

XET Protégé Plug-in Environment

Photchanan Ratanajaipan¹, Vilas Wuwongse², Ekawit Nantajeewarawat³, Chutiporn Anutariya¹

¹ Computer Science Program, Shinawatra University, Thailand

² Computer Science and Information Management, Asian Institute of Technology, Thailand

³ Sirindhorn International Institute of Technology, Thammasat University, Thailand
photchanan@shinawatra.ac.th, vw@cs.ait.ac.th, ekawit@siit.tu.ac.th, chutiporn@shinawatra.ac.th

Abstract

XML Equivalent Transformation (XET) is a general-purpose XML-based rule language developed based on the XML Declarative Description (XDD) theory and the Equivalent Transformation (ET) computation theory. Of central importance to the XET approach, it uses XML expressions as its underlying data structure; consequently, one can seamlessly specify information to be extracted as well as implicit information to be derived from XML data. An XET Protégé Plug-in environment supporting editing and executing XET rules has been developed. Treating XML-based Protégé ontology as input for an XET program (a set of XET rules), various kinds of query processing and inference can be performed through an XET execution engine.

1. Introduction

XML Equivalent Transformation (XET) [1] is an XML-based rule language for the Web that deals directly with XML data—it treats XML elements and XML expressions, i.e., ordinary XML elements extended with variables, are first-class programming entities. With XET, arbitrary XML documents, representing application data, information or knowledge on the Web possibly encoded in certain XML applications, become immediately a program's input data. Manipulation and computation of such an input document (data) is performed by semantically-equivalently transforming the document successively until a desirable one is obtained. The input document could be, for example, an XML database query, and thus the output or the desirable document is a set of XML elements yielding the answer to the query. Application of XET in several domains has been demonstrated, e.g., in e-business [1], e-learning [2], and system modeling [3]. XET uses the XML Declarative Description (XDD) theory [4, 5] as a theoretical basis for XML-based knowledge representation, and computation using XET rules is founded on the Equivalent Transformation (ET) paradigm [6, 7]. After recalling some basic concepts of the XDD theory and the ET theory, this paper introduces XET using some simple examples, and gives an overview of an XET Protégé Plug-in that provides support for editing and executing XET rules in Protégé.

2. XDD and ET

The *XML Declarative Description (XDD)* theory [4, 5] is an XML-based declarative modeling language, which extends ordinary, well-formed XML elements into XML expressions by incorporation of variables for an enhancement of expressive power and representation of implicit information. An *XDD description* is a set of *XDD clauses*, each of which is a formula of the form

$$H \leftarrow B_1, \dots, B_m, \beta_1, \dots, \beta_n,$$

where $m, n \geq 0$, H and the B_i are XML-expressions, and the β_j are constraints. H is called the *head*, and $\{B_1, \dots, B_m, \beta_1, \dots, \beta_n\}$ the *body* of the clause. Variables of several kinds, with different syntactical usage and different instantiation characteristics, are employed in XML expressions. They include name variables (*N*-variables), string-variables (*S*-variables), attribute-value-pair-variables (*P*-variables), and XML expression-variables (*E*-variables). Intuitively, an *N*-variable will only be instantiated into either a tag name or an attribute name, an *S*-variable into a string, a *P*-variable into zero or more attribute-value pair(s), and an *E*-variable into zero or more XML expression(s). A variable is used for a dual purpose: it denotes a specialization wildcard (i.e., a variable can be specialized into an XML expression or a part thereof) and, at the same time, specifies an equality constraint (i.e., any occurrence of a variable within the same scope must be instantiated in the same way). A constraint in the body is an expression that specifies certain restriction on XML elements or their components. The reader is referred to [5] for the formal semantics of XDD descriptions.

