

Verbalizing Business Rules: Part 7

Terry Halpin
Northface University

Business rules should be validated by business domain experts, and hence specified using concepts and languages easily understood by business people. This is the seventh in a series of articles on expressing business rules formally in a high-level, textual language. The first article [4] discussed criteria for a business rules language, and verbalization of simple uniqueness and mandatory constraints on binary associations. The second article [5] examined hyphen-binding, and verbalization of internal uniqueness constraints that span a whole association, or that apply to n-ary associations. The third article [6] covered verbalization of basic external uniqueness constraints. The fourth article [7] considered relational-style verbalization of external uniqueness constraints involving nesting or long join paths, as well as attribute-style verbalization of uniqueness constraints and simple mandatory constraints. The fifth article [8] discussed verbalization of mandatory constraints on roles of n-ary associations, and disjunctive mandatory constraints (also known as inclusive-or constraints) over sets of roles. The seventh article [9] considered verbalization of value constraints. This article discusses verbalization of subset constraints.

Set-comparison constraints

Recall that an association *role* (or role for short) is simply a part played in a relationship. Two roles are said to be *compatible* if and only if they are played by the same object type, or their object types have a common supertype. A *role sequence* is an ordered list of one or more roles. Two role sequences are compatible if and only if they have the same number of roles, and their corresponding roles are compatible. If two roles or role sequences are compatible, it is meaningful to compare their populations (sets of fact instances).

Set-comparison constraints restrict the way the population of one role (or role sequence) compares with that of another compatible role (or role sequence). In principle, there are just three kinds of set-comparison constraint:

- Subset constraint
- Equality constraint
- Exclusion constraint

Although sets may be related in other ways (e.g. proper subset, overlap, and proper overlap), these set comparisons require that at least one of the sets has a non-empty population, and hence cannot be used as a static constraint (which must apply to *each* state of the business domain, including an empty domain). Recall that information systems typically start their life with an empty database, so any static constraint must apply to that as well.

Of the information modeling approaches used in industry, only Object-Role Modeling (ORM) provides complete, built-in support for set-comparison constraints. The Unified Modeling language (UML) provides limited support for subset constraints (when they apply between whole associations), and very limited support for exclusion constraints (when they occur within an exclusive-or constraint). Within UML, most set comparison constraints need to be expressed in a textual language such as OCL. Entity Relationship (ER) modeling typically provides no support at all for set comparison constraints. Although subset constraints may sometimes be modeled alternatively using subtyping, this alternative is often unnatural and awkward for such cases. In this article we discuss how to verbalize subset constraints. The next article considers verbalization of equality and exclusion constraints.

Verbalization of subset constraints between single roles

Figure 1 shows a simple ORM model about hospital patients. There are two subset constraints, each of which applies between two single roles. A subset constraint is displayed as a circled “ \subseteq ” within a dotted arrow that runs from the subset role to the superset role.

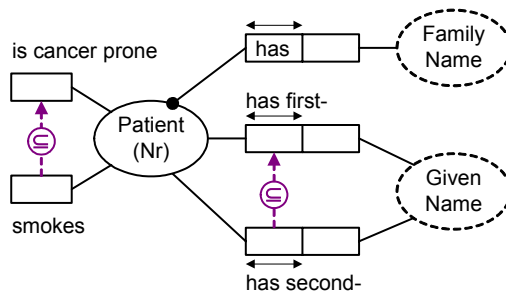


Figure 1 Subset constraints between single roles.

Consider the subset constraint between the unary fact types Patient smokes, Patient is cancer prone. This declares that for each state of the business domain, the set of patients who play the smokes role is a subset of the patients that play the is-cancer-prone role. The other subset constraint indicates that the set of patients who have a second given name is always a subset of the set of patients who have a first given name (it is possible in some cultures that no given name is used). These constraints may be formally verbalized thus:

Each Patient who smokes also is cancer prone.
Each Patient who has a second GivenName also has a first GivenName.

The pronoun “who” may be replaced by “that” or “which”. The quantifier “a” may be replaced by “some”, “at least one”, or “an”. As explained in an earlier article [5], the use of hyphens in the GivenName predicates binds the adjectives “first” and “second” to the object type name, so the quantifier “a” precedes them in the verbalization. An alternative verbalization uses the logical if-then operator:

If a Patient smokes then that Patient is cancer prone.
If a Patient has a second GivenName then that Patient has a first GivenName.

