A Metamodel and UML Profile for Rule-Extended OWL DL Ontologies

Saartje Brockmans, Peter Haase, Pascal Hitzler, and Rudi Studer

Institute AIFB, Universität Karlsruhe, Germany
{brockmans, haase, hitzler, studer}@aifb.uni-karlsruhe.de

Abstract. In this paper we present a MOF compliant metamodel and UML profile for the Semantic Web Rule Language (SWRL) that integrates with our previous work on a metamodel and UML profile for OWL DL. Based on this metamodel and profile, UML tools can be used for visual modeling of rule-extended ontologies.

1 Introduction

An ontology defines a common set of concepts and terms that are used to describe and represent a domain of knowledge. The manual creation of ontologies is a labor-intensive, expensive, often difficult, and – without proper tool support – an error-prone task. Visual syntaxes have shown to bring many benefits that simplify conceptual modeling [18]. As for other modeling purposes, visual modeling of ontologies decreases syntactic and semantic errors and increases readability. It makes the modeling and use of ontologies much easier and faster, especially if tools are user-friendly and appropriate modeling languages are applied.

The usefulness of a visual syntax for modeling languages has been shown in practice; visual modeling paradigms such as the Entity Relationship (ER, [4]) model or the Unified Modeling Language (UML, [5]) are used frequently for the purpose of conceptual modeling. Consequently, the necessity of a visual syntax for KR languages has been argued frequently in the past [6, 12]. Particular representation formalisms such as conceptual graphs [19] or Topic Maps [11], for example, are based on well-defined graphical notations.

Description Logic-based ontology languages such as OWL, however, are usually defined in terms of an abstract (text-based) syntax and most care is spent on the formal semantics, neglecting the development of good modeling frameworks. In our previous work [3], we therefore have developed a Meta Object Facility (MOF, [14]) metamodel for the purpose of defining ontologies, called Ontology Definition Metamodel (ODM), with specific focus on the OWL DL language, along with a UML profile for the purpose of visual modeling.

In the meantime, rule extensions for OWL have been heavily discussed [20]. Just recently the W3C has chartered a working group for the definition of a Rule Interchange Format [21]. One of the most prominent proposals for an extension of OWL DL with rules is the Semantic Web Rule Language (SWRL, [9]). SWRL proposes to allow the use of Horn-like rules together with OWL axioms.

Y. Sure and J. Domingue (Eds.): ESWC 2006, LNCS 4011, pp. 303–316, 2006.

A high-level abstract syntax is provided that extends the OWL abstract syntax described in the OWL Semantics and Abstract Syntax document [17]. An extension of the OWL model-theoretic semantics provides a formal meaning for SWRL ontologies.

The definition of a visual notation for SWRL rules is currently missing. Therefore, this paper defines a metamodel and UML profile for SWRL that extends and complements our previous metamodel and UML profile for OWL DL. Our goal is to achieve an intuitive notation, both for users of UML and description logics as well as for rule-based systems. Naturally, the proposed metamodel has a one-to-one mapping to the abstract syntax of SWRL and OWL DL and thereby to their formal semantics.

The paper is organized as follows: Section 2 introduces the Meta Object Facility (MOF) and our previous work on an OWL DL metamodel along with its UML profile. Section 3 presents our extensions of the ODM towards SWRL rules. Section 4 introduces a UML Profile for the modeling of rules and explains the major design choices made in order to make the notation readable and intuitive both for users with UML background and for users with a background in OWL and rule-based systems. In Section 5 we discuss related work. We conclude in Section 6 by summarizing our work and discussing future research.

2 An Ontology Definition Metamodel of OWL Within the MOF Framework

This section introduces the essential ideas of the Meta Object Facility (MOF) and shows how the Ontology Definition Metamodel (ODM) and the UML Ontology Profile (UOP) fit into this more general picture. The need for a dedicated visual ontology modeling language stems from the observation that an ontology cannot be sufficiently represented in UML [8]. The two representations share a set of core functionalities such as the ability to define classes, class relationships, and relationship cardinalities. But despite this overlap, there are many features which can only be expressed in OWL, and others which can only be expressed in UML. Examples for this disjointness are transitive and symmetric properties in OWL or methods in UML. For a full account of the conceptual differences we refer the reader to [10].