The *Equivalent Transformation (ET)* paradigm [6, 7] is a computational model that solves a given problem, described in an appropriated language, by simplifying it through repetitive application of meaning-preserving transformation rules, called *equivalent transformation rules (ET rules)*. Given a description D_1 representing a

$$\begin{aligned}
\text{Head } \{ \text{Cond} \} &\Rightarrow \{ \text{Exec}_1 \}, \text{Body}_1; \\
&\Rightarrow \{ \text{Exec}_2 \}, \text{Body}_2; \\
&\quad \dots \\
&\Rightarrow \{ \text{Exec}_n \}, \text{Body}_n.
\end{aligned}$$

Fig. 1. Abstract syntax of XET rule

| Rule #1 | Rule #2 |
|--|--|
| <pre> <xet:Rule xet:name="hasBrother" xet:priority="1"> <xet:Head> <person rdf:ID="Svar_x"> <hasBrother rdf:resource="Svar_y"/> </person> </xet:Head> <xet:Body> <person rdf:ID="Svar_x"> <hasParent rdf:resource="Svar_z"/> </person> <person rdf:ID="Svar_y"> <hasParent rdf:resource="Svar_z"/> </person> <man rdf:ID="Svar_y"/> </xet:Body> </xet:Rule> </pre> | <pre> <xet:Rule xet:name="hasUncle" xet:priority="1"> <xet:Head> <person rdf:ID="Svar_x"> <hasUncle rdf:resource="Svar_y"/> </person> </xet:Head> <xet:Body> <person rdf:ID="Svar_x"> <hasParent rdf:resource="Svar_z"/> </person> <person rdf:ID="Svar_z"> <hasBrother rdf:resource="Svar_y"/> </person> </xet:Body> </xet:Rule> </pre> |

Fig. 2. Examples of XET rules

problem in an application domain, ET rules are applied in order to construct a transformation sequence $D_1 \Rightarrow D_2 \Rightarrow \dots \Rightarrow D_n$ such that the description D_n is in a form from which the answer set of the problem can be derived effortlessly and $\mathcal{M}(D_1) = \mathcal{M}(D_2) = \dots = \mathcal{M}(D_n)$, where $\mathcal{M}(D_i)$ denotes the meaning of D_i , i.e., a set of concrete statements each of which is a surrogate of a real, tangible or intangible object or relationship in the domain of interest that D_i represents. An example of an ET rule is the unfolding transformation rule, which is a fundamental computation rule in logic programming.

3. XET and XET Rule Editor

XML Equivalent Transformation (XET) [1] is an XML-based, procedural, rule language, designed for processing XML declarative descriptions using the ET computation model. An *XET program* is an XML document comprising a set of *XET rules* and XML elements/documents, which are regarded as the program's data or facts. It receives, as an input, a description (any XML data), then process—semantically-equivalent transforms—the input until a desirable description is obtained without any need of scheme or data conversion. Computation or execution of an XET program can be done through the XET execution engine by successively applying XET rules to a given XDD description until a desirable XDD description yielding its answer is obtained.

An abstract syntax of XET rules is shown in Fig. 1, where *Head* is an XML expression, *Cond* is a sequence of applicability conditions, *Exec_i* is a sequence of predefined operations, and *Body_i* is a sequence of XML expressions. A rule can be read as follows: if the XML expression in *Head* matches with a target XML expression and the applicability condition of the rule is satisfied, then the n bodies fire simultaneously, i.e., the predefined operations in *Exec_i* are carried out and the target XML expression is replaced with the XML expressions *Body_i*. Fig. 2 gives examples of XET rules. Rule #1 replaces a target person-element describing a hasBrother-instance with its corresponding pair of person-elements with hasParent-instances and one man-element, thereby expanding a hasBrother-instance in terms of two hasParent-instances and a man-instance. Likewise, Rule #2 expands a hasUncle-instance in terms of a hasParent-instance and a hasBrother-instance. One can use these two rules, for example, to infer XML elements representing hasUncle-instances. In addition, since variables can appear in any part of an XML expression, one can also use the rules to discover relations that hold between two given individuals. For simplicity, applicability conditions and predefined operations are not shown in Fig. 2.

