Verbalizing Business Rules: Part 12

Terry Halpin
Northface University

Business rules should be validated by business domain experts, and hence specified in a language easily understood by business people. This is the twelfth in a series of articles on expressing business rules formally in a high-level, textual language. The first article [2] discussed criteria for a business rules language, and verbalization of simple uniqueness and mandatory constraints on binary associations. Article two [3] examined hyphen-binding, and verbalization of internal uniqueness constraints that span a whole association, or that apply to *n*-ary associations. Article three [4] covered verbalization of basic external uniqueness constraints. Article four [5] 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. Article five [6] 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. Article six [7] considered verbalization of value constraints. Article seven [8] examined verbalization of subset constraints. Article eight [9] discussed verbalization of equality constraints. Article nine [10] covered verbalization of exclusion constraints. Article ten [11] dealt with verbalization of internal frequency constraints on single roles. Article eleven [12] considered verbalization of multi-role frequency constraints and external frequency constraints. This article discusses verbalization of ring constraints.

Compatible Roles and Ring Constraints

Two roles are said to be *compatible* if and only if they are played either by the same object type, or by different object types that overlap. Figure 1 shows four cases of compatible roles, using the ORM graphic notation. For illustration purposes the compatible roles have been shaded in each case. In case (a) the roles comprise a binary predicate, and each is played by the same object type *A*. In case (b) the compatible roles form part of a ternary predicate, and each is played by the same object type *A*. In case (c) the roles are played by object types *B* and *C* that share a common supertype *A*. In the absence of an explicit or implicit exclusion constraint between the subtypes (see next article), the subtypes are assumed to overlap (i.e. they have some instances in common). In case (d) the compatible roles are played by a subtype *C* and its supertype *A*.

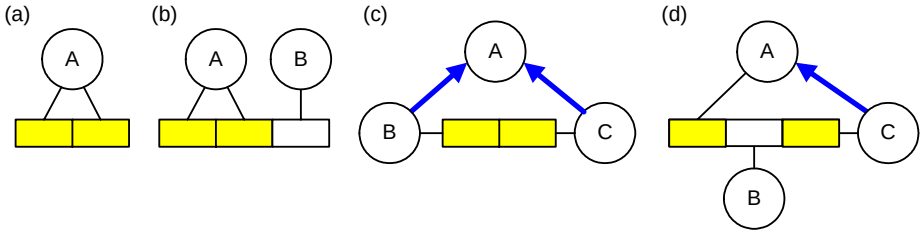

Figure 1 The shaded roles in each case are compatible.

In each case, a path from the object type *A* through the compatible role pair and back to *A* may be visualized as a circle or ring. A *ring constraint* may apply only to a pair of compatible roles. In principle, these roles might belong to different predicates, in which case the ring constraint is external. But in practice, ring constraints almost always apply to roles from the same predicate, and we confine our attention to such internal cases. *An internal ring constraint may apply only to a pair of compatible roles in the same predicate* (binary or longer).

To illustrate some ring constraints, consider the report extract shown in Table 1. The table is intended to list each country, as well as the countries that share a land border with it. For example, Australia has no bordering neighbors of this kind, whereas Belgium has four countries that border it. The full table lists every country in the left-hand column, but to conserve space only an extract is shown here (e.g. the rows for Austria, Czech Republic, Denmark, Italy, Poland, and Switzerland are among those omitted here).

Table 1 Extract of Country Borders.

Country	*Bordering Neighbors*
Australia	
Belgium	France, Germany, Luxembourg, The Netherlands
France	Belgium, Germany, Italy, Luxembourg, Switzerland
Germany	Austria, Belgium, Czech Republic, Denmark, France, Luxembourg, Poland, Switzerland, The Netherlands
Luxembourg	Belgium, France, Germany
Portugal	Spain
Spain	France, Portugal
The Netherlands	Belgium, Germany
…	…

An ORM schema for this example, as well as a sample population extract, is shown in Figure 2(a). A UML class diagram for this case is shown in Figure 2(b). The binary fact type Country borders Country is many:many, so has a spanning uniqueness constraint. In addition it has two ring constraints, depicted as "*ir*" (irreflexive) and "*sym*" (symmetric) following the ring symbol "○". A binary predicate *R* with compatible roles is said to be *irreflexive* if and only if no instance of its object type may bear the relationship to itself (i.e. *R* is irreflexive iff $\forall x \sim xRx$). No country may border itself, so we must not include any fact with the same country playing both roles (e.g. "Belgium borders Belgium" would violate the constraint).