Figure 2 shows a UML class diagram for our patient example. Here the fact types are all modeled as attributes. Given a closed world interpretation of the unary fact types, these are modeled as mandatory, Boolean (true/false) attributes. As there is no graphic way of depicting the subset constraints in UML, these have been expressed in OCL, using an attached note. Because the note is attached to the Patient class, this is understood to provide the context for the OCL constraints. Clearly, each of the verbalization patterns shown above provides a higher level declaration of the constraint that is more likely to be understood by a non-technical domain expert.

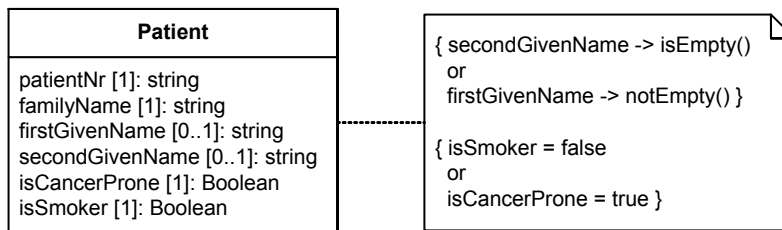


Figure 2 The ORM model from Figure 1 expressed in UML.

In addition to the relational-style verbalization already discussed, an attribute-style verbalization may be provided, as follows. Although less natural than the relational-style verbalization, it is likely to be more understandable to non-technical people than the OCL formulation.

For each Patient:
 not exists secondGivenName or exists firstGivenName;
 not isSmoker or isCancerProne

Note that this verbalization may also be applied directly to the ORM model, so long as we supply the attribute names used in the UML model as corresponding role names on the ORM model.

Verbalization of subset constraints between role sequences

Now suppose that a patient may undertake zero or more tests, and that for each test we record whether or not he/she passed that test. This may be modeled in ORM or UML as shown in

Figure 3. Here the subset constraint is from the pair of roles comprising the association Person passed Test to the pair of roles forming the association Person took Test, indicating that the set of Person-Test role pairs instantiating the pass fact type must be a subset of the set of Person-Test role pairs instantiating the took fact type. Here the subset constraint applies between whole associations. This is the only case with graphical support for subset constraints in UML. In such cases, UML displays “{subset}” besides a dotted arrow directed to the super-association, as shown.

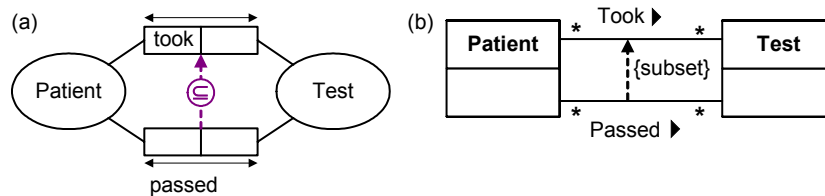


Figure 3 A pair-subset constraint in (a) ORM, and (b) UML.

This pair-subset constraint may be verbalized in relational style, in either of the following ways. The usual synonyms for keywords may also be used. As discussed in earlier articles, if the same object type plays more than one role in an association, either role names, or numeric subscripts appended to the object type name, may be used to distinguish the role players.

Each Patient who passed a Test also took that Test.
If a Patient passed a Test then that Patient took that Test.

This verbalization pattern may be extended in obvious ways to cater for other cases, where either role-pair may come from just part of a longer association, or is projected from a role path spanning multiple associations. As a simple example of the former, replace binary pass association by the ternary fact type Patient on Test obtained Result, and run the subset constraint from the first two roles of this fact type. In this case the constraint may be verbalized using an existential quantifier for the Result role, e.g.

Each Patient who on a Test obtained some Result also took that Test.
If a Patient on a Test obtained some Result then that Patient took that Test.

In UML, this ternary example could be handled with an OCL version of the constraint, or by objectifying the Patient took Test association as an association class, and then modeling the result fact type as an attribute or association.

As a simple example of a subset constraint involving a *projection from a role path*, consider the ORM model shown in Figure 4(a). The subset constraint means that each advisor who serves in a country must also speak at least one language that is used by that country. Here the source (subset) role pair for the constraint is the association Advisor serves in Country. The target (superset) role pair for the constraint is obtained by projecting on the first and last roles of the join path Advisor speaks Language that is used by Country. This path involves a *conceptual join* between the shaded roles played by Language.

