Elements of a First Visual Rule Language for the Semantic Web

Grigoris Antoniou, Institute of Computer Science, FORTH, P.O. Box 1385, 71110 Heraklion, Greece, antoniou@ics.forth.gr
Mikael Berndtsson, School of Humanities and Informatics, University of Skövde, Box 408, 541 28 Skövde, Sweden, spiff@ida.his.se
Silvie Spreeuwenberg, LibRT, Amsterdam, The Netherlands, silvie@librt.com
Kuldar Taveter, VTT Information Technology, P.O.Box 1201, FIN-02044 VTT, Finland, kuldar.taveter@vtt.fi
Gerd Wagner, Faculty of Technology Management, Eindhoven University of Technology, The Netherlands, G.Wagner@tm.tue.nl

Introduction

Models, and modeling languages, are a particularly important means of investigating analysis and design problems. For analyzing and designing the rules of a rule-based Web application we need a rule modeling language based on a general language for information modeling. In addition, business/domain models are used in the communication between business/domain analysts and business/domain experts for analyzing and documenting system requirements. Consequently, the modeling language used must not be 'technical', but should allow (semi-)visual and/or natural-language-like rule expressions, which can be understood by business/domain experts without extensive technical training. The design of a Web rule language, however, is a meta-modeling problem.

Rules can be visualized in many ways. Different types of rules require different graphical expressions. In addition, there are several purposes of rule visualizations. For instance:

- expressing the syntactical structure of rules
- expressing dependencies between rules and their constituents
- including rules in model diagrams
- supporting testing and debugging of rules

This paper presents work in progress on developing a visual rule language for the Semantic Web. In particular, we survey three areas that are important for developing visual rule languages for the Semantic Web:

–Visualizing vocabularies: Rules are built on vocabularies, and vocabularies are built on names and terms. Thus, rules are an important means for defining terms in a glossary or vocabulary and for formalizing (parts of) business policies.
–Visualizing derivation rules: Derivation rules are primarily used for reasoning.
–Visualizing reaction rules: Reaction rules are primarily used for reacting to events occurring at remote web resources. They can also be used for monitoring events on the local web resource.

The work reported in this paper is performed as part of the research network REWERSE (REasoning on the WEb with Rules and Semantics). REWERSE is a four-year research network founded by the EU-Commission and Switzerland, and it involves about 100 computer science researchers and professionals. The interested reader can find further details about this survey in [AB+04].

Visualizing Vocabularies

There are various formalisms for representing vocabularies: e.g., predicate logic, UML class diagrams, RDF Schema and OWL. While UML and RDF have a graphical notation, neither predicate logic nor OWL comes with any visual syntax. RDF provides a graphical notation for visualizing fact statements (more precisely, conjunctive sentences including terminological sentences involving classes and properties) in the form of directed labeled graphs with two kinds of nodes. UML class diagrams allow visualizing not only fact statements in the form of links relating two or more entities and/or data values, but also fact type expressions in the form of associations between types/classes.

The visual language of RDF graphs is semantically overloaded since it consists of only three language elements (two kinds of labeled node shapes and one kind of labeled arc arrow). Thus, it seems to be too poor

for being used to visualize real-world vocabularies. However, UML class diagrams have an extensive set of language elements and seem to be a good basis for visualizing vocabularies. Therefore, we suggest that any rule visualization effort should investigate and possibly adopt the ongoing work in the OMG for defining a standard representation of OWL ontologies as UML class diagrams.

Visualizing Derivation Rules

Derivation rules specify how certain logical sentences may be derived from others. They consist of one or more conditions and one or more conclusions. For specific types of derivation rules, such as definite Horn clauses or normal logic programs, the types of condition and conclusion are specifically restricted.

Previous work on the visualization of derivation rules is based on visualizing the dependency graph of a rule set or logic program in the form of an AND/OR tree. E.g., [DC91, BE91, NKD97] focus mainly on the visualization of proof trees and the control flow by displaying the success or failure of rules and the associated unification process. These works are motivated by the desire to support the debugging, and the execution analysis, of logic programs.

Other visualizations are motivated by the desire to increase the understanding of the logical structure of complex conditions in rules. Homogenous rule conditions can be presented in a compact and structured way in a decision table. Non-homogenous conditions are better represented in decision trees or fishbone diagrams.

