, the amount of questions generated highly depends on the amount of rules available in the ontology file, as well as the number of atoms available in the consequent part (which is always one atom per rule in the current experiments).

In Table 3 the average number of answers for each question is displayed for two different Individual-valued Property question/answer generation strategies. The first one displays the sub-properties of the correct property's super-property as distracting answers. While the second strategy "relaxes" this heuristic by producing a greater number of possible answers. This is done by displaying all properties with the same domain and range. Obviously, as can also be seen in Table 3, there was a significant increase in the number of answers using this "more relaxed" strategy, especially in domains where many Individual-valued Property questions had been generated. For example, a greater average number of possible answers can be seen for the Family domain, since the vast majority of properties involved in it are Individual-valued properties (Family relationships) and all of the 14 questions generated were Individual-valued Property ones. This increase cannot be seen in the Musical Instruments though, and this is because it only provides Data-range questions.

Table 2: Questions generated per ontology

Domain	Questions (Class, Individual, Data)	Variables/Rule
Musical Instruments	**1 (1, 0, 0)**	1
Family	**12 (0, 12, 0)**	1.4
Oil Pollution	**4 (2, 1, 1)**	1.85

Samples of questions generated are given below.

- **Super property's sub-properties strategy**

Table 3: Average answers per question for each method

Domain	Super property's sub-properties	Same domain and range
Musical Instruments	3	3
Family	1.5	4.8
Oil Pollution	1	1

  - **Musical Instruments:** *What is a(n) stringed musical instrument that has number of strings 6?*
    a. Guitar, b. Ukulele, c. Banjo
  - **Family:** *What is the relation between a(n) person that has parent a(n) woman and this woman?*
    a. has mother, b. has father

- **Properties with same domain and range strategy**
  - **Family ontology:** *What is the relation between a(n) person that has parent a(n) 2nd person that has consort a(n) 3rd person and this 3rd person?*
    a. has parent, b. has sibling, c. has child, d. has consort

In the above examples we see that syntactic correctness is not always achieved. For example, property *has_parent* should be manually replaced by *parent* in order to provide a syntactically correct answer.

The above question items samples also demonstrate the improvement in question semantics in comparison with questions generated by OWL. While the question on musical instruments could be generated without the use of SWRL, by appropriately inspecting OWL datatype properties, the question about the family ontology utilizes a rule associating binary predicates (relationships), which would not be possible by using OWL. Unlike OWL, its successor, OWL 2 (W3C 2009), allows property relationships such as the above in the form of property chains, a feature which is not investigated in this research.

Future work

Currently the proposed framework contains prototype strategies and hosts a variety of string processing and Knowledge Engineering algorithms. The enrichment of this algorithm collection will make the development and application of more complicated algorithms easier, thus, easing the exploration of the problem of automatic MCQ generation both from a linguistic as well as from a Knowledge Engineering perspective.

An evaluation with real users would reveal the practical usefulness of the method. Furthermore, a detailed comparison of our approach to other MCQ generation techniques, especially non rule-based methods, is a theme of great interest that is part of our future plans to conduct.

Other areas of future work include the extension of this framework so that it can provide a set of metrics for the questions and answers generated. Those metrics could include measures such as grammatical correctness, question complexity (number of entities involved), difficulty and others. An introduction that would certainly ease the process of comparing different strategies in terms of question difficulty, complexity and readability.

References

Boyer, K. E., and Piwek, P., eds. 2010. *Proceedings of QG2010: The Third Workshop on Question Generation*.

Cubric, M., and Tosic, M. 2010. Towards automatic generation of e-assessment using semantic web technologies. In *Proceedings of the 2010 International Computer Assisted Assessment Conference, Jul 2010*. University of Southampton.

Holohan, E.; Melia, M.; McMullen, D.; and Pahl, C. 2005. Adaptive courseware generation based on semantic web technology. In *International Workshop on Applications of Semantic Web Technologies for E-Learning (SW-EL 2005)*, 29–36.

Holohan, E.; Melia, M.; McMullen, D.; and Pahl, C. 2006. The generation of e-learning exercise problems from subject ontologies. In *ICALT*, 967–969. IEEE Computer Society.

Kotis, K.; Papasalouros, A.; and Nikitakos, N. 2009. Supporting decision making in maritime environmental protection with a knowledge-based education and awareness approach. In Kotis, K., ed., *Artificial Intelligence Applications in Environmental Protection Workshop, collocated with 5th IFIP Conference on Artificial Intelligence Applications and Innovations 23-25 April, Thessaloniki, Greece*.

Mitkov, R.; Ha, L.; and Karamanis, N. 2006. A computer-aided environment for generating multiple-choice test items. *Natural Language Engineering* 12(02):177.

Pal, S.; Mondal, T.; Pakray, P.; Das, D.; and Bandyopadhyay, S. 2010. QGSTEC system description – JUQGG: A rule based approach. In Boyer and Piwek (2010), 76–79.

Papasalouros, A.; Kanaris, K.; and Kotis, K. 2008. Automatic generation of multiple choice questions from domain ontologies. In Nunes, M. B., and McPherson, M., eds., *e-Learning*, 427–434. IADIS.

Papasalouros, A.; Kotis, K.; and Nikitakos, N. 2010. Towards an intelligent tutoring system for environmental decision makers. *Environmental Engineering and Management Journal* 9(2):197–204.

Soldatova, L., and Mizoguchi, R. 2003. Ontology of tests. In *Proceedings of Computers and Advanced Technology in Education*, 175–180.

Soldatova, L., and Mizoguchi, R. 2007. Testing and understanding by use of an ontology methodology. In *Proceedings of Joint Workshop of Cognition and Learning*, 202–205.

W3C. 2009. OWL 2 Web Ontology Language: Document Overview. Available at `http://www.w3.org/TR/owl2-overview/`.

Yao, X., and Zhang, Y. 2010. Question generation with minimal recursion semantics. In Boyer and Piwek (2010), 68–75.