Date: August 2016

Semantics of Business Vocabulary and Business Rules (SBVR), v1.4

Annex L - ORM Examples Related to the Logical Foundations for SBVR

OMG Document Number: dtc/2016-08-30

Note: This SBVR Annex is published as a separate document solely for convenience and ease of use. The fact that it is published as a separate SBVR specification document makes no change to its status as part of the SBVR specification, or the way in which it can be updated under OMG Policies and Procedures.

Annex L - ORM Examples Related to the Logical Foundations for SBVR

(informative)

L.1 Introduction

This annex provides some detailed examples to illustrate how foundational concepts described in sub clause 24.2.1 can be captured in an existing logic-based approach. The examples use Object-Role Modeling (ORM), which has a well-defined mapping to formal logic [Halp1989]. A basic introduction to ORM may be found in [Halp2000] and a detailed treatment in [Halp2001]. ORM takes a fact-based approach to modeling business scenarios that is compatible with the SBVR approach.

L.2 Simple Database Example

Figure L.1 shows an ORM schema for the simple Employee/Car database example. In ORM, *objects* are either *entities* (non-lexical objects that are identified by definite descriptions, and that typically change state) or *values* (lexical constants that identify themselves, such as character strings). In ORM 2 (the latest version of ORM, used here), entity types and value types are depicted as named, soft rectangles with solid or dotted lines respectively (previous versions of ORM used ellipses instead of soft rectangles). Logical *predicates* are depicted as named sequences of role boxes, where each *role* is a part played in the relationship. For binary fact types, if forward and inverse predicate readings are displayed on the same side of the role boxes, they are separated by a slash "/". By default, predicates are read left-to-right and top-to-bottom.

A large dot on a role connector indicates that the attached *role is mandatory* (i.e., for each state of the fact model, each instance in the population of the object type must play that role). The object type's population in the fact model is not necessarily the same as the real world population in that state, and is typically far smaller than the extension of the object type (which covers all possible states). For example, each employee has an employee name, but it is optional whether an employee drives a car.

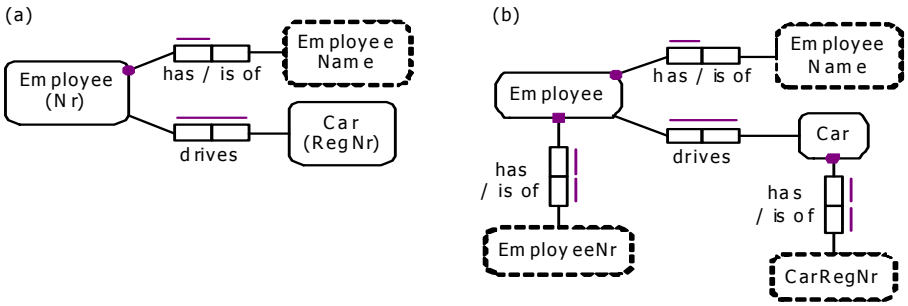


Figure L.1 - ORM schema for the simple Employee/Car database example

A bar beside a role box depicts a *uniqueness constraint*, indicating that for each state of the fact model each object that instantiates that role does so only once. For example, each employee has at most one employee name. A bar that spans two or more roles depicts a uniqueness constraint over that role combination, indicating that for each state of the fact model each object sequence that instantiates that role sequence does so only once. For example, the fact type Employee drives Car is many:many, and in each state any instance of this fact type appears at most once.

Figure L.1(b) displays simple injective (mandatory, 1:1 into) reference schemes explicitly as binary relationships. Employees are referenced by their employee numbers, and cars by their registration numbers. Figure L.1(a) displays these reference schemes compactly as parenthesized reference modes.

L.3 Open/Closed World

Consider the populated unary fact type in Figure L.2(a). For simplicity, we omit reference schemes, and assume people may be identified by their first names. We know that Fred smokes. If we use open world semantics, then it is unknown whether Sue or Tom smoke. If the ORM schema is mapped to a UML class, then the open world interpretation leads to an optional isSmoker attribute with only one possible value ('Y' for yes), as shown in Figure L.2(b). If we apply closed world semantics, then the absence of facts that Sue or Tom smoke entails that they don't smoke; this leads to a mandatory, Boolean isSmoker attribute, as shown in Figure L.2(c).