UML methodology, tools and technology, however, seem to be a feasible approach for supporting the development and maintenance of ontologies. The general idea of using MOF-based metamodels and UML profiles for this purpose is depicted in Figure 1 and explained in the following:

1. The ODM and the UOP are grounded in MOF, in that they are defined in terms of the MOF meta-metamodel, as explained in Section 2.1.
2. The UML profile defines a visual notation for OWL DL ontologies, based on the above mentioned metamodel. Furthermore, mappings in both directions between the metamodel and this profile are established.
3. Specific OWL DL ontologies instantiate the Ontology Definition Metamodel. The constructs of the OWL DL language have a direct correspondence with

those of the ODM. Analogously, specific UML models instantiate the UML Ontology Profile. The translation between the UML models and OWL ontologies is based on the above mappings between the ODM and the UOP.

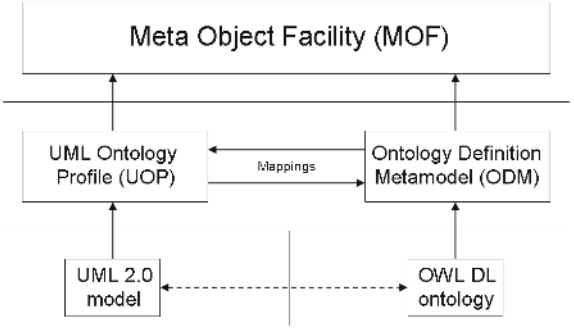

Fig. 1. An ontology to UML mapping allows existing tools to operate on compatible aspects of ontologies

2.1 Meta Object Facility

The Meta Object Facility (MOF) is an extensible model driven integration framework for defining, manipulating and integrating metadata and data in a platform independent manner. The goal is to provide a framework that supports any kind of metadata and that allows new kinds to be added as required. MOF plays a crucial role in the four-layer metadata architecture of the Object Management Group (OMG) shown in Figure 2. The bottom layer of this architecture encompasses the raw information to be described. For example, Figure 2 contains information about a wine called ElyseZinfandel and about the Napa region, where this wine grows. The model layer contains the definition of the required structures, e.g. in the example it contains the classes used for grouping information. Consequently, the classes wine and region are defined. If these are combined, they describe the model for the given domain. The metamodel defines the terms in which the model is expressed. In our example, we would state that models are expressed with classes and properties by instantiating the respective meta classes. Finally, the MOF constitutes the top layer, also called the meta-metamodel layer. Note that the top MOF layer is hard wired in the sense that it is fixed, while the other layers are flexible and allow to express various metamodels such as the UML metamodel or the ODM.

2.2 Ontology Definition Metamodel

The Ontology Definition Metamodel (ODM, [3]) defines a metamodel for ontologies. This metamodel is built on the MOF framework, which we explained in Section 2.1. We defined an Ontology Definition Metamodel for OWL DL using a notation which is accessible for users of UML as well as for OWL DL ontology engineers. A metamodel for a language that allows the definition of ontologies