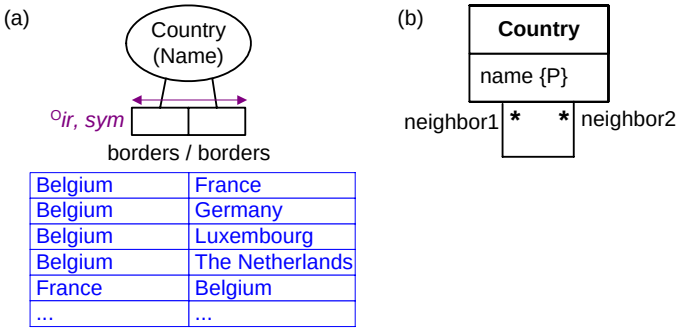

Figure 2 (a) ORM schema and sample population, and (b) UML class diagram for Table 1.

A binary predicate *R* with compatible roles is said to be *symmetric* if and only if for each relationship instance, the inverse relationship also holds, i.e. *R* is symmetric iff $\forall xy$ ($xRy \rightarrow yRx$). The borders relation is symmetric because if one country borders a second country, then the second country must border the first. For example, Belgium borders France, so France must border Belgium. Another symmetric fact type is Person is a sibling of Person. For symmetric binary associations, the forward and inverse predicate readings are the same, as shown ("borders"). Figure 2 includes an extract of the fact instances. In the full population, each pair of countries listed on one row appears in the reverse order on another row.

Notice that the irreflexive ring constraint is a *negative* constraint in that it forbids some facts (of the form *xRx*) from being entered. In contrast, the symmetric ring constraint is a *positive* constraint: given one fact (of the form *xRy*), the constraint requires the existence of another fact (*yRx*). All the other ring constraints we consider are negative in this sense.

Two other positive ring constraints (reflexive and transitive) are commonly discussed in logic and mathematics. A relation *R* is *reflexive* iff $\forall x\ xRx$ and is *transitive* iff $\forall xyz$ (xRy & $yRz \rightarrow xRz$). But reflexive and transitive constraints typically do not apply to base fact types in business domains, so we ignore them.

The names and ORM symbols for the other ring constraints that we do consider are: asymmetric (○*as*); antisymmetric (○*ans*); intransitive (○*it*); and acyclic (○*ac*). With an *asymmetric* relation, if a relationship holds then its inverse cannot hold, i.e. $\forall xy$ ($xRy \rightarrow \sim yRx$). For example, Person is taller than Person is asymmetric. With an *antisymmetric* relation, if a relationship holds between non-identical objects then its inverse cannot hold, i.e. $\forall xy$ [($x \neq y$ & xRy) $\rightarrow \sim yRx$)]. For example, Number is less than or equal to Number is antisymmetric but not asymmetric.

With an *intransitive* relation, if a first object bears the relationship to a second, and the second bears the relationship to a third, then the first cannot bear the relationship to the third, i.e. $\forall xyz\ (xRy\ \&\ yRz \rightarrow \sim xRz)$. For example, Person is older than Person is intransitive. With an *acyclic* relation, no cycles of any length are allowed (so no object can cycle back to itself through one or more applications of the relationship). For example, Person is older than Person is acyclic.

As the last example shows, some ring constraints imply other ring constraints (e.g. acyclic implies asymmetric). In such cases, the stronger constraint should be declared and the weaker (implied) constraint should be omitted. The following ring constraint implications are the most important to remember: asymmetric implies irreflexive; intransitive implies irreflexive; irreflexive and functional implies intransitive; acyclic implies asymmetric; exclusion implies asymmetric. For further discussion of these implication theorems, see [1, sec. 7.3].