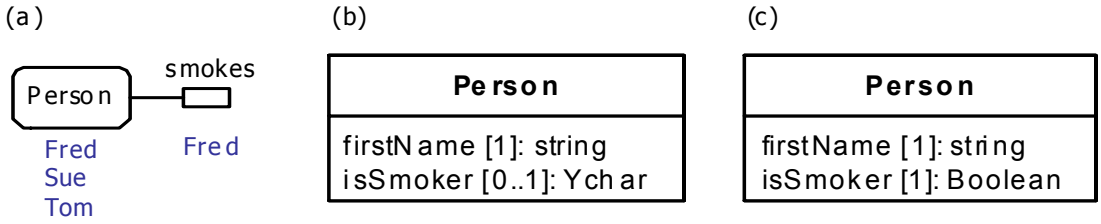


Figure L.2 - An ORM model (a), and UML classes based on (b) open world and (c) closed world semantics

Currently most ORM tools adopt the closed world assumption for unaries. However for the next generation of ORM tools that are designed to interoperate with SBVR tools, it is anticipated that unaries will be treated as open by default.

For many fact types in a business domain, especially those without functional roles, it is impractical to include all the negative instances as base facts. For example, for the fact type Employee drives Car, there might be many thousands of cars, so one would normally not explicitly include negated facts such as Employee 1 does **not** drive Car 'AAA246'. In some cases however, especially with functional roles or when the population is small, it is practical to include negated facts as base facts.

Figure L.3 shows two ways to model a business domain where for each person in the population of the domain it is known whether that person smokes or not. In each case, negated facts are explicitly treated as base facts, and the predicates are given open world semantics. Semi-closure is implied because of the constraints. In Figure L.3(a) the xor constraint (circled mandatory dot overlaid by 'X" for exclusion) declares that each person referenced in the fact model population plays exactly one of the two roles (smoking or not smoking). In Figure L.3(b) the mandatory, uniqueness and value constraints collectively ensure the same thing. When either of the ORM schemas is mapped to a UML class, a mandatory Boolean isSmoker attribute results, as shown in Figure L.3(c).

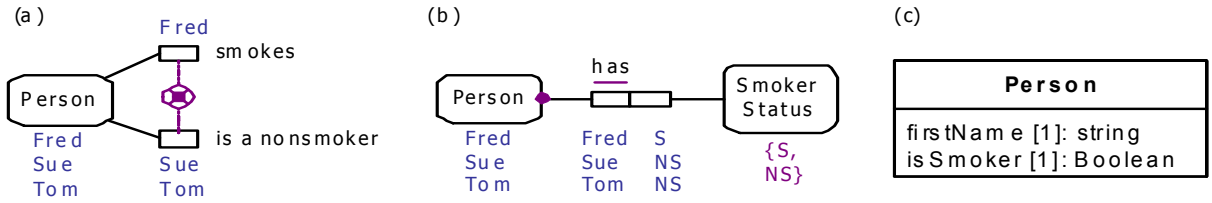


Figure L.3 - Open world semantics plus negated facts and constraints that ensure semi-closure

Now consider a business domain where we know that Fred smokes, and that Sue doesn't smoke, but are unsure whether Tom smokes. To model this at all, we need open world semantics. Figure L.4 shows three ways to model this in ORM, as well as the equivalent UML class. Figure L.4(a) uses an exclusion constraint, Figure L.4(b) uses an optional binary, and Figure L.4(c) uses a mandatory binary and a special value (here shown as "?") to indicate that the smoking status is unknown. We treat this special value like any other value, using 2-valued logic, rather than adopt a generic null based on 3-valued logic (as in SQL). The equivalent UML class notation is shown in Figure L.4(d).