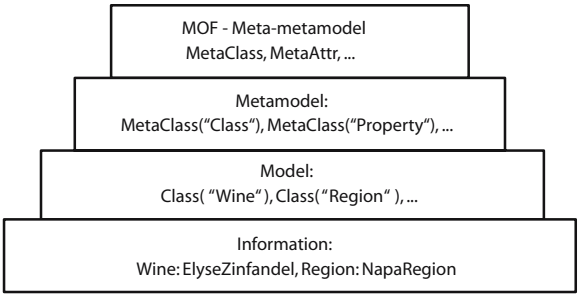

Fig. 2. OMG Four Layer Metadata Architecture

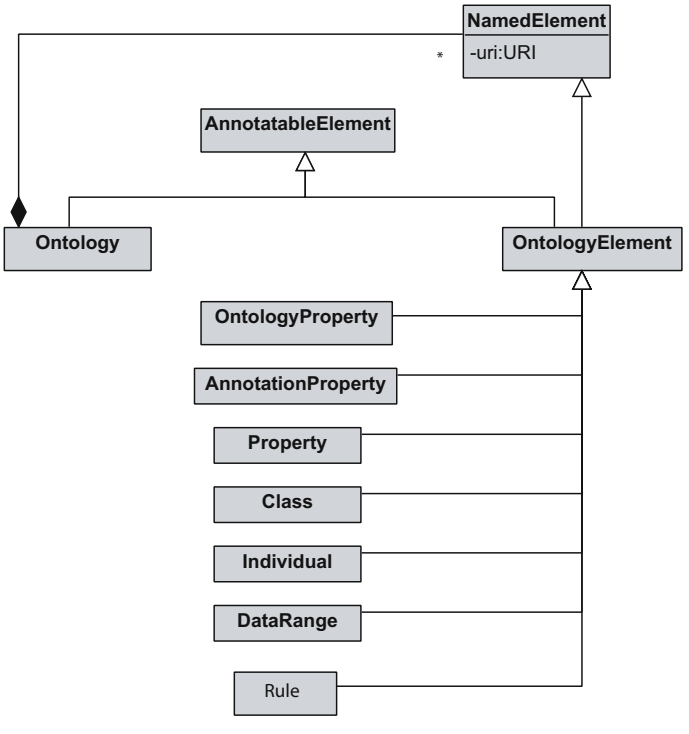

Fig. 3. Main Elements of the Ontology Definition Metamodel

naturally follows from the modeling primitives offered by the ontology language. The proposed metamodel has a one-to-one mapping to the abstract syntax of OWL DL and thereby to the formal semantics of OWL. It primarily uses basic well-known concepts from UML2, which is the second and newest version of UML. Additionally, we augmented the metamodel with constraints specifying invariants that have to be fulfilled by all models that instantiate the metamodel. These constraints are expressed in the Object Constraint Language [23], a declar-

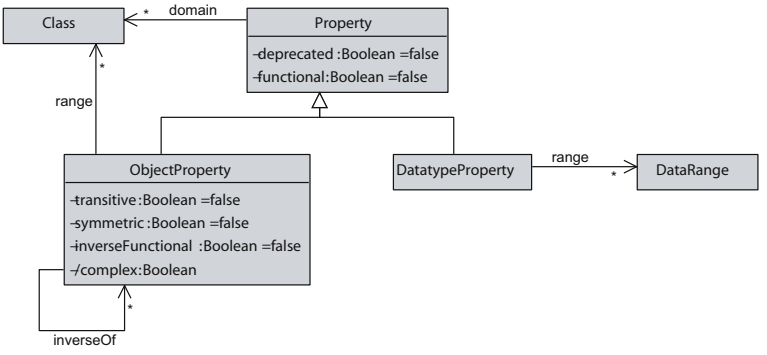

Fig. 4. Properties

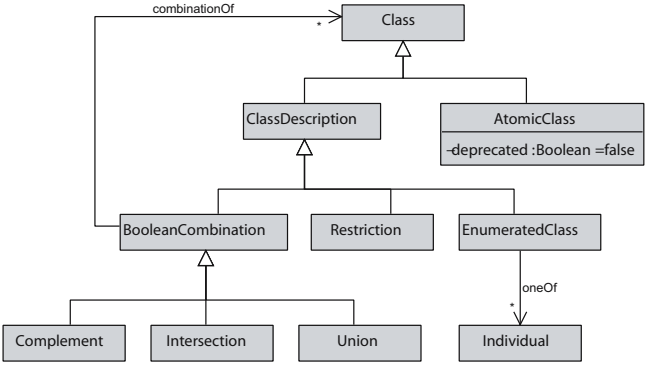

Fig. 5. Classes

ative language that provides constraint and object query expressions on object models that cannot otherwise be expressed by diagrammatic notation.

Figure 3 shows the main elements of the ODM. Every element of an ontology is a `NamedElement` and hence a member of an `Ontology`.

Properties, as shown in Figure 4, represent named binary associations in the modeled knowledge domain. OWL distinguishes two kinds of properties, so-called object properties and datatype properties. A common generalization of them is given by the abstract metaclass `Property`. Properties can be functional and their domain is always a class. Object properties may additionally be inverse functional, transitive, symmetric, or inverse to another property. Their range is a class, while the range of datatype properties are datatypes.

Users can relate properties by using two types of axioms: property subsumption (`subPropertyOf`) specifies that the extension of a property is a subset of the related property, while property equivalence (`equivalentProperty`) defines extensional equivalence.

Class descriptions are depicted in Figure 5. In contrast to UML, OWL DL does not only allow to define simple named classes. Instead, classes can be formed using

a number of class constructors. One can conceptually distinguish the boolean combination of classes, class restrictions, and enumerated classes. `EnumeratedClass` is defined through a direct enumeration of named individuals. Boolean combinations of classes are provided through `Complement`, `Intersection` and `Union`.

OWL does not follow the clear conceptual separation between terminology (T-Box) and knowledge base (A-box) that is present in most description logics and in MOF, which distinguishes between model and information. The knowledge base elements (cf. Figure 6) are part of an ontology. An `Individual` is an instantiation of a `Class` and is the subject of a `PropertyValue`, which instantiates a `Property`. Naturally, an `ObjectPropertyValue` relates its subject with another `Individual` whilst a `DatatypePropertyValue` relates its subject with a `DataValue`, which is an instance of a `DataType`.