As an aid to memory, Figure 3 shows an Euler diagram that I devised many years ago to visualize the relationships between the basic ring constraints, as well as what constraint combinations make sense. For example, the set of intransitive (^{O}it) relations is a subset of the irreflexive (^{O}ir) relations, so intransitive implies irreflexive. Symmetric (^{O}sym) may be paired with intransitive or irreflexive but no other ring constraint. Further discussion of this diagram may be found in [1, sec. 7.3].

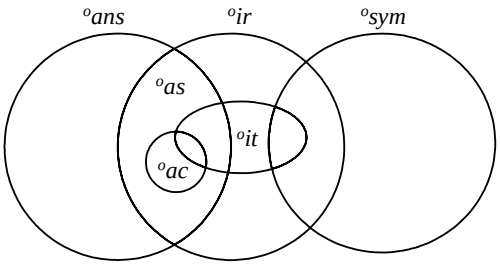

Figure 3 Relationships between ring constraints.

Verbalization of ring constraints

The *irreflexive* and *symmetric* ring constraints in Figure 2 may be respectively verbalized as shown below. In the symmetric constraint declaration, the object type variables Country$_1$ and Country$_2$ are understood to be universally quantified. This assumption may be made explicit by prepending "For each Country$_1$, Country$_2$:" to the formulation.

No Country borders itself. -- irreflexive
If Country$_1$ borders Country$_2$ then Country$_2$ borders Country$_1$. -- symmetric

If any ring constraint applies to two roles from a ternary or longer association, then an existential quantifier (e.g. "some", "a" or "an") precedes the names of the object types playing the other roles. For example, the irreflexive constraint on the first two roles of the ternary association in Figure 4 may be verbalized thus:

No Part contains itself in some Quantity. -- irreflexive

The binary containment relation is not just irreflexive, but acyclic. So we would normally assert an acyclic constraint here and omit the irreflexive constraint because it is implied by acyclicity.

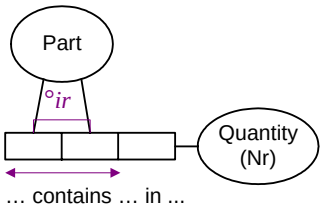

Figure 4 The sub-relation formed from the first two roles is irreflexive.

If the object type playing the roles subject to the ring constraint is declared personal, then "itself" may be replaced by "himself/herself". For example, an irreflexive constraint on Person is a parent of Person may be verbalized as "No Person is a parent of himself/herself". If the object type is declared to be both personal and male, then "itself" may be replaced by "himself". If the object type is declared to be both personal and female, then "itself" may be replaced by "herself". For example, if "Woman" is declared personal and female, example, then an irreflexive constraint on Woman is a parent of Woman may be verbalized as "No Woman is a parent of herself".

As an example of an *asymmetric* constraint, consider the reporting relationship depicted in Figure 5(a). In practice, the reporting relationship is typically acyclic, which implies that it is asymmetric. But for discussion purposes, suppose we merely wish to declare asymmetry for this example. The asymmetric constraint may be verbalized thus:

If $Employee_1$ reports to $Employee_2$
then it is impossible that $Employee_2$ reports to $Employee_1$.

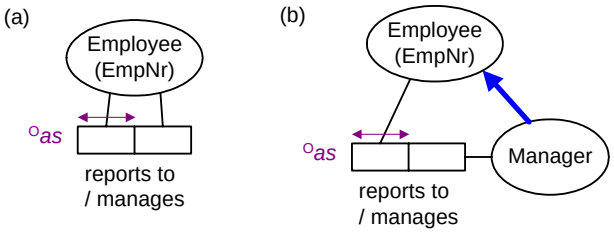

Figure 5 The reporting relation is asymmetric.

If we wish to treat this as a *deontic* constraint (an obligation that might be violated) rather than as an alethic constraint (a necessity that cannot be violated), then we replace "impossible" by "forbidden" thus:

If $Employee_1$ reports to $Employee_2$ then it is forbidden that $Employee_2$ reports to $Employee_1$.

As a variation involving subtyping, the ring asymmetric constraint in Figure 5(b) may be verbalized thus:

If an Employee reports to a Manager then it is impossible that that Manager reports to that Employee.