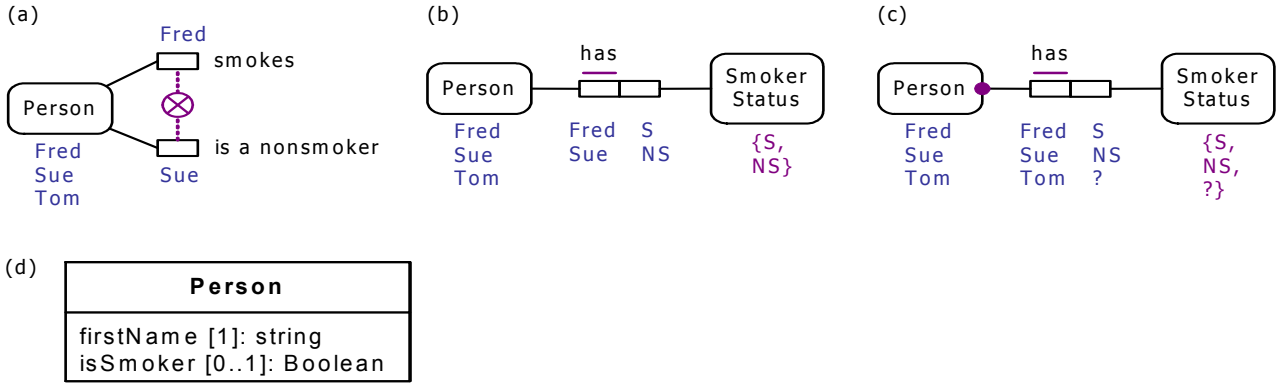


Figure L.4 - We may indicate whether a person smokes or not, or that this is unknown.

L.4 Deontic Constraints

In the ORM schema shown in Figure L.5, the fact type Person is a husband of Person is declared to be many to many, as shown by the alethic, spanning uniqueness constraint over the top of the predicate. In addition a deontic uniqueness constraint has been added (depicted by a bar starting with an "o" for "obligatory") to each role to indicate that the fact type *ought* to be 1:1. The leftmost deontic constraint verbalizes as: **It is obligatory that each** Person is a husband of **at most one** Person. The other deontic constraint (each wife should have at most one husband) may be handled in a similar way.

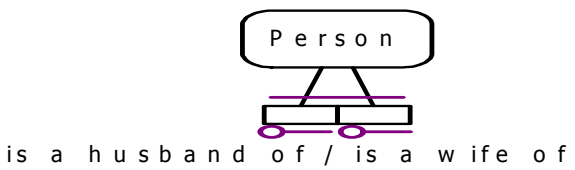
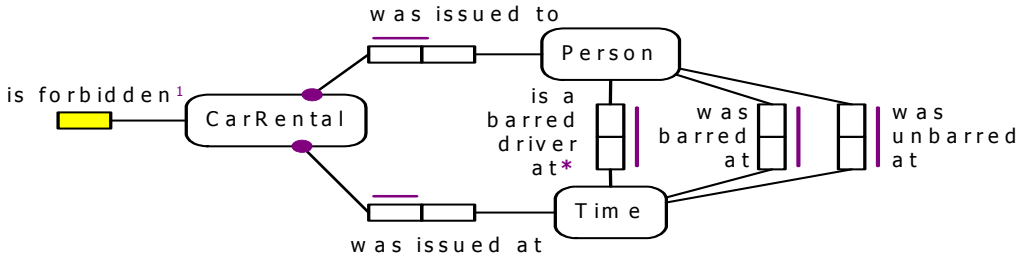


Figure L.5 - Deontic constraints obligate the marriage relationship to be 1:1.

The deontic constraint “Car rentals ought not be issued to people who are barred drivers at the time the rental was issued” may be captured by the textual constraint on the domain fact type CarRental is forbidden, as shown in the ORM schema in Figure L.6. The fact type Person is a barred driver at Time is derived from other base fact types (Person was barred at Time, Person was unbarred at Time) using the derivation rule shown.

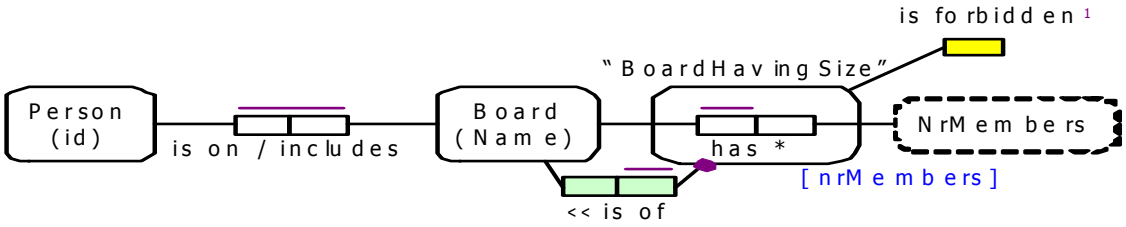



```
*Person is a barred driver at Time1 iff
   Person was barred at a Time2 <= Time1 and
   Person was not unbarred at a Time3 between Time2 and Time1

1 CarRental is forbidden if
          CarRental was issued at Time and
          CarRental was issued to Person and
          Person is a barred driver at Time
```

Figure L.6 - Specifying a deontic constraint forbidding rentals to barred drivers using a domain level predicate

The deontic constraint “It is forbidden that more than three people are on the EU-Rent Board” is captured by the textual constraint on the derived fact type BoardHavingSize is forbidden in the ORM schema shown in Figure L.7. The derivation rule is stated in attribute style, but its underlying relational style is used in invoking the derivation rule within the body of the deontic constraint.