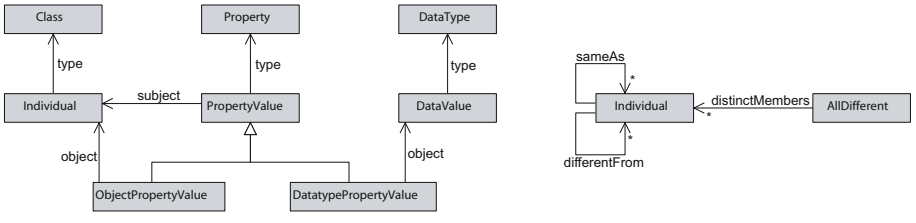

Fig. 6. Knowledge Base

Individuals can be related via three special axioms: The `sameAs` association allows users to state that two individuals (with different names) are equivalent. The `differentFrom` association specifies that two individuals are not the same. `AllDifferent` is a simpler notation for the pairwise difference of several individuals.

For a full specification of the OWL DL metamodel, we refer to [3].

2.3 UML Ontology Profile

The UML ontology profile (UOP) describes a visual UML syntax for modeling ontologies. We provide a UML profile that is faithful to both UML2 and OWL DL, with a maximal reuse of UML2 features and OWL DL features. Since the UML profile mechanism supports a restricted form of metamodeling, our proposal contains a set of extensions and constraints to UML2. This tailors UML2 such that models instantiating the ODM can be defined. Our UML profile has a basic mapping, from OWL class to UML class, from OWL property to binary UML association, from OWL individual to UML object, and from OWL property filler to UML object association. Extensions to UML2 consist of custom UML stereotypes, which usually carry the name of the corresponding OWL DL language element, and dependencies. Figure 7 (left) shows an example of two classes `Wine` and `WineGrape`, visually depicted as UML classes, which are connected via the object property `madeFromGrape`, depicted as a UML association. Some extensions to UML2 are used in the example in Figure 7 (right),

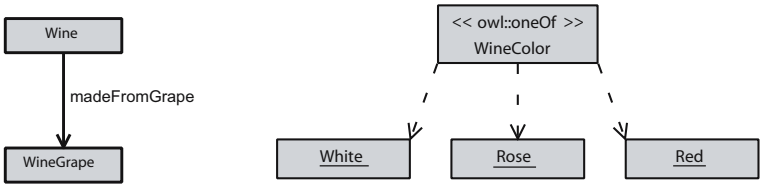

Fig. 7. A fragment of the UML profile: The ObjectProperty and oneOf constructs

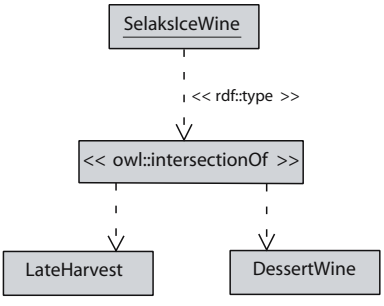

Fig. 8. A fragment of the UML profile: An individual of a complex class description

which shows that an enumerated class is connected to the enumerated individuals by dependencies. A stereotype denotes the enumerated class, whereas the UML notation for objects is used for individuals. Another example, depicted in Figure 8, shows an individual `SelaksIceWine` of the intersection between the classes `LateHarvest` and `DessertWine`.

3 A Metamodel for SWRL Rules

We propose a metamodel for SWRL rules as a consistent extension of the metamodel for OWL DL ontologies which we described in the previous section of this paper. Figure 9 shows the metamodel for SWRL rules. We discuss the metamodel step by step along the SWRL specifications. Interested readers may refer to the specifications [9] for a full account of SWRL. For a complete reference of the formal correspondence between the metamodel and SWRL itself and the OCL constraints for the rule metamodel, we refer the reader to [2].

3.1 Rules

SWRL defines rules as part of an ontology. The SWRL metamodel defines `Rule` as a subclass of `OntologyElement`. `OntologyElement` is defined in the OWL DL metamodel (Figure 3) as an element of an `Ontology`, via the composition link between `NamedElement` and `Ontology`. As can also be seen in Figure 3, the class `OntologyElement` is a subclass of the class `AnnotatableElement`, which defines that rules can be annotated. As annotations are modeled in the ODM, a URI reference can be assigned to a rule for identification.

A rule consists of an antecedent and a consequent, also referred to as body and head of the rule, respectively. Both the antecedent and the consequent consist of a set of atoms which can possibly be empty, as depicted by the multiplicity in Figure 9. Informally, a rule says that *if* all atoms of the antecedent hold, *then* the consequent holds. An empty antecedent is treated as trivially true, whereas an empty consequent is treated as trivially false.