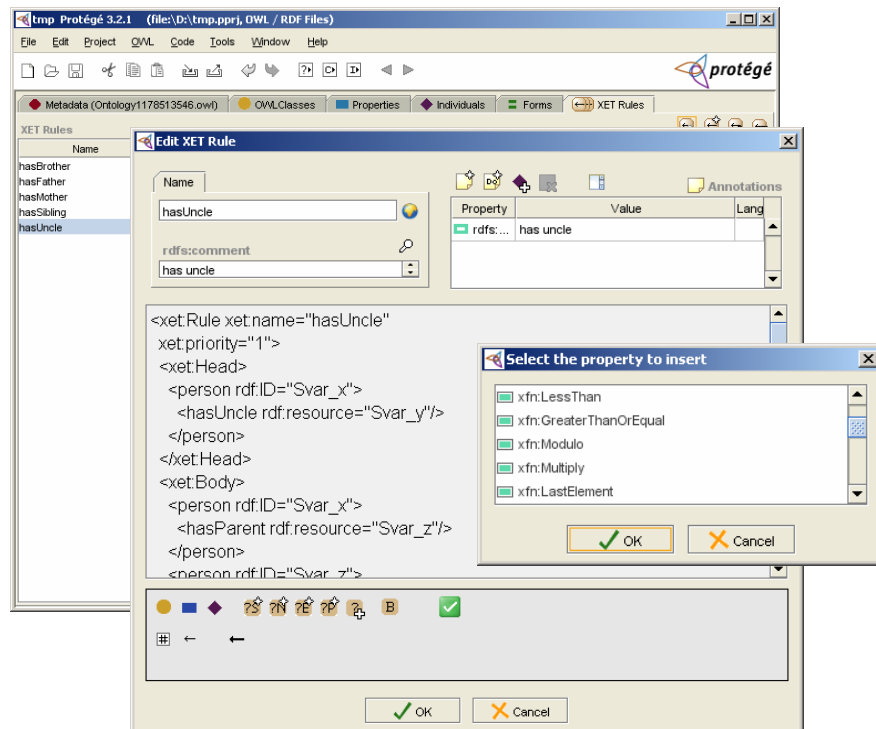


Fig. 3. XET Rule Editing dialog with features for inserting Protégé entities

Protégé *XET Rule Editor* supporting XET program development in Protégé-OWL has been implemented. Like the well-known open source rule editor for SWRL, *SWRL Rule Editor* [8], which is an extension to Protégé-OWL, XET Rule Editor provides basic rule manipulating functions, e.g., creating, editing, reading and writing rules; however, it is specifically tailored to XET. The editor is integrated with Protégé and can be accessed through the XET Rules tab. It supports rules authoring by providing buttons for selecting protégé entities, e.g., classes, properties, individuals, along with XET built-in functions, and for inserting various possible kind of variables for specifying information to be extracted from XML-based ontology components. Fig. 3 illustrates XET rule editing dialogs. The editor is equipped with an XET execution engine, called NxET, developed at Asian Institute of Technology.

References

- [1] C. Anutrariya, V. Wuwongse, and V. Wattanapailin, "An Equivalent-Transformation-Based XML Rule Language," Proceedings of the International Workshop on Rule Markup Languages for Business Rules in the Semantic Web, Sardinia, Italy, pp. 1–15, 2002.
- [2] P. Ratanajaipan, E. Nantajeewarawat, and V. Wuwongse, "Representing and Reasoning with Application Profiles Based on OWL and OWL/XDD," Proceedings of the First Asian Semantic Web Conference, Beijing, China, Lecture Notes in Computer Science, vol.4185, pp. 256–262, 2006.
- [3] E. Nantajeewarawat and V. Wuwongse, "Knowledge-Based Inconsistency Detection in UML Models," Handbook of Software Engineering & Knowledge Engineering, vol.3, pp. 177–201, World Scientific, 2005.
- [4] V. Wuwongse, C. Anutariya, K. Akama, and E. Nantajeewarawat, "XML Declarative Description: A Language for the Semantic Web," IEEE Intelligent Systems, vol.1, no.3, pp.54–65, 2001.
- [5] V. Wuwongse, K. Akama, C. Anutariya, and E. Nantajeewarawat, "A Data Model for XML Databases," Journal of Intelligent Information Systems, vol.20, no.1, pp.63–80, 2003.
- [6] K. Akama, Y. Shigeta, and E. Miyamoto, "Solving Logical Problems by Equivalent Transformation: A Theoretical Foundation," Journal of the Japanese Society for Artificial Intelligence, vol.13, pp. 928–935, 1998.
- [7] K. Akama and E. Nantajeewarawat, "Formalization of the Equivalent Transformation Computation Model," Journal of Advanced Computational Intelligence and Intelligent Informatics, vol.10, no.3, pp.245–259, 2006.
- [8] M.O.Conner, H. Knublauch, S. Tu, B. Groszof, M. Dean W. Grosso, and M. Musen, "Supporting Rule System Interoperability on the Semantic Web with SWRL," Proceedings of the Fourth International Semantic Web Conference, Galway, Ireland, Lecture Notes in Computer Science, vol. 3729, pp. 974–986, 2005.