ORM 2 Graphical Notation

Technical Report ORM2-01, September 2005

Terry Halpin

www.neumont.edu

Contents

ORM 2 Graphical Notation

Terry Halpin

Neumont University
terry@neumont.edu

Abstract: Object-role Modeling (ORM) is a fact-oriented modeling approach for expressing, transforming and querying information at a conceptual level. Unlike Entity-Relationship modeling and Unified Modeling Language class diagramming, fact-oriented modeling is attribute-free, treating all elementary facts as relationships expressed in natural language sentences. For information modeling, ORM graphical notations are typically far more expressive than other notations, and ORM's attribute-free nature promotes semantic stability and facilitates natural verbalization. Based on industrial experience that identified ways to improve current ORM languages and associated tools, a project is under way to provide open-source tool support for a second generation ORM (ORM 2), that has significant advances over current ORM technology. This paper provides an overview of, and motivation for, the enhancements introduced by the ORM 2 graphical notation.

1 Introduction

Introduced in the mid 1970s, *Object-Role Modeling* (*ORM*) facilitates modeling, transforming, and querying business information in terms of the underlying *facts* of interest, where all facts and rules may be verbalized in language easily understood by non-technical users of those business domains. Unlike Entity-Relationship (ER) modeling [6] and Unified Modeling Language (UML) class diagramming [31, 32, 33], ORM treats all elementary facts as relationships (unary, binary, ternary, etc.), thus regarding decisions for grouping facts into structures (e.g. attribute-based entity types, classes, relation schemes, XML schemas) as implementation concerns irrelevant to business semantics. By avoiding attributes in the base model, ORM enhances semantic stability and populatability, and facilitates natural verbalization. For information modeling, fact-oriented graphical notations are far more expressive than ER and UML graphical notations. Fact-oriented textual languages are based on formal subsets of native languages, so are easier to understand by business people than technical languages like the Object Constraint Language (OCL) [35]. Fact-oriented modeling includes procedures for mapping to attribute-based structures, such as those of ER or UML.

The fact-oriented modeling approach comprises a family of closely related dialects. Some use the generic term "Object-Role Modeling" (ORM) [15], while others use different names such as Natural language Information Analysis Method (NIAM) [12, 36], and Fully-Communication Oriented Information Modeling (FCO-IM) [1, 2]. ORM languages include RIDL [30], LISA-D [26, 27] and FORML [15]. Though notationally different, the Object-oriented Systems Model (OSM) [11] is a close relative to ORM, with its attribute-free approach. Commercial ORM tools include the ORM solution within Microsoft's Visio for Enterprise Architects [23], and the FCO-IM tool CaseTalk [5]. Free ORM tools include VisioModeler [34], Infagon [28], and academic prototypes such as Dogma Modeler [10], an ORM-based tool for specifying ontologies. An introduction to ORM is included in [14], and a detailed treatment in [15].

Building on industrial experience that identified ways to improve current ORM languages and tools, our project aims to provide tool support for a second generation ORM (called *ORM 2*), that has significant advances over current ORM technology in functionality and usability. This paper overviews and motivates enhancements introduced by the ORM 2 graphical notation, and discusses a tool under development to support it. The initial development team comprised of faculty and students at Neumont University is being expanded to include external collaborators from industry and academia. The current implementation is coded in C# as a free, open-source plug-in to Microsoft Visual Studio .NET, using the new Microsoft Designer Framework Software Development Kit for building domain specific languages.

The rest of this report is structured as follows. Section 2 focuses on improvements to the ORM graphical notation. Section 3 briefly discusses enhancements to the ORM textual notation. Section 4 summarizes the main results, suggests topics for further research, and lists references. An appendix includes sample schemas to illustrate the impact of changing to the new notation.

2 The ORM 2 Graphical Notation

This technical report is concerned almost exclusively with the ORM 2 graphical notation. Later technical reports will discuss other aspects of ORM 2 tool support. We begin by summarizing the main objectives for the new ORM notation, and then examine specific notational areas in turn, including reasons for changes from the previous notation.

Most of the ORM 2 figures in this report were drawn using an ORM 2 Visio stencil that our team created. Currently, this stencil may be used for drawing purposes only. In contrast, the ORM 2 modeling tool is intended to support automatic transformation between graphical and textual representations, as well as transformation to/from other schemas (e.g. relational, object, and XML schemas), and code generation (e.g. to DDL or program code). The tool is also intended to front-end other modeling tools (e.g. one might enter and transform an ORM schema to a relational schema for export to another database design tool). As schemas developed in the tool are fully exposed as XML, there is significant scope for inter-operability.

2.1 Main Objectives

While being far more expressive graphically than UML or industrial ER for static data models, ORM models tend to consume more space because of their attribute-free nature. This diagram size problem may be ameliorated by providing attribute-views on demand, and/or by redesigning the ORM graphic notation to be more compact. The first solution includes displaying "minor fact types" as attributes on an ORM diagram, as well as automatically generating attribute-based schemas for implementation targets (e.g. relational schemas, object schemas, and XML schemas). We postpone work on this first solution for now, instead focusing on a new ORM graphical notation that is more compact and hopefully more acceptable in other aspects. The main objectives for the ORM 2 graphical notation are:

- More compact display of ORM models without compromising clarity
- Improved internationalization (e.g. avoid English language symbols)
- Notation changes that are reasonably acceptable to an identified short-list of key ORM users
- Simplified drawing rules to reduce the effort to create a graphical editor
- Full support of textual annotations (e.g. footnoting of textual rules)
- Extended use of views for selectively displaying/suppressing detail
- Support for new features (e.g. role path disambiguation, modalities, open world aspects)

For convenience, we use the term "ORM1 notation" to refer to the ORM notation currently supported by Microsoft's Visio-based ORM solution, which we call the "Visio ORM1 tool". We use the term "ORM2 notation" for the new notation supported by our ORM 2 tool. The ORM2 notation typically yields schema diagrams that consume about 65% the size of an equivalent diagram in the ORM1 notation. See the appendix for a representative schema displayed in both the old and new notations. All English-specific symbols in the ORM1 notation have been replaced by language-neutral symbols to improve localization in different language communities. A survey was issued to eighteen ORM experts regarding the proposed ORM2 notation, and each change introduced by the new notation was acceptable to a majority of the responders.

2.2 Object Type Shapes

In the ORM1 notation, object type (entity type and value type) shapes are depicted by named ellipses. While ellipses are faster for users to draw manually on paper, they consume more usable space than rectangles (hard or soft), especially when long names are involved. For ORM 2, the default shape for object types is a *soft rectangle* (rectangle with rounded corners). Besides providing a more compact container for the enclosed text, this is consistent with the current notation for nested object types. The shape auto-sizes to provide appropriate white space around the text, while maintaining a convenient minimum size, and users may spread text over multiple lines (as in the Visio ORM1 tool). Text settings and controls follow existing conventions (i.e. text is displayed in a user-definable default style, individual text elements may be user-selected for alternate styles, text may be left/center/right justified, etc.). To make this notation change more acceptable, we allow the use of an *ellipse* or a *hard rectangle* as an alternative shape for object types, as determined by a configuration option. Figure 1 shows some examples.