The same antecedent or consequent can be used in several rules, as indicated in the metamodel by the multiplicity on the association between `Rule` on the one hand and `Antecedent` or `Consequent` on the other hand. Similarly, the multiplicities of the association between `Antecedent` and `Atom` and of the association between `Consequent` and `Atom` define that an antecedent and a consequent can hold zero or more atoms. The multiplicity in the other direction defines that the same atom can appear in several antecedents or consequents. According to the SWRL specifications, every `Variable` that occurs in the `Consequent` of a rule must occur in the `Antecedent` of that rule, a condition referred to as "safety".

3.2 Atoms, Terms and Predicate Symbols

The atoms of the antecedent and the consequent consist of predicate symbols and terms. According to SWRL, they can have different forms:

- $C(x)$, where C is an OWL description and x an individual variable or an OWL individual, or C is an OWL data range and x either a data variable or an OWL data value;
- $P(x, y)$, where P is an OWL individual valued property and x and y are both either an individual variable or an OWL individual, or P is an OWL data valued property, x is either an individual variable or an OWL individual and y is either a data variable or an OWL data value;

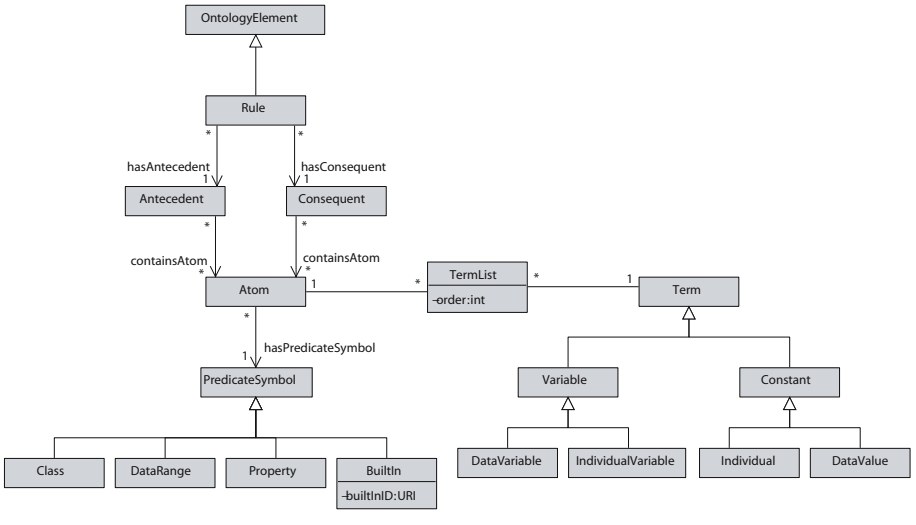

Fig. 9. The Rule Definition Metamodel

- sameAs(x, y), where x and y are both either an OWL individual or an individual variable;
- differentFrom(x, y), where x and y are both either an OWL individual or an individual variable;
- builtIn$(r, x, ...)$, where r is a built-in predicate and x is a data variable or OWL data value. A builtIn atom could possibly have more than one variable or OWL data value.

The first of these, OWL description, data range and property, were already provided in the ODM, namely as metaclasses `Class`, `DataRange` and `Property`, respectively. As can be seen in Figure 9, the predicates `Class`, `DataRange`, `Property` and `BuiltIn` are all defined as subclasses of the class `PredicateSymbol`, which is associated to `Atom`. The remaining two atom types, `sameAs` and `differentFrom`, are represented as specific instances of `PredicateSymbol`.

To define the order of the atom terms, we put a class `TermOrder` in between `Atom` and `Term`. This UML association class connects atoms with terms and defines the term order via the attribute `order`.

4 A UML Profile for Rules

UML provides an extension mechanism, the UML profile mechanism, to tailor the language to specific application areas. The definition of such a UML extension is based on the standard UML metamodel. In this section, we propose a UML profile for modeling SWRL rules which is consistent with the design considerations taken for the basic UML Ontology Profile. For a complete reference of the relationship between the UML profile and the metamodel introduced in Section 3, we refer the reader to [2]. Figure 10 shows an example of a rule, which defines that when a vintager does not like the wine made in his winery, he is a bad vintager. We introduce the profile in an order based on the SWRL metamodel introduced in Section 3.