Since the reporting relation is functional (the first role has a simple uniqueness constraint), it is implied to be intransitive. So an intransitive constraint need not be declared for this case. The intransitivity implication holds regardless of which role is functional. For example, the association Person is the father of Person is 1:many, and hence is implied to be intransitive. If you have trouble seeing this, try it out with a sample population.

As an example of an *intransitive* ring constraint that is not implied, consider the parenthood association shown in Figure 6. This association is many:many, so is not functional, and hence intransitivity is not implied. Because of the regrettable possibility of incest, the intransitive constraint is deontic (indicated here by a different color) rather than alethic. The intransitive constraint may be verbalized thus:

If $Person_1$ is a parent of $Person_2$ and $Person_2$ is a parent of $Person_3$
then it is forbidden that $Person_1$ is a parent of $Person_3$.

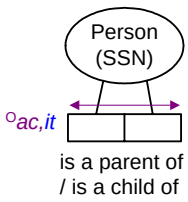

Figure 6 The parenthood association is acyclic and deontically intransitive.

If we exclude reincarnation from consideration, the parenthood association is *acyclic*, as indicated in Figure 6. This acyclic constraint means that nobody can be one of their own descendants (or one of their own ancestors). The constraint may be verbalized thus:

No Person may cycle back to itself via a chain of one or more instances of the association Person is a parent of Person.

If the object type playing the roles constrained by the acyclic constraint plays other roles in the association, the verbalization must indicate the constrained roles (e.g. by subscripting the relevant object type variables in the association).

Antisymmetric ring constraints are rare in business domains. As a somewhat contrived example, consider the inclusion relationship depicted in Figure 7. If we treat "includes" as reflexive (so that each group includes itself), then this association is antisymmetric, but not asymmetric. It may then be verbalized as follows:

If $Group_1$ includes $Group_2$ and $Group_1$ is not identical to $Group_2$
then it is impossible that $Group_2$ includes $Group_1$.

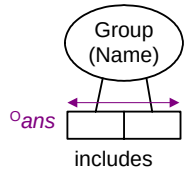

Figure 7 The group inclusion relationship is antisymmetric.

Of course, if we use "includes" in an irreflexive sense (so that no group includes itself), the inclusion relationship is asymmetric (which implies that it is antisymmetric). Indeed, it would then also be acyclic (which implies asymmetric).

That completes our coverage of ring constraints. The next article considers verbalization of subtype constraints.

References

1. Halpin, T. A. 2001, *Information Modeling and Relational Databases*, Morgan Kaufmann, San Francisco.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.
8. Halpin, T. A. 2004, 'Verbalizing Business Rules: Part 7', *Business Rules Journal*, Vol. 5, No. 7 (July, 2004), URL: http://www.BRCommunity.com/a2004/b198.html.
9. Halpin, T. A. 2004, 'Verbalizing Business Rules: Part 8', *Business Rules Journal*, Vol. 5, No. 9 (September, 2004), URL: http://www.BRCommunity.com/a2004/b205.html.
10. Halpin, T. A. 2004, 'Verbalizing Business Rules: Part 9', *Business Rules Journal*, Vol. 5, No. 12 (December, 2004), URL: http://www.BRCommunity.com/a2004/b215.html.
11. Halpin, T. A. 2005, 'Verbalizing Business Rules: Part 10', *Business Rules Journal*, Vol. 6, No. 4 (April, 2005), URL: http://www.BRCommunity.com/a2005/b229.html.
12. Halpin, T. A. 2005, 'Verbalizing Business Rules: Part 11', *Business Rules Journal*, Vol. 6, No. 6 (June 2005), URL: http://www.BRCommunity.com/a2005/b238.html.
13. Halpin, T., Evans, K., Hallock, P. & MacLean, B. 2003, *Database Modeling with Microsoft Visio for Enterprise Architects*, Morgan Kaufmann, San Francisco.
14. Object Management Group 2003, *UML 2.0 Infrastructure*, URL: http://www.omg.org/uml.
15. Object Management Group 2003, *UML 2.0 Object Constraint Language*, URL: http://www.omg.org/uml.