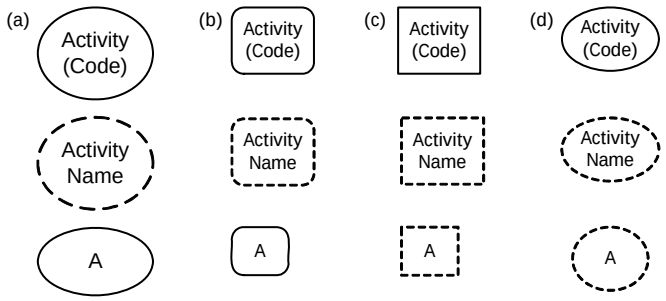


Figure 1 Object type shapes in (a) old notation and new notations: (b) default; (c) and (d) alternatives.

If the ellipse option is chosen, the white space is reduced compared to the old notation. Object type shape examples in the rest of this document all use the default shape (soft rectangle). Of the 18 experts surveyed, 12 preferred the soft rectangle, 5 preferred the ellipse shape, and one preferred the hard rectangle.

2.3 Shapes and Readings for Predicates and Roles

To save space, the *size of the role boxes is significantly reduced*. A line connecting a role box to an object type shape is always directed from the mid-point of an outer edge of the role box to the *center* of the object type shape (unlike the Visio ORM1 tool, which connects to the closest connection point on the object type shape). When using the ORM 2 stencil to connect a role to the object type, drag the role connector line onto the object type shape until the connection is confirmed by a red rectangle display around the object type. As with the current solution, predicate readings may be user-positioned beside the predicate shape. Figure 2 shows a simple example. Although there is no longer room to place a predicate reading inside a role box, as in Figure 2(a), this option was rarely used anyway. By default, all text is in *7 point Tahoma* (the Visio ORM1 tool uses 8 point Arial).

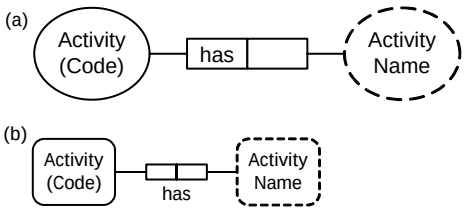


Figure 2 The old role box size (a) is significantly reduced (b) with text in 7 point Tahoma.

The reduced role box size still allows space for multiple single-role set-comparison constraints to be conveniently aligned, as shown in Figure 3. The old notation in Figure 3(a) is roomy enough to allow an additional set-comparison constraint between the role-pairs, without moving the other constraints. The new notation would require moving one of the constraints to make room for the third constraint, but since such cases are rare we believe this inconvenience to be minor and acceptable. The size of the constraint bubbles is slightly reduced in the new notation.

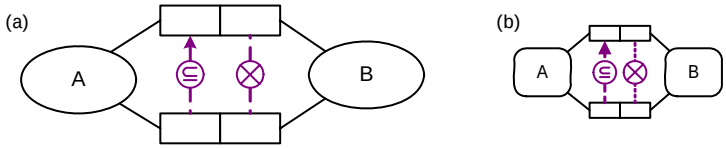


Figure 3 An example with set-comparison constraints in the old notation (a) and new notation (b).

As with the ORM1 notation for binary fact types, a forward predicate reading is read left-to-right or top-to-bottom, and an inverse predicate reading (pre-pended by "« ") is read right-to-left or bottom-to-top. For a binary fact type, *forward and inverse predicate readings* may be displayed together, separated by a slash without the "« " (as in the Visio ORM1 tool), or separately *on either side of the predicate shape* with the inverse reading pre-pended by "« ". At some future stage, we might replace the "«" symbol by an arrowhead (e.g. "◀"). The display of any predicate reading may be toggled on/off. Multi-line reading displays are allowed. Figure 4 shows some of the possibilities.

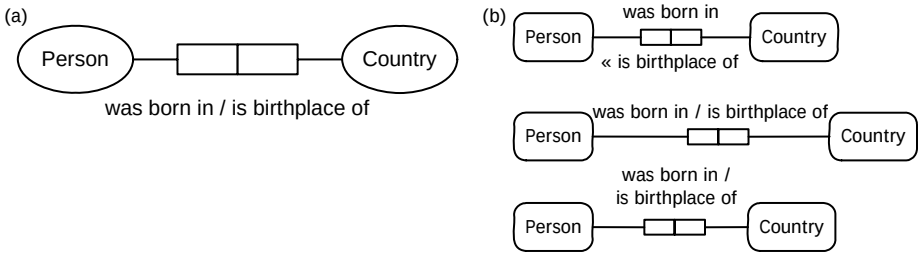


Figure 4 Forward and inverse predicate reading display in (a) ORM and (b) ORM 2.

The current ORM 2 stencil stores only one reading with the predicate shape (for now, other readings must be entered in a text box). The ORM 2 modeling tool, however, is much more flexible. For a fact type with *n* roles ($n > 0$), ORM 2 allows *predicate readings* for all possible ($n!$) permutations of role orderings. For each such role ordering, one or more *alias readings* may be supplied (e.g. "is employed by" as an alias of "works for"). Query navigation in relational style from any role of an *n*-ary fact type is enabled by just *n* predicate readings (one starting at each role), but industrial modelers requested this additional flexibility. For non-binary fact types, at most one predicate reading is displayed on the diagram. ORM 2 supports *fact type names* (as distinct from fact type readings) for the fact type (e.g. "Birth" for Person was born on Date), though this is not normally displayed on the diagram. One use of fact type names is to generate a suitable target name for fact types that map to a single table or class. Multi-line fact type names are allowed.

The *display of role names* in square brackets may be toggled on/off by the user (e.g. using layers). Multi-line role names are allowed. The display toggle may be set globally or on an individual role basis. Although each fact type has at least one predicate reading, the display of predicate readings may be suppressed (e.g. to focus on role names). By default, role names are displayed in a different color (e.g. indigo). Role names may be positioned individually where the user desires. Figure 5 shows some possibilities. The ORM 2 stencil does not yet provide special support for role names, but these may be entered in text boxes.

Figure 5 Role names may be displayed in square brackets.

2.4 Objectified Associations

Apart from a reduction in size, the soft-rectangle notation for objectified associations is retained in ORM 2. Figure 6 shows a simple example. Although the shape is similar, the formal semantics for objectification have been updated for ORM 2, so the result of objectifying a binary or longer relationship type may now be viewed as an entity type that has a composite reference scheme whose reference projection bears an equality constraint to the fact type being objectified [22]. When using the ORM 2 stencil to connect a role to the nested object type, Ctrl-drag the role connector line onto the nested object type shape until the connection is confirmed by a red rectangle display around the object type.

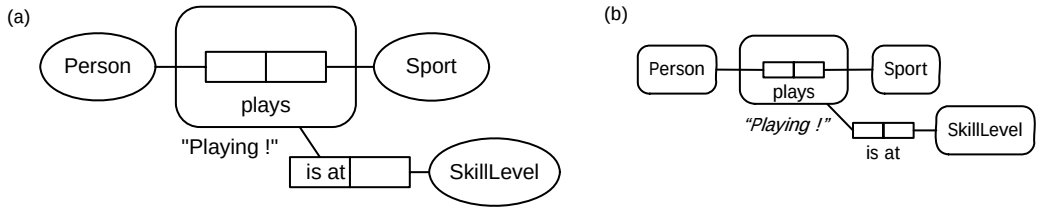


Figure 6 Nesting in (a) ORM 1 and (b) ORM 2.