4.1 Rules

As can be seen in Figure 10, a rule is depicted by two boxes connected via a dependency with the stereotype `rule`. All atoms of the antecedent are contained

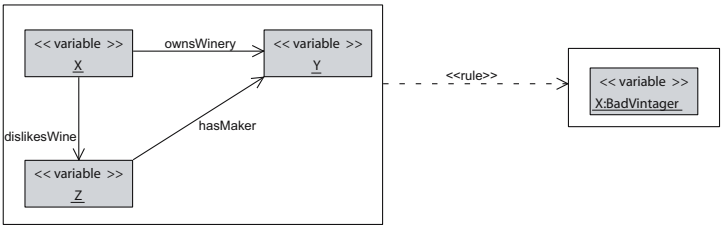

Fig. 10. BadVintager$(x) \leftarrow$ ownsWinery$(x, y) \wedge$ dislikesWine$(x, z) \wedge$ hasMaker(z, y)

in the box at the origin of the dependency, whereas the box at the end contains the consequent. This way, antecedent and consequent can easily be distinguished, and it also allows to distinguish between the rule atoms and the OWL DL facts which are depicted in similar ways. The left box of our example contains the three variable definitions and the three properties that are defined between these variables. The consequent-box on the right contains the definition of the variable X from which it is known which class it belongs to. We explain the specific design considerations of these concepts in the following subsections.

4.2 Terms

Although the existing UOP already comprises a visual syntax for individuals and data values, namely by applying the UML object notation, it does not include a notation for variables since OWL DL ontologies do not contain variables. We decided to depict variables in the UML object notation as well, since a variable can be seen as a partially unknown class instance. We provide a stereotype `variable` to distinguish a variable. Figure 11 shows a simple example for a variable, an individual and a data value.

Fig. 11. Terms

4.3 Predicate Symbols in Atoms

Class description and data range. A visual notation for individuals as instances of class descriptions is already provided in the UOP for OWL DL. An atom with a class description and a variable as its term, is illustrated similarly. An appropriate stereotype is added. An example of this can be seen in the consequent in Figure 10. A visual construct for a data range definition using individuals is contained in the UOP for OWL DL as well, namely represented in the same way as class individuals. Data range constructs containing variables are also depicted in a similar fashion.

Properties. Object properties are depicted as directed associations between the two involved elements. A datatype property is pictured as an attribute. These notations were provided for properties of individuals by the UOP for OWL DL, and we follow them to depict properties of variables. The antecedent of the rule in Figure 10 contains three such object properties between variables, `ownsWinery`, `dislikesWine` and `hasMaker`. The other example rule, depicted in Figure 13, contains amongst other things twice the datavalued property `yearValue`.

sameAs and differentFrom. According to the UOP, equality and inequality between objects are depicted using object relations. Because of the similarity between individuals and variables, as shortly explained in Section 4.2, we propose to use the same visual notation for `sameAs` and `differentFrom` relations between two variables or between a variable and an object.

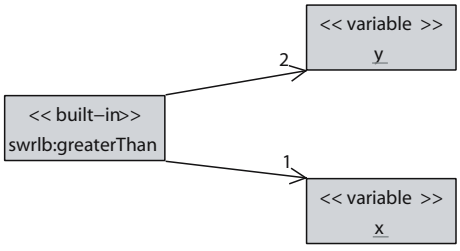

Fig. 12. Built-in predicates

Built-in predicates. For the visual representation of built-in relations, we use usual associations to all participating variables and data values, similar to the `owl:AllDifferent` concept provided in the basic UOP. To denote the built-in relation, we provide the stereotype `built-in` together with the specific built-in ID. The names of the associations denote the order of the arguments, by numbers. Figure 12 shows an example of a built-in relation `swrlb:greaterThan`, which is defined to check whether the first involved argument is greater than the second one. For the six most basic built-ins, `swrlb:equal`, `swrlb:notEqual`, `swrlb:lessThan`, `swrlb:lessThanOrEqual`, `swrlb:greaterThan` and `swrlb:greaterThanOrEqual`, we provide an alternative notation. Instead of depicting the stereotype and the name of the built-in, an appropriate icon can be used. Figure 13 depicts a rule example using this alternative notation for built-in predicates. This rule states that if the year value of a wine (y) is greater than the year value of another wine (x), then the second wine (x) is older than the first one (y). Next to the built-in predicate, Figure 13 shows six variables with the properties `hasVintageYear`, `yearValue` and `olderThan`.