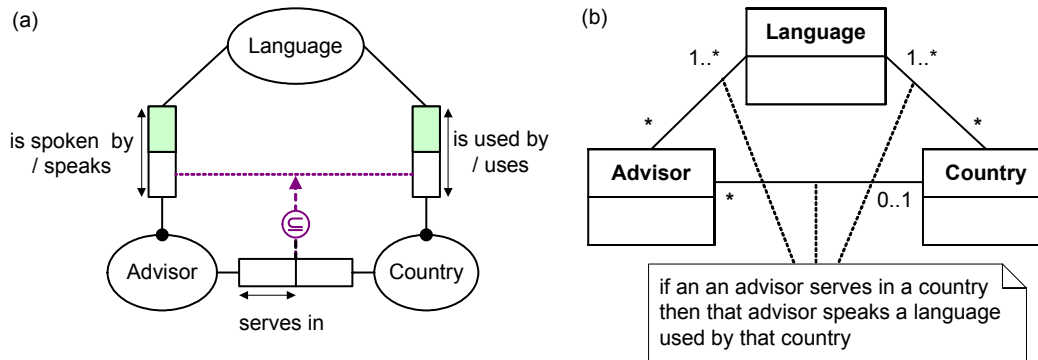


Figure 4 A subset constraint involving a join path expressed in (a) ORM and (b) UML.

The target Advisor-Country role-pair is indicated by a dotted line between the relevant roles. UML has no graphic notation for this kind of constraint, but the constraint may be noted either by an OCL expression or by an informal comment (as shown in Figure 4(b)). The join-subset constraint may be formally verbalized as follows. For further discussion of such join-constraints, see [2].

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

That completes our coverage of subset constraints and their verbalization. The next article considers equality and exclusion constraints.

References

1. Halpin, T. A. 2001, *Information Modeling and Relational Databases*, Morgan Kaufmann, San Francisco.
2. Halpin, T.A. 2002, 'Join Constraints', *Proc. Seventh CAiSE/IFIP-WG8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, eds. T. Halpin, J. Krogstie, K. Siau, Toronto, Canada, pp. 121-131, URL: <http://www.orm.net/pdf/JoinConstraints.pdf>.
3. Halpin, T. A. 2002, 'Metaschemas for ER, ORM and UML Data Models: A Comparison', *Journal of Database Management*, vol. 13, No. 2, pp. 20-29, Idea Publishing Group, Hershey PA, USA.
4. Halpin, T. A. 2003, 'Verbalizing Business Rules: Part 1', *Business Rules Journal*, Vol. 4, No. 4 (April 2003), URL: <http://www.BRCommunity.com/a2003/b138.html>.
5. Halpin, T. A. 2003, 'Verbalizing Business Rules: Part 2', *Business Rules Journal*, Vol. 4, No. 6 (June 2003), URL: <http://www.BRCommunity.com/a2003/b152.html>.
6. Halpin, T. A. 2003, 'Verbalizing Business Rules: Part 3', *Business Rules Journal*, Vol. 4, No. 8 (August 2003), URL: <http://www.BRCommunity.com/a2003/b163.html>.
7. Halpin, T. A. 2003, 'Verbalizing Business Rules: Part 4', *Business Rules Journal*, Vol. 4, No. 10 (October 2003), URL: <http://www.BRCommunity.com/a2003/b172.html>.
8. Halpin, T. A. 2004, 'Verbalizing Business Rules: Part 5', *Business Rules Journal*, Vol. 5, No. 2 (February 2004), URL: <http://www.BRCommunity.com/a2004/b179.html>.
9. Halpin, T. A. 2004, 'Verbalizing Business Rules: Part 6', *Business Rules Journal*, Vol. 5, No. 4 (April 2004), URL: <http://www.BRCommunity.com/a2004/b183.html>.
10. Halpin, T., Evans, K., Hallock, P. & MacLean, B. 2003, *Database Modeling with Microsoft Visio for Enterprise Architects*, Morgan Kaufmann, San Francisco.
11. Object Management Group 2003, *UML 2.0 Infrastructure*, URL: <http://www.omg.org/uml>.
12. Object Management Group 2003, *UML 2.0 Object Constraint Language*, URL: <http://www.omg.org/uml>.