Moreover, *nesting of unary predicates* is now allowed (Figure 7 gives an example), as well as *nesting of predicates with non-spanning uniqueness constraints*, in accordance with new formal semantics and guidelines for objectification for ORM 2 [16, 22].

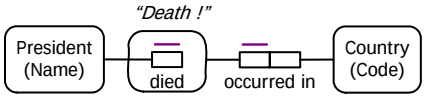


Figure 7 ORM2 allows objectification of unary fact types.

2.5 Internal Uniqueness Constraints

Internal uniqueness constraints in ORM 1 are indicated by arrow-tipped lines. ORM 2 replaces these by *simple lines*. Simple lines are faster to draw manually, and intuitively correspond to the common practice of underlining keys. The line for an individual role box is shorter than the box length to avoid ambiguity in interpretation. Figure 8 shows some examples (the constraints may also be displayed below the predicate). In the current ORM notation, a primary uniqueness constraint is marked "P". In ORM 2, a *preferred* uniqueness constraint is indicated by a *double line* (intuitively corresponding to one common practice of doubly underlining primary keys when alternate keys exist)—see the bottom example. In ORM 2 the notion of preferred uniqueness is conceptual, corresponding to a business decision to prefer a particular identification scheme. By default, all ORM 2 constraints are displayed in violet color.

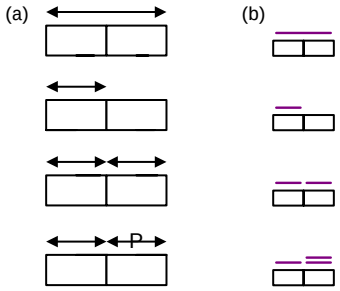


Figure 8 Internal uniqueness constraints in (a) ORM and (b) ORM 2.

In the case of an internal uniqueness constraint that spans non-contiguous roles, a *dashed line* bridges the gap across the inner role(s) that are excluded from the constraint. If the association is elementary, such a case may arise only if the association is ternary or higher. For example, the upper constraint in Figure 9 spans the first and last roles. Of the 18 experts surveyed, 17 preferred the new internal constraint notation.

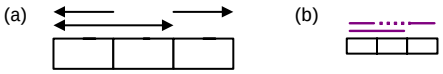


Figure 9 A uniqueness constraint over non-contiguous roles in (a) ORM and (b) ORM 2.

2.6 External Uniqueness Constraints

In the ORM1 tool, *external uniqueness* is denoted by a circled "U" (for unique), modified to a circled "P" (for "primary") if chosen as the primary reference scheme of an entity type. This notation is biased towards the English language, and has no resemblance to the notation for internal uniqueness. For internationalization and consistency, the ORM 2 notation uses a *circled underline* for external uniqueness constraints, and a *circled double underline* if the constraint provides the preferred identification scheme (see Figure 10). This is consistent with the new internal uniqueness constraint notation and the horizontal notation for relational schemas [15]. Of the 18 experts surveyed, 14 preferred this new constraint notation.

Figure 10 External uniqueness constraints in (a) ORM and (b) ORM 2.

2.7 Mandatory Role Constraints

The ORM1 tool indicates *simple mandatory constraints* by a solid dot either (a) at the intersection of an entity type shape and the line connecting it to a role, or (b) at the role end. Option (b) is needed to avoid ambiguity when an object type plays many mandatory roles whose connections to the object type are too close to distinguish which role the dot applies to. Currently, *disjunctive mandatory* (*inclusive-or*) *constraints* are depicted by placing the solid dot in a circle connected by dotted lines to the roles it applies to. ORM 2 retains this notation, except that the solid dot is consistently colored *violet* and a global *configuration option* determines the default placement of simple mandatory dots at the role or object type end. Users may override this global setting on an individual role basis. Figure 11 shows some simple examples.

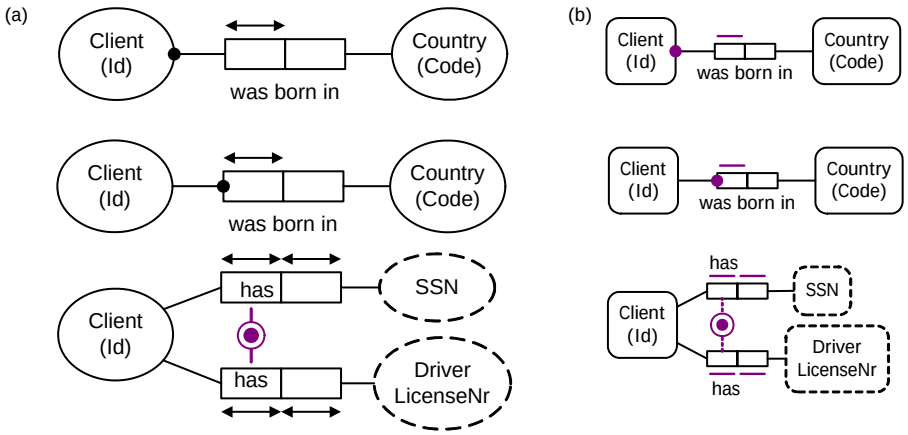


Figure 11 Mandatory constraints in (a) ORM and (b) ORM 2.

2.8 Set-comparison and Exclusive-Or Constraints

Subset, exclusion, and equality constraints will continue to be denoted by a circle containing ⊆, ×, = respectively, connected to the associated roles with dashed lines (as in the Visio ORM1 tool), except that the ORM 2 shapes are slightly smaller, with refined symbols. In addition, ORM 2 will support the *n-ary version of the equality constraint*. *Exclusive-or* constraints will continue to be denoted by combining the circled × with the circled dot, and users will have the option of displaying the two component constraints overlaid or separately (as in ORM 1). Figure 12 shows the basic shapes.

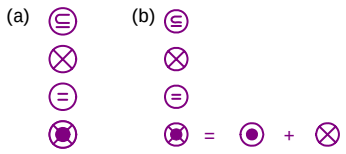


Figure 12 Set-comparison and Xor constraints in (a) ORM and (b) ORM 2.

2.9 Frequency and Value Constraints

Frequency constraints will be denoted as in the Visio ORM1 tool, except that single symbols (≤, ≥) replace double symbols (<=, >=), for example 2, ≥3, 2..5. *Value constraints* are denoted as in ORM 1, except that many values may be displayed on a single line (e.g. {'M', 'F'}, {'a',..'z'}), and *open ranges* are supported (e.g. "> 0" for PositiveReal).

ORM 2 allows value constraints to apply to *roles*, not just object types. For example, the metamodel fragment in Figure 13 includes value constraints on the minimumMultiplicity and maximumMultiplicity roles. Of the 18 experts surveyed, all favored support for open ranges, and 17 favored support for role-based value constraints.