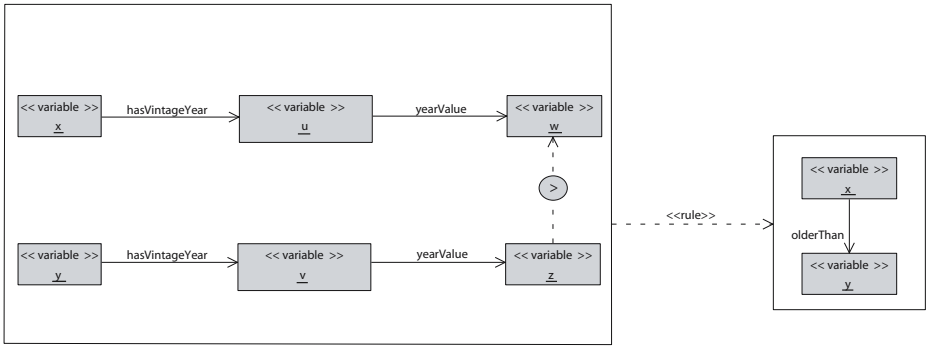

Fig. 13. olderThan$(x, y) \leftarrow$ hasVintageYear$(x, u) \wedge$ hasVintageYear$(y, v) \wedge$ yearValue$(u, w) \wedge$ yearValue$(v, z) \wedge$ swrlb:greaterThan(z, w)

5 Related Work

As a response to the original call of the OMG for an Ontology Definition Metamodel [16], the OMG has received a number of diverse proposals (see [3] for a comparison). The various proposals have been merged into one submission [10] that covered several metamodels for RDF, OWL, Common Logic, and Topic Maps, as well as mappings between them. Our proposed metamodel departs from this approach as it strictly focuses on OWL DL and is tailored to its specific features, with the advantage that it has a direct mapping between the metamodel and OWL DL. Also, none of the other OMG proposals so far has considered rule extensions. To the best of our knowledge, our work presents the first MOF-based metamodel and UML profile for an ontology rule language.

DL-safe rules [13] are a decidable subset of SWRL. As every DL-safe rule is also a SWRL rule, DL-safe rules are covered by our metamodel. Using additional constraints it can be checked whether a rule is DL-safe. It should be noted that SWRL is not the only rule language which has been proposed for ontologies. Other prominent alternatives for rule languages are mentioned in the W3C Rule Interchange Format Working Group charter [21], namely the Web Rule Language WRL [1] and the rules fragment of the Semantic Web Service Language SWSL [7]. These languages differ in their semantics and consequently also in the way in which they model implicit knowledge for expressive reasoning support. From this perspective, it could be desirable to define different metamodels, each of which is tailored to a specific rules language.

From the perspective of conceptual modeling, however, different rule languages appear to be very similar to each other. This opens up the possibility to reuse the SWRL metamodel defined in this paper by augmenting it with some features to allow for the modeling of language primitives which are not present in SWRL. As a result, one would end up with a common metamodel for different rules languages. An advantage of the latter approach would be a gain in flexibility. Intricate semantic differences between different ontology languages may often be difficult to understand for the practitioner, and hence it may be desirable to provide simplified modeling support in many cases. A common visual modeling language would for example allow a domain expert to model a domain independent of a concrete logical language, while an ontology engineer could decide on the language paradigm most suitable for the application domain.

As a complementary approach to using visual modeling techniques for writing rules, [22] discusses a proposal to use (controlled) natural language.

6 Conclusion

We have presented a MOF metamodel for the Semantic Web Rule Language SWRL. This metamodel tightly integrates with our previous metamodel for OWL DL. The validity of instances of this metamodel is ensured through OCL constraints. We also provided a UML profile for this metamodel. It employs the extensibility features of UML2 to allow a visual notation for the modeling of

rule-extended ontologies which is particularly adequate for users familiar with UML. We plan to provide an implementation as a next step.

Future work may also include the modeling of other logics-based rule languages. The outcome of the W3C working group to establish a Rule Interchange Format is currently open. It is likely that several rule languages will need to co-exist, which will require techniques for rule language interoperability. Here, the model driven approaches of MOF might provide useful techniques to achieve such interoperability, for example based on the Query View and Transformation (QVT, [15]) framework, which allows the definition and automated execution of mappings between MOF-based metamodels.

Acknowledgements

Research for this paper has been partially funded by the EU under the projects SEKT (IST-2003-506826) and NeOn (IST-2005-027595), by the German Federal Ministry of Education and Research (BMBF) under the SmartWeb project (01 IMD01 A) and by the German Research Foundation (DFG) under the Graduate School IME – Universität Karlsruhe (TH).

References

1. J. Angele, H. Boley, J. de Bruijn, D. Fensel, P. Hitzler, M. Kifer, R. Krummenacher, H. Lausen, A. Polleres, and R. Studer. *Web Rule Language (WRL).* World Wide Web Consortium, September 2005. W3C Member Submission, `http://www.w3.org/Submission/WRL/`.
2. S. Brockmans and P. Haase. A Metamodel and UML Profile for Rule-extended OWL DL Ontologies –A Complete Reference. Technical report, Universität Karlsruhe, March 2006. `http://www.aifb.uni-karlsruhe.de/WBS/sbr/publications/owl-metamodeling.pdf`.
3. S. Brockmans, R. Volz, A. Eberhart, and P. Loeffler. Visual Modeling of OWL DL Ontologies using UML. In F. van Harmelen, S. A. McIlraith, and D. Plexousakis, editors, *The Semantic Web – ISWC 2004*, pages 198–213. Springer-Verlag, 2004.
4. P. P. Chen. The entity-relationship model – toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.
5. M. Fowler. *UML Distilled.* Addison-Wesley, third edition, 2004.
6. B. R. Gaines. An Interactive Visual Language for Term Subsumption Languages. In J. Mylopoulos and R. Reiter, editors, *Proc. of 12th Int. Joint Conf. on Art. Int.*, pages 817–823, Sydney, Australia, August 1991. Morgan Kaufmann.
7. B. Grosof, M. Kifer, and D. L. Martin. Rules in the Semantic Web Services Language (SWSL): An overview for standardization directions. In *Proceedings of the W3C Workshop on Rule Languages for Interoperability, 27-28 April 2005, Washington, DC, USA*, 2005.
8. L. Hart, P. Emery, B. Colomb, K. Raymond, S. Taraporewalla, D. Chang, Y. Ye, and M. D. Elisa Kendall. OWL full and UML 2.0 compared, March 2004. `http://www.itee.uq.edu.au/$\sim$colomb/Papers/UML-OWLont04.03.01.pdf`.

9. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. World Wide Web Consortium, May 2004. W3C Member Submission, http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.
10. IBM, Sandpiper Software. *Ontology Definition Metamodel, Fourth Revised Submission to OMG*, November 2005.
11. ISO/IEC. Topic Maps: Information Technology – Document Description and Markup Languages. ISO/IEC 13250, http://www.y12.doe.gov/sgml/sc34/document/0129.pdf, December 1999.
12. R. Kremer. Visual Languages for Knowledge Representation. In *Proc. of 11th Workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*, Voyager Inn, Banff, Alberta, Canada, April 1998. Morgan Kaufmann.
13. B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. In F. van Harmelen, S. McIlraith, and D. Plexousakis, editors, *International Semantic Web Conference*, Lecture Notes in Computer Science, pages 549–563, Hiroshima, Japan, 2004. Springer.
14. Object Management Group. Meta Object Facility (MOF) Specification. Technical report, Object Management Group (OMG), April 2002. http://www.omg.org/docs/formal/02-04-03.pdf.
15. Object Management Group. MOF 2.0 Query / Views / Transformations – Request for Proposal. http://www.omg.org/docs/ad/02-04-10.pdf, 2002.
16. Object Management Group. Ontology Definition Metamodel – Request For Proposal, March 2003. http://www.omg.org/docs/ontology/03-03-01.rtf.
17. P. F. Patel-Schneider, P. Hayes, and I. Horrocks. *OWL Web Ontology Language Semantics and Abstract Syntax*. World Wide Web Consortium, 10. Februar 2004. Recommendation. http://www.w3.org/TR/2004/REC-owl-semantics-20040210/.
18. W. Schnotz. Wissenserwerb mit Texten, Bildern und Diagrammen. In L. J. Issing and P. Klimsa, editors, *Information und Lernen mit Multimedia und Internet*, pages 65–81. Belz, PVU, Weinheim, third, completely revised edition, 2002.
19. J. F. Sowa. Conceptual graphs summary. In P. Eklund, T. Nagle, J. Nagle, and L. Gerholz, editors, *Conceptual Structures: Current Research and Practice*, pages 3–52. Ellis Horwood, New York, 1992.
20. *Accepted Papers of the W3C Workshop on Rule Languages for Interoperability, 27-28 April 2005, Washington, DC, USA*, 2005. http://www.w3.org/2004/12/rules-ws/accepted.
21. W3C. Rule interchange format working group charter. http://www.w3.org/2005/rules/wg/charter, 2005.
22. A. Walker. Understandability and semantic interoperability of diverse rules systems. http://www.w3.org/2004/12/rules-ws/paper/19, April 2005. Position Paper for the W3C Workshop on Rule Languages for Interoperability.
23. J. Warmer and A. Kleppe. *Object Constraint Language 2.0*. MITP Verlag, 2004.