Visualizing Reaction Rules

Reaction rules consist of a mandatory triggering event term, an optional condition, and a triggered action term or a post-condition (or both). There are basically two types of reaction rules: those that do not have a post-condition, which are the well-known Event-Condition-Action (ECA) rules, and those that do have a post-condition, which can be called ECAP rules. Reaction rules can be used for specifying the reactive behavior of a system and for expressing interaction patterns.

Rule Modeling	Diagrams	Semantically overloaded notation	Multiple rule firing	Rule interaction
AOR	Rules are visualized in interaction pattern diagrams	No	Yes	Yes
A/OODMT	Rules visualized in nested rule diagram, rule interaction diagrams, nested event model	No	Yes	Yes
$(ER)^2$	Rules are visualized in ECA^2 nets	Yes	Yes	Yes
IDEA	Rules visualized in class diagram	Yes	No	No
OMT-A and UML-A	Rules visualized in class diagrams and statechart diagrams	OMT_A: Yes diagram UML-A: No	Yes	Yes
OMT+	Rules visualized in class diagram	Yes	Yes	No

Table 1 Visualization of reaction rules

Several approaches for visualizing reactive rules have been suggested, most of them are found in the active database literature. Table 1 presents an overview on how reaction rules are visualized in the literature. Previous approaches tend to have problems with semantically overloaded notation for rule visualization. However, support for visualizing what happens if an event triggers several rules (multiple rule firing) and how rules interact (rule interaction) are in many cases acceptable.

The limitation with previous work on visualizing reactive rules is that most of them do not explicitly consider a distributed environment. Thus existing centralized approaches have to be refined to be useful, e.g., when modeling the interaction between two web resources.

Summary

Our survey in the previous sections shows that there are many possibilities to visualize rules in different ways with different purposes. We have argued that a graphical notation for rules in UML class diagrams would be of particular importance, since class diagrams visualize vocabularies (resp. ontologies) and rules are based on vocabularies. Such a notation would therefore allow showing:

- how rules are based on vocabularies
- how rules extend ontologies
- how rules can be used in a model-driven software engineering approach such as OMG's Model-Driven Architecture (MDA)

Based on our survey we can identify several requirements for such a visual rule language. In particular, we need:

-a graphical symbol for rules: Rule is a new metaclass to be added to the UML metamodel. A graphical symbol for rules should have a shape that is different from all other model element shapes currently used in UML class diagrams.

-a (semi-)graphical notation for certain types of logical formulas expressed in terms of other class modeling elements.

-a notation for event types.

For visualizing reaction rules, we need to distinguish two kinds of events:

- action events, which can be the result of firing a reaction rule that can trigger a reaction rule
- non-action events, such as time events, which cannot be the result of firing a reaction rule but which can trigger a reaction rule

Logical formulas may play the role of conditions, conclusions and postconditions in a rule. In each role, their syntax may be specifically restricted. It seems to be natural to use the UML symbol for states, a rectangle with rounded corners for expressing:

- status predicate conditions and conclusions by using the name of the Boolean attribute as the name of the state rectangle
- general state (post-)conditions by putting a Boolean OCL expression in the state rectangle (instead of a name)

How to express other types of atomic formulas is an issue for further research. This preliminary report is necessarily incomplete and leaves many issues unsolved. In particular, it would be important:

- to relate rule modeling and visualization to the three abstraction levels defined by the Model-Driven Architecture for the Object Management Group
- to integrate our rule modeling concepts with the UML metamodel
- to investigate what are the specific issues of modeling and visualizing Semantic Web rules based on RDF and OWL

Acknowledgements

This research has been funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (cf. http://rewerse.net).

References

[AB+04] Antoniou, A., Berndtsson, M., Spreeuwenberg, S., Taveter, K., and Wagner, G. A First-Version Visual Rule Language, Unpublished manuscript, I1-D1 deliverable, REWERSE, 2004. (available at http://rewerse.net)

[BE91] Brayshaw, M., Eisenstadt, M. A practical graphical tracer for Prolog. International Journal of Man-Machine Studies, 35(5):597 631, 1991.

[DC91] Dewar, A. D., Cleary, J. G. Graphical display of complex information within a Prolog debugger. International Journal of Man-Machine Studies, 25(5):503 521, 1991.

[NKD97] Neufeld, E., Kusalik, A., Dobrohoczki, M. Visual metaphors for understanding logic program execution. In: Davis, W., Mantel, M., Klassen, V. (eds.), Graphics Interfaces, pages 114 120, 1997.