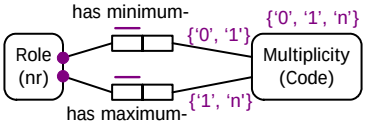


Figure 13 This Information Engineering metamodel fragment includes value constraints on roles.

2.10 Ring constraints

The ORM1 Visio tool uses the following English abbreviations for various *ring constraints*: ir = irreflexive, as = symmetric, ans = antisymmetric, it = intransitive, ac = acyclic, sym =symmetric. Ring constraints are displayed as a list of one or more of these options, appended to a ring symbol "O", and connected to the two relevant roles (if placed very close, the connection display is suppressed). For example, the reporting relationship is declared to be acyclic and intransitive as shown in Figure 14(a).

This ORM1 notation for ring constraints has two main disadvantages: the abbreviations are specific to English (e.g. Japanese readers might not find "ir" to be a memorable choice for irreflexivity); also the ring constraint display tends to cross over other lines (as in the example).To remove the English bias, and help the user understand the semantics of the constraint, ORM 2 uses *intuitive icons*. To reduce edge crossings, ORM 2 *omits role links* if the predicate has just two roles played by the same object type (or compatible object types). For example, in ORM 2 the reporting relationship is declared to be acyclic and intransitive as shown in Figure 14(b).

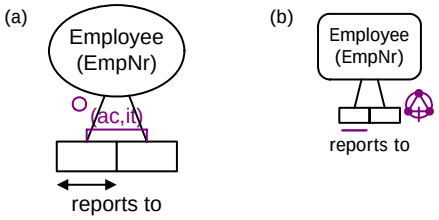


Figure 14 Acyclic and Intransitive ring constraints depicted in (a) ORM and (b) ORM 2.

The ORM 2 icons for ring constraints are loosely based on my icons for teaching ring constraints, where small circles depict objects, arrows depict relationships, and a bar indicates what is forbidden (see Figure 15).

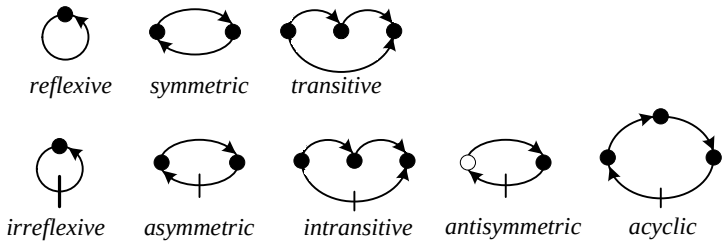


Figure 15 The original icons used for teaching ring constraints.

Reflexive means the object bears the relationship to itself. Symmetric means that if the first bears the relationship to the second, then the second bears that relationship to the first. Transitive means that if the first bears the relationship to the second, and the second to the third, then the first bears the relationship to the third. Irreflexive means the object cannot bear the relationship to itself. Asymmetric means that if the first bears the relationship to the second, then the second cannot bear that relationship to the first. Intransitive means that if the first bears the relationship to the second, and the second to the third, then the first cannot bear the relationship to the third. Anti-symmetric means that if the objects are different, then if the first bears the relationship to the second, then the second cannot bear that relationship to the first. The different fills in the anti-symmetric icon indicate that the inverse relationship is forbidden only if the two objects differ (in the other icons, we allow that the objects may be the same). Acyclic means that a chain of one or more instances of that relationship cannot form a cycle (loop). The meaning of these original icons should be intuitively obvious.

For diagramming purposes, these teaching icons they take up too much room, especially when combinations of ring constraints apply. So simplifying adaptations were made to ensure that the final icons are distinguishable on screen and in print, while maintaining a compact footprint. The proposed icons print clearly at 600 dpi, and are readable on screens at typical resolutions used for industrial modeling. They may be distinguished on low resolution screens by increasing the zoom level.

ORM 2 provides an icon for each the ten simple or combined ring constraint choices supported by the current Visio ORM1 tool (see Figure 16). In contrast to the teaching icons, arrow-heads are removed (they are assumed), and relevant pairs of constraints are collapsed to a single icon. While the simplified icons are less intuitive than the teaching icons, once their origin is explained it should be easy to remember their meaning. Of the 18 experts surveyed, 14 agreed to the new ring constraint icons.

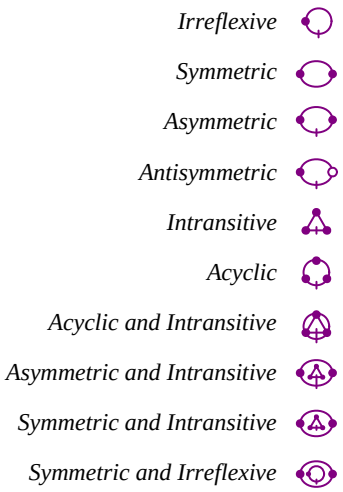


Figure 16 ORM 2 icons for ring constraints.

2.11 Subtyping

The ORM1 arrow notation for subtyping will remain, perhaps supplemented by use of Euler diagrams as an alternative display option for simple cases. ORM 2 adds support for explicit display of *subtype exclusion* (⊗) *and exhaustion* (⊙) *constraints*, overlaying them when combined, as shown in Figure 17. As such constraints are typically implied by other constraints in conjunction with subtype definitions, their display may be toggled on/off. Of the 18 experts surveyed, 17 approved this extension.

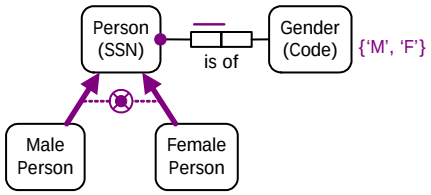


Figure 17 Explicit display of exclusion and exhaustion constraints for a subtyping scheme.

ORM 2 will allow both *defined subtypes and asserted subtypes* (no subtype definition provided). *Subtype definition*s will be supported in a high level formal language rather than mere comments, and may be displayed in text boxes on the diagram.

2.12 Role Path Disambiguation

In ORM, it was possible to encounter constraints involving role paths where the diagram is ambiguous as to which role paths are intended. ORM 2 removes that ambiguity by enabling the display of role-sequence numbers to be toggled on/off. For example, the objectification example discussed earlier has implicit link fact types as shown in Figure 18(a) to enable navigation between Playing, Person and Sport. This objectification is treated as a view of the internal representation shown in Figure 18(b), where the role-pairs that feature as arguments of the equality constraint are highlighted by including their role sequence numbers (1.1, 1.2; 2.1, 2.2).

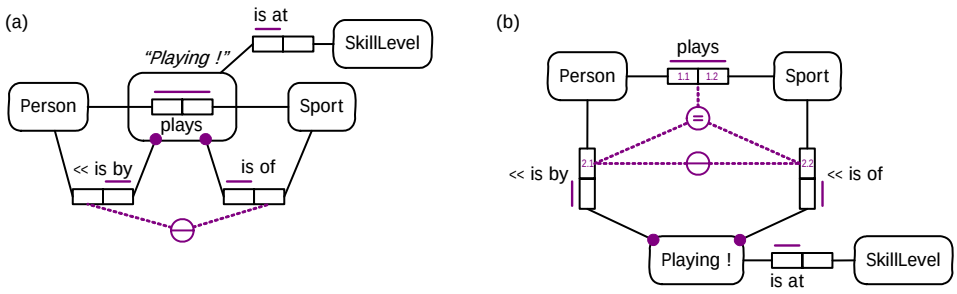


Figure 18 Explication of the objectification in Figure 6 of Person plays Sport as Playing.

2.13 Modalities

In ORM 2, business rules have a *modality* that is either alethic or deontic [19]. *Alethic* rules are necessarily true of the business domain (e.g. [It is necessary that] Each Person was born in at most one Country), whereas *deontic* rules are obligations that might be violated (e.g. It is obligatory that each Person is the husband of at most one Person).

ORM 2 distinguishes modalities by using different colors (by default, violet for alethic and blue for deontic). A further mark (e.g. "o" for obligatory) will be added to deontic constraints to enable them to be distinguished by means other than color. A final decision on this mark is yet to be made.

3 The ORM 2 Textual Notation

As in the Visio ORM1 tool, all components of an ORM schema (e.g. fact types and graphical constraints) will have automated *verbalizations*. The ORM 2 constraint verbalization mechanism is superior in several ways to that of the ORM1 tool, and uses improvements discussed elsewhere [19]. Details of the improved verbalization support will be provided in subsequent technical reports.

Unlike the Visio ORM1 tool, the ORM 2 tool will also support a high level, formal *textual language for inputting ORM schemas* (including fact types, constraints and derivation rules), *ORM queries*, and possibly population changes (fact addition and deletion). Although similar in many respects to the verbalization language, the input language will often enable constraints to be entered in a more concise form. The tool will generate code to implement the semantics conveyed by the textual language. The textual language will cover all the semantics conveyed by the graphical language, as well as additional semantics (e.g. constraints that cannot be captured graphically).

Textual constraints may be noted on the diagram by *footnote numbers*, with the textual reading of the constraints provided in *footnotes* that can be both printed and accessed interactively by clicking the footnote number. Figure 19 provides an example. Of the 18 experts surveyed, 17 approved the use of footnotes for textual constraints.

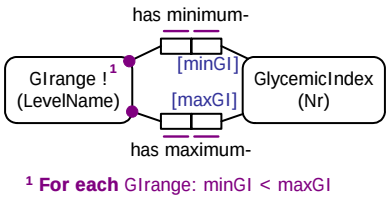


Figure 19 Textual constraints appear as footnotes in ORM 2.

Derivation rules may be specified for derived object types (subtype definitions) and derived fact types. ORM 2 allows *fully-derived subtypes* (full subtype definition provided), *partly-derived subtypes* (partial subtype definition provided) and *asserted subtypes* (no subtype definition provided). *Subtype definition*s will be supported as derivation rules in a high level formal language rather than mere comments, and may be displayed in text boxes as footnotes on the diagram. Iff-rules are used for full derivation, and if-rules for partial derivation. Here are sample rules in ORM 2's textual language for fully and partly derived subtypes respectively:

Each Australian is a Person who was born in Country 'AU'.

$Person_1$ is a Grandparent if $Person_1$ is a parent of some $Person_2$ who is a parent of some $Person_3$.

The final grammar for the textual language is still to be determined, but should support declaration of ORM models and queries in *relational style*, *attribute style*, and *mixed style*. Relational style uses predicate readings (e.g. the subtype definitions above), while attribute style uses role names. Attribute style is especially useful for derivation rules and textual constraints of a mathematical nature (e.g. see Figure 19).

As an example of a derivation rule for a derived fact type, we may define the uncle association in relational style thus: $Person_1$ is an uncle of $Person_2$ iff $Person_1$ is a brother of a $Person_3$ who is a parent of $Person_2$. Assuming the role names "brother" and "parent" are declared, we may also specify the rule in attribute style thus: For each Person: uncle = brother of parent. Further examples may be found elsewhere [19, 21].

4 Conclusion

This report discussed a project under way to specify and provide open source tool support for a second generation ORM (called ORM 2), that provides significant advances over current ORM technology. Proposed changes to the graphical notation were described, and their motivation explained. Results from a survey of ORM experts with regard to these changes were noted. Various enhancements to the ORM notation were examined, and some improvements from a tooling perspective were identified.

The tools project team is currently researching extensions to ORM in several areas, including richer support for join constraints (e.g. distinguishing inner-outer join semantics, displaying complex join constraints, and role path disambiguation [20]), extended open/closed world semantics, and deontic/alethic constraint distinctions.

Parties who own a copy of Visio (Standard edition or higher) and who wish to explore the new notation using models of their own may download a zip file containing the Visio ORM 2 stencil and template, plus a sample model file from the following site: www.orm.net/ORM2_Beta.zip. This ORM 2 stencil is for drawing only—it does not generate code.

At the time of writing, the ORM 2 modeling tool is in the early stages of development. Considerable progress has been made on entering ORM models, verbalizing them, and mapping them to object models, but mapping ORM schemas to relational and XML schemas has not yet begun. The diagrammer is currently limited to a single page diagrams. Future work is planned to support multi-page diagrams with layer toggles to hide/show features, and several abstraction mechanisms (e.g. decomposing models into multiple levels of refinement).

Acknowledgements

This paper benefited from discussion with other members of the ORM 2 tool project team at Neumont University. The new notation for ring constraints is due largely to Tyler Young, Scott Baldwin, Nickolas Maly, and Jeremy Wilde. These students also created the ORM 2 Visio stencil, with technical assistance from Scott LeGendre (Microsoft) on use of dynamic glue. Other students who worked on the tool project include Aaron Bowen, Jaron Briggs, Orion Buhler, Adam Burt, Korvyn Dornseif, Paul Ewert, Darren Flynn, Brant Frank, Adam Greenwood, Corey Kaylor, Michelle Ouzts, Kevin Owen, Aaron Shumway, Joseph Tanner and Jeff Wilde. The team also includes five faculty members (Andy Carver, Ed Crowson, Matt Curland, Terry Halpin, and Tony Morgan) with Matt Curland as lead programmer.

The following kindly responded to our survey on the ORM graphical notation: Don Baisley (Unisys, USA); Dick Barden (InConcept, USA); Scott Becker (Orthogonal Software, USA); Linda Bird (DSTC, Australia); Anthony Bloesch (Microsoft, USA); Necito dela Cruz (Guidant Corporation, USA); Dave Cuyler (Sandia National Labs, USA); Lance Delano (Microsoft, USA); Jan Dietz (Delft University of Technology, The Netherlands); Ken Evans (AMBO.biz Ltd , UK); Gordon Everest (University of Minnesota, USA); Brian Farish (Vertaasis, USA); Pat Hallock (InConcept, USA); Bill MacLean (Avnet, Inc, USA); John Miller (Perpetual Data Systems, USA); Borje Olsson (Object Role Consulting, Sweden); Erik Proper (Radbaud University Nijmegen, The Netherlands); Pieter Verheyden (STARLAB, University of Brussels).

References

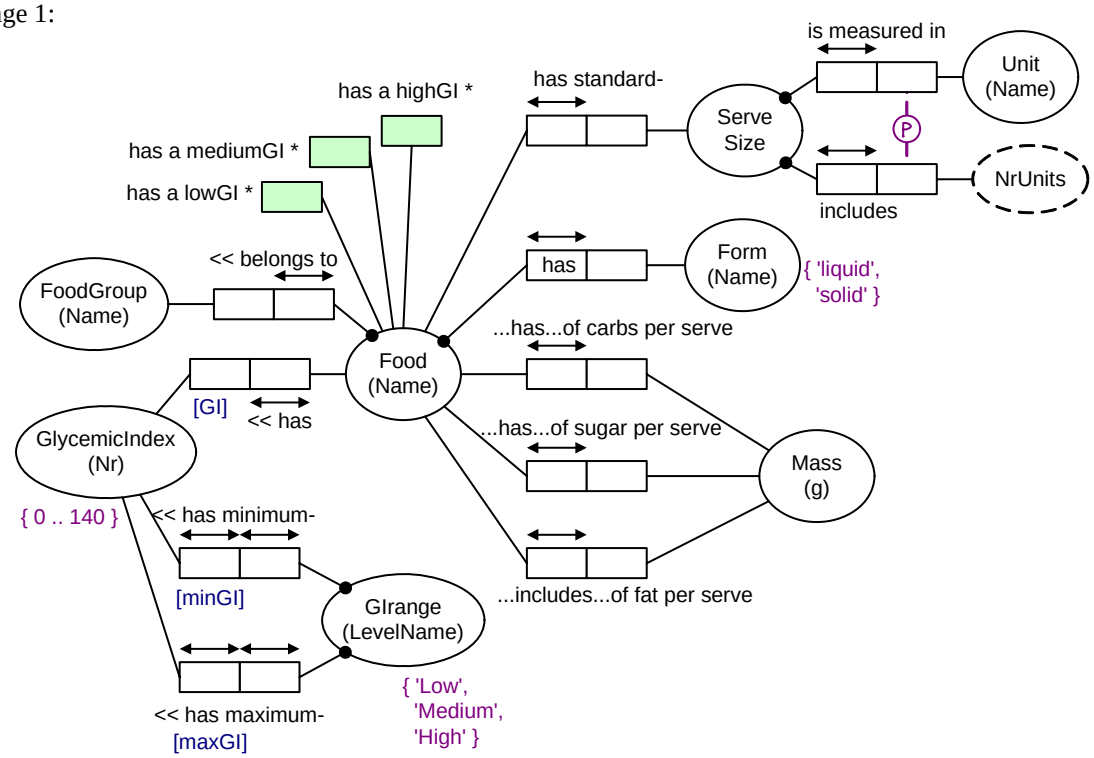
1. Bakema, G., Zwart, J. & van der Lek, H. 1994, 'Fully Communication Oriented NIAM', *NIAM-ISDM 1994 Conf. Working Papers*, eds G. M. Nijssen & J. Sharp, Albuquerque, NM, pp. L1-35.
2. Bakema, G., Zwart, J. & van der Lek, H. 2000, *Fully Communication Oriented Information Modellin*g, Ten Hagen Stam, The Netherlands.
3. Bird, L., Goodchild, A. & Halpin, T.A. 2000, 'Object Role Modeling and XML Schema', *Conceptual Modeling – ER2000*, Proc. 19th Int. ER Conference, Salt Lake City, Springer LNCS 1920, pp. 309-322.
4. 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.
5. CaseTalk website: http://www.casetalk.com/php/.
6. Chen, P. P. 1976, 'The entity-relationship model—towards a unified view of data'. *ACM Transactions on Database Systems*, 1(1), pp. 9–36.
7. Cuyler, D. & Halpin, T. 2005, 'Two Meta-Models for Object-Role Modeling', *Information Modeling Methods and Methodologies, eds* J. Krogstie, T. Halpin, & K. Siau, Idea Publishing Group, Hershey PA, USA (pp. 17-42).
8. Demey J., Jarrar M. & Meersman R. 2002, 'A Markup Language for ORM Business Rules', in Schroeder M. & Wagner G. (eds.), *Proc. International Workshop on Rule Markup Languages for Business Rules on the Semantic Web* (RuleML-ISWC02 workshop), pp. 107-128, online at http://www.starlab.vub.ac.be/publications/STARpublications.htm.

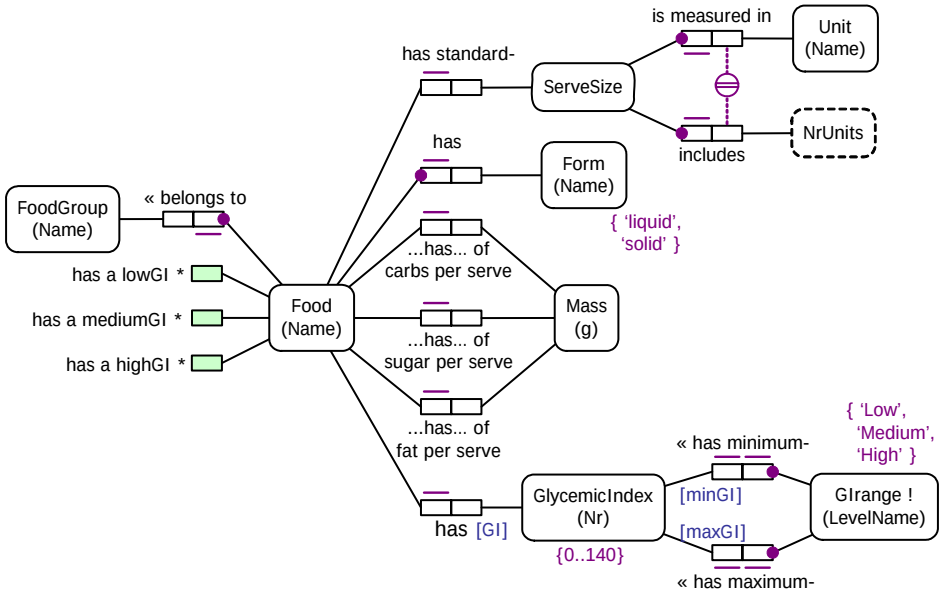
9. De Troyer, O. & Meersman, R. 1995, 'A Logic Framework for a Semantics of Object Oriented Data Modeling', *OOER'95, Proc. 14th International ER Conference*, Gold Coast, Australia, Springer LNCS 1021, pp. 238-249.
10. DogmaModeler: www.starlab.vub.ac.be/research/dogma/dogmamodeler/dm.htm.
11. Embley, D.W. 1998, *Object Database Management*, Addison-Wesley, Reading MA, USA.
12. Falkenberg, E. D. 1976, 'Concepts for modelling information', in Nijssen G. M. (ed) *Proc 1976 IFIP Working Conf on Modelling in Data Base Management Systems*, Freudenstadt, Germany, North-Holland Publishing, pp. 95-109.
13. Halpin, T. 1989, 'A Logical Analysis of Information Systems: static aspects of the data-oriented perspective', doctoral dissertation, University of Queensland.
14. 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.
15. Halpin, T. 2001, *Information Modeling and Relational Databases*, Morgan Kaufmann, San Francisco.
16. Halpin, T. 2003, 'Uniqueness Constraints on Objectified Associations', *Journal of Conceptual Modeling*, Oct. 2003. URL: www.orm.net/pdf/JCM2003Oct.pdf.
17. 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).
18. Halpin, T. 2004, 'Information Modeling and Higher-Order Types', *Proc. CAiSE'04 Workshops*, vol. 1, (eds Grundspenkis, J. & Kirkova, M.), Riga Tech. University, pp. 233-48. Online at http://www.orm.net/pdf/EMMSAD2004.pdf.
19. 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.
20. Halpin, T. 2005, 'Constraints on Conceptual Join Paths', *Information Modeling Methods and Methodologies, eds J.* Krogstie, T. Halpin, T.A. & K. Siau, Idea Publishing Group, Hershey PA, USA, pp. 258-77.
21. Halpin, T. 2005, 'Verbalizing Business Rules: Part 11', *Business Rules Journal*, Vol. 6, No. 6. URL: http://www.BRCommunity.com/a2005/b238.html.
22. Halpin, T. 2005, 'Objectification', *Proc. CAiSE'05 Workshops, vol. 1*, eds J. Calestro & E. Teniente, FEUP Porto (June), pp. 519-31.
23. Halpin, T., Evans, K, Hallock, P. & MacLean, W. 2003, *Database Modeling with Microsoft® Visio for Enterprise Architects*, Morgan Kaufmann, San Francisco.
24. Halpin, T. & Proper, H. 1995, 'Database schema transformation and optimization', *Proc. OOER'95: Object-Oriented and Entity-Relationship Modeling*, Springer LNCS, vol. 1021, pp. 191-203.
25. Halpin, T. & Wagner, G. 2003, 'Modeling Reactive Behavior in ORM'. *Conceptual Modeling – ER2003*, Proc. 22nd ER Conference, Chicago, October 2003, Springer LNCS.
26. ter Hofstede, A. H. M., Proper, H. A. & Weide, th. P. van der 1993, 'Formal definition of a conceptual language for the description and manipulation of information models', *Information Systems*, vol. 18, no. 7, pp. 489-523.
27. ter Hofstede A. H. M, Weide th. P. van der 1993, 'Expressiveness in conceptual data modeling', *Data and Knowledge Engineering*, 10(1), pp. 65-100.
28. Infagon website: http://www.mattic.com/Infagon.html.
29. Lyons, J. 1995, *Linguistic Semantics: An Introduction*, Cambridge University Press: Cambridge, UK.
30. Meersman, R. M. 1982, *The RIDL conceptual language*. Research report, Int. Centre for Information Analysis Services, Control Data Belgium, Brussels.
31. Object Management Group 2003, *UML 2.0 Infrastructure Specification*. Online: www.omg.org/uml.
32. Object Management Group 2003, *UML 2.0 Superstructure Specification*. Online: www.omg.org/uml.
33. Rumbaugh J., Jacobson, I. & Booch, G. 1999, *The Unified Language Reference Manual*, Addison-Wesley, Reading, MA.
34. VisioModeler download site: http://www.microsoft.com/downloads/results.aspx?displaylang=en&freeText=VisioModeler.
35. Warmer, J. & Kleppe, A. 1999, *The Object Constraint Language: precise modeling with UML*, Addison-Wesley.
36. Wintraecken J. 1990, *The NIAM Information Analysis Method: Theory and Practice*, Kluwer, Deventer, The Netherlands.

Appendix: Sample models

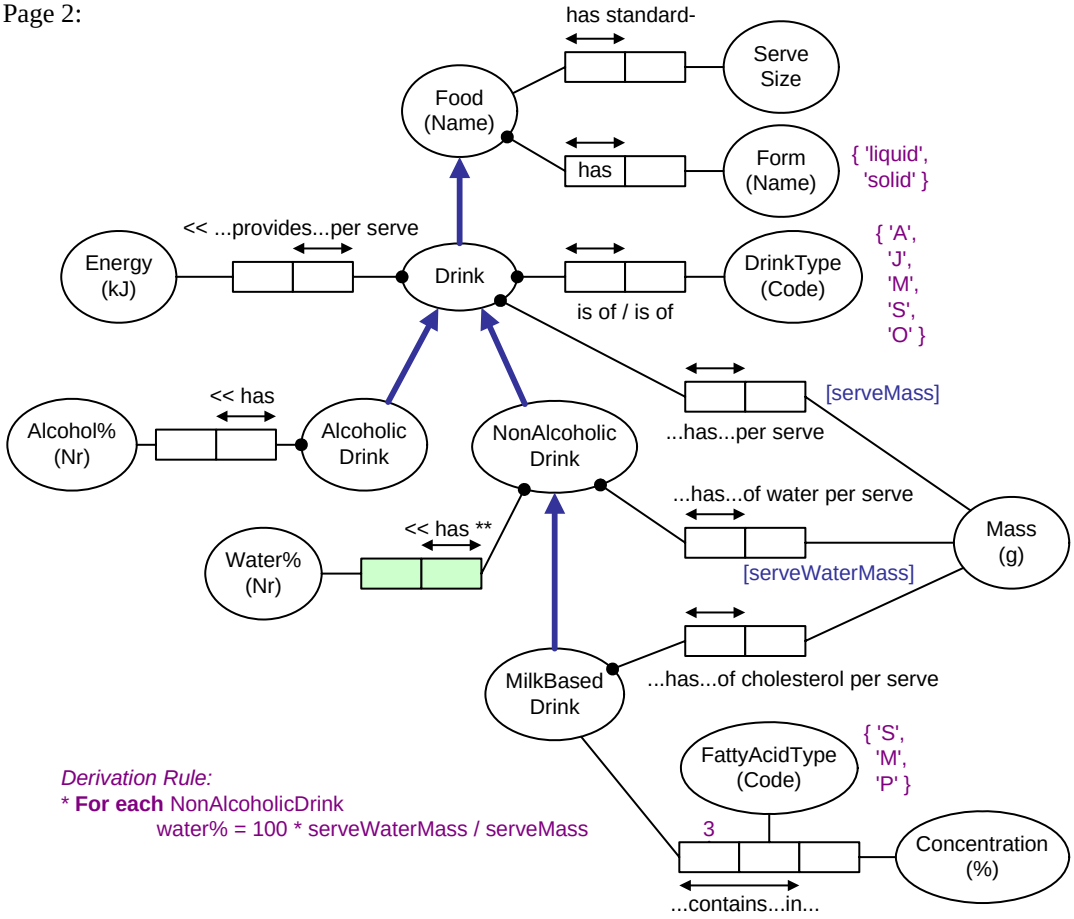
To give a feeling for the difference made by the ORM 2 notation, a 3 page Diet model (minus subtype definitions, textual constraints, and some derivation rules) is shown in both the Visio ORM source model notation and the ORM 2 notation. With either notation, the mandatory role dots may be placed at either the object type end or the role end. The color fill for derived fact types is an optional choice.

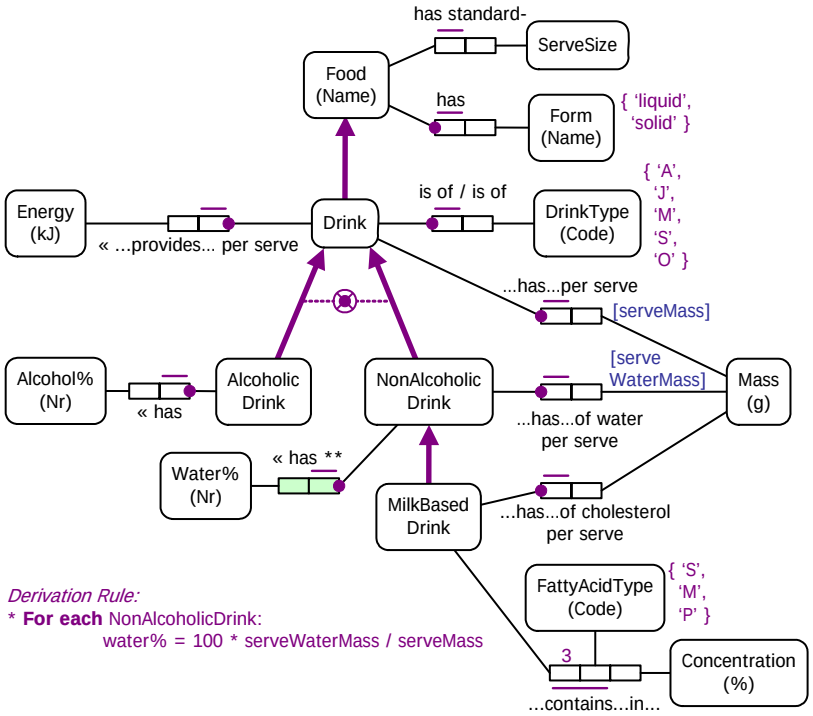
Page 1:



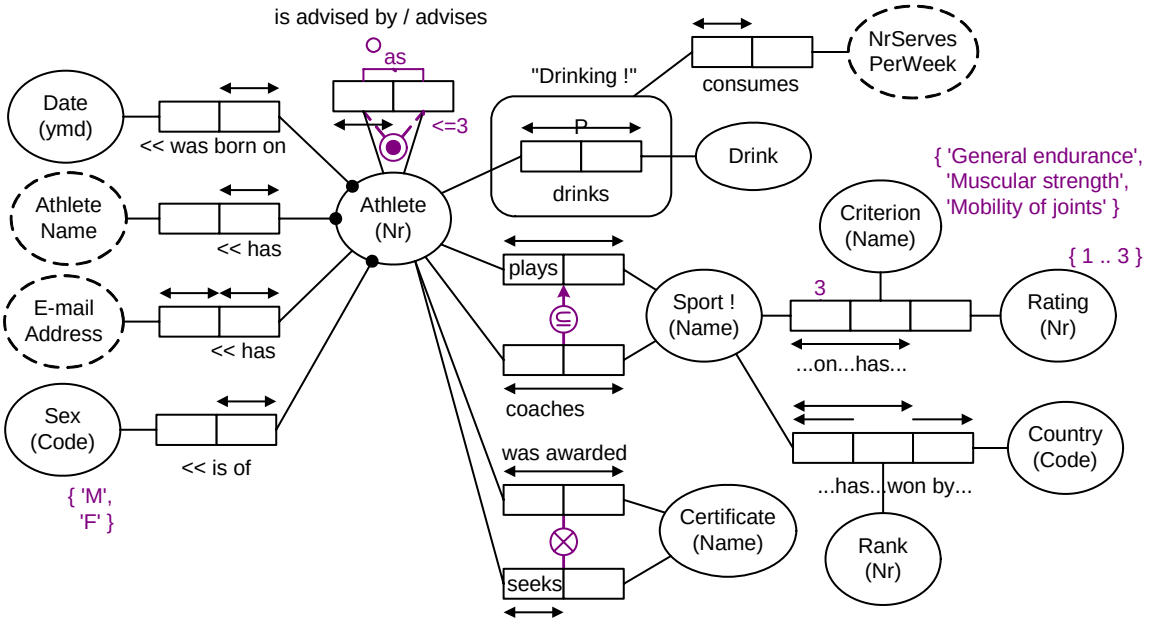


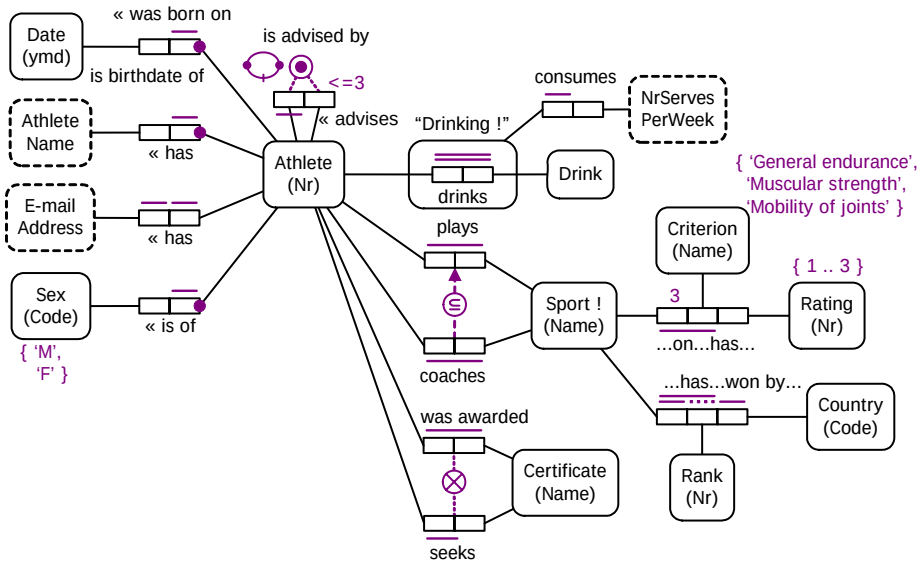
Page 2:


has standard-
Serve Size
Food (Name)
has
Form (Name)
{ 'liquid', 'solid' }
<< ...provides...per serve
Energy (kJ)
Drink
DrinkType (Code)
{ 'A', 'J', 'M', 'S', 'O' }
is of / is of
[serveMass]
...has...per serve
<< has
Alcohol% (Nr)
Alcoholic Drink
NonAlcoholic Drink
...has...of water per serve
Mass (g)
<< has **
Water% (Nr)
[serveWaterMass]
MilkBased Drink
...has...of cholesterol per serve
FattyAcidType (Code)
{ 'S', 'M', 'P' }
3
Concentration (%)
...contains...in...
Derivation Rule:
* For each NonAlcoholicDrink
water% = 100 * serveWaterMass / serveMass


has standard-
ServeSize
Food (Name)
has
Form (Name)
{ 'liquid', 'solid' }
Energy (kJ)
« ...provides... per serve
Drink
is of / is of
DrinkType (Code)
{ 'A', 'J', 'M', 'S', 'O' }
...has...per serve
[serveMass]
Alcohol% (Nr)
« has
Alcoholic Drink
NonAlcoholic Drink
[serve WaterMass]
Mass (g)
...has...of water per serve
« has **
Water% (Nr)
MilkBased Drink
...has...of cholesterol per serve