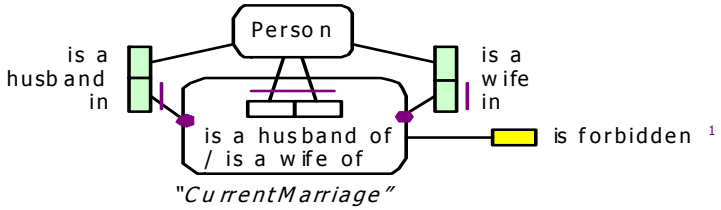



```
* nrMembers of Board = count each Person who is on Board

1 BoardHavingSize is forbidden if
          BoardHavingSize is of a Board
                    that has BoardName 'EU-Rent Board'
                     and has NrMembers > 3
```

Figure L.7 - Specifying a deontic constraint on the size of the EU-Rent board using a domain level predicate

The deontic constraints that require each person to have at most one spouse may be formulated as textual constraints on the fact type CurrentMarriage is forbidden, as shown in Figure L.8.

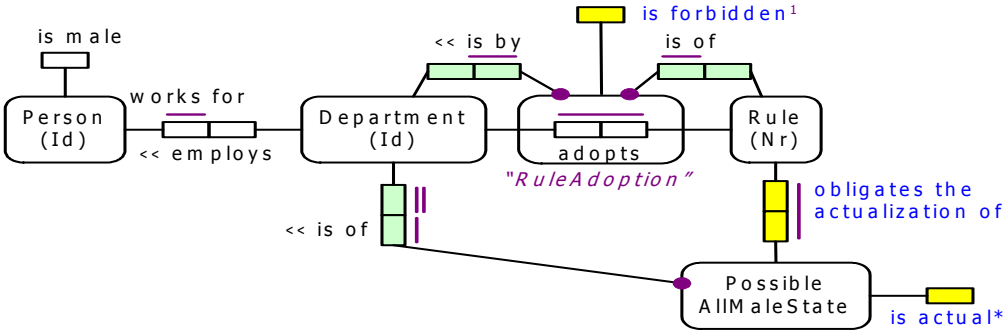


[1] CurrentMarriage is forbidden **if**
a Person$_1$ **who** is a husband in CurrentMarriage
is a husband of **more than one** Person$_2$.
CurrentMarriage is forbidden **if**
a Person$_1$ **who** is a wife in CurrentMarriage
is a wife of **more than one** Person$_2$.

Figure L.8 - An alternative way to capture the deontic constraints in Figure 5

The ORM schema in Figure L.9 relates to the following deontic constraint: “It is not permitted that some department adopts a rule that says it is obligatory that each employee of that department is male.” This example includes the mention (rather than use) of an open proposition in the scope of an embedded deontic operator. The schema uses the special predicates “obligates the actualization of” and “is actual,” as well as an object type “PossibleAllMaleState” which includes all conceivable all-male-states of departments, whether actual or not.

The formalization of the deontic constraint works, because the relevant instance of PossibleAllMaleState exists, regardless of whether or not the relevant depart actually is all male. The “obligates the actualization of” and “is actual” predicates embed a lot of semantics, which is left implicit. While the connection between these predicates is left informal, the derivation rule for PossibleAllMaleState is actual provides enough semantics to enable human readers to understand the intent.



* PossibleAllMaleState is actual **iff**
PossibleAllMaleState is of **a** Department **and**
each Person **who** works for **that** Department is male

[1] RuleAdoption is forbidden **if**
RuleAdoption is by **a** Department
and is of **a** Rule
that obligates the actualization of **a** PossibleAllMaleState
that is of **the same** Department

* $\forall x$:PossibleAllMaleState
[x is actual $\equiv \exists y$:Department (x is of y & $\forall z$:Person (z works for y $\supset$ z is male))]

[1] $\forall x$:RuleAdoption
[$\exists y$:Department $\exists z$:Rule $\exists w$:PossibleAllMaleState
(x is by y & x is of z & z obligates the actualization of w & w is of y)
$\supset$ x is forbidden]

aliter:

[1] $\forall x$:RuleAdoption $\forall y$:Department $\forall z$:Rule $\forall w$:PossibleAllMaleState
[(x is by y & x is of z & z obligates the actualization of w & w is of y) $\supset$ x is forbidden]

Figure L.9 - A complex case involving embedded mention of propositions