FORML Position paper for W3C Workshop on Rule Languages for Interoperability

Terry Halpin
Northface University, USA
terry.halpin@northface.edu

Northface University emphasizes model-driven development in its teaching and research programs as a key technology for empowering software engineering, is an active contributor to the Business Rules Team submission to the OMG's RFP for adding a business semantics for business rules layer, and is currently developing an open source modeling tool suite based largely on ORM 2 (the next generation of Object-Role Modeling), which is intended to be interoperable with a wide range of other rules-based tools. As such, we are interested in efforts to standardize languages for business rules to facilitate:

- Rule declaration that can be readily understood and validated by non-technical domain experts;
- Rule enforcement (automated, semi-automated, and manual);
- Rule reuse and exchange with other tools, especially over the web.

While we value much of the work contributed to other proposed rule languages such as RDF, OWL, SWRL etc., we feel that rules expressed directly in those languages are often too technically demanding to be readily understood by the business rule owners (subject matter experts who are typically not experts in mathematical syntax).

As part of our efforts to address this problem, we are extending and refining ORM, a fact-oriented approach for specifying, transforming, and querying information models at the semantic level that has been used productively in industry for over 30 years [3, 4], to its next generation (ORM 2), as well as building open source tool support for it. ORM 2 includes a new version of FORML (Formal Object-Role Modeling Language), a very high level language for specifying and querying information models including business rules, as well as a set of procedures for transforming the rules into code in a variety of implementations (relational database, object-oriented code, XML etc.). Unlike UML and ER, ORM expresses all facts in terms of logical predicates, rather than using attributes as a base constructs [5].

FORML has both graphical and textual versions, with automatic translation between the graphical notation and the core of the (more expressive) textual notation. The new version of FORML under development extends the conceptual modeling capabilities of the earlier FORML language supported by Microsoft's ORM tool [7], while also extending the conceptual query capabilities of the ConQuer-II language supported by ActiveQuery [1].

While formally grounded in first-order logic plus bag comprehension, the surface syntax of FORML is high level enough to enable most business rules to be declared in ways that are easily understood by non-technical business users. Mixfix predicates of any arity (unary upwards) are supported, and arbitrary object-variable correlation is supported via use of pronouns (when unambiguous) or subscripted variables. The main design criteria for FORML are:

- *expressibility* the language is able to express a wide range of business rules
- *clarity* the rules are understandable by non-technical domain experts
- *flexibility* the language directly supports predicates of any arity
- *localizability* the language constructs are expressible in different native languages
- *formality* the rules are unambiguous, and should ideally be executable

For our purposes, *business rules* are either *constraints* or *derivation rules*. Some simple examples of business rules expressed in FORML are now given. A more detailed discussion of such business rule verbalizations may be found in [6].

1. A uniqueness constraint spanning the first two roles of the ternary association Room at HourSlot is booked for Course may be expressed in *positive form* thus:

Given any Room and HourSlot
that Room at that HourSlot is booked for at most one Course

Every constraint has an associated *modality*, determined by the logical modal operator that functions explicitly or implicitly as its main operator. In positive verbalizations, the modality is often assumed, but may be explicitly prepended. For the constraint just considered, we have:

[It is necessary that | It is obligatory that] -- [alethic | deontic]
given any Room and HourSlot
that Room at that HourSlot is booked for at most one Course

The deontic version indicates that the rule *ought* to be obeyed, while recognizing that the constraint *might* be violated. Most business constraints are deontic in nature. Negative verbalizations typically make the modality explicit. For example, the deontic constraint above may be verbalized thus:

It is forbidden that
the same Room at the same HourSlot
is booked for more than one Course

2. Consider a uniqueness constraint over the 1st, 2nd and 4th roles of the quaternary fact type Person judged Person to have SkillLevel in Language. Here the same object type Person plays more than one role in the rule. In such cases, it is convenient to append numbered *subscripts* to distinguish the object variables. The constraint may now be verbalized as follows.

   Given any $Person_1$, $Person_2$ and Language
   $Person_1$ judged $Person_2$ to have at most one SkillLevel in that Language.

3. An exclusion constraint between the binary associations Person directed Movie and Person reviewed Movie may be specified in relational style thus:

   No Person directed and reviewed the same Movie.

   ORM 2 supports rule verbalization in *relational style* (using predicate names), *attribute style* (using association role names) and a *mixture* of both. Assuming the relevant role names "director" and "reviewer" are declared, the exclusion constraint above may be verbalized in attribute style thus:

   For each Movie
   no director is a reviewer.

4. Consider a subset constraint from the set of advisor-country pairs instantiating the Advisor serves-in Country association to the set of advisor-country pairs projected from a schema path that conceptually joins the Language roles in the associations Advisor speaks language, and Language is used by Country. This constraint may be verbalized thus:

   Each Advisor who serves in a Country also speaks a Language that is used by that Country.

Derivation rules may be specified to derive an object type or a fact type from other types.

5. For example, given the fact types City is in Country and City has Population we may define the *subtype* LargeUSCity thus:

   Each LargeUSCity is a City that is in Country 'US' and has Population >= 1000000.

6. A derivation rule for the fact type Person is an uncle of Person may be declared in relational style thus:

   Define $Person_1$ is an uncle of $Person_2$ as
   $Person_1$ is a brother of some $Person_3$ who is a parent of $Person_2$

   Using role names, the derivation rule may be specified in attribute style thus:

   For each Person: uncle = brother of parent.

References

1. Bloesch, A. & Halpin, T. 1997, 'Conceptual queries using ConQuer-II', *Proc. ER'97: 16th Int. Conf. on conceptual modeling*, Springer LNCS, no. 1331, pp. 113-26.
2. Halpin, T. 1989, 'A Logical Analysis of Information Systems: static aspects of the data-oriented perspective', doctoral dissertation, University of Queensland.
3. Halpin, T. 1998, 'ORM/NIAM Object-Role Modeling', *Handbook on Information Systems Architectures*, eds P. Bernus, K. Mertins & G. Schmidt, Springer-Verlag, Berlin, pp. 81-101.
4. Halpin, T. 2001, *Information Modeling and Relational Databases*, Morgan Kaufmann, San Francisco.
5. Halpin, T. 2004, 'Comparing Metamodels for ER, ORM and UML Data Models', *Advanced Topics in Database Research, vol. 3*, ed. K. Siau, Idea Publishing Group, Hershey PA, USA, Ch. II (pp. 23-44).
6. Halpin, T. 2004, 'Business Rule Verbalization', *Information Systems Technology and its Applications*, Proc. ISTA-2004, (eds Doroshenko, A., Halpin, T. Liddle, S. & Mayr, H), Salt Lake City, Lec. Notes in Informatics, vol. P-48, pp. 39-52.
7. Halpin, T., Evans, K, Hallock, P. & MacLean, W. 2003, *Database Modeling with Microsoft® Visio for Enterprise Architects*, Morgan Kaufmann, San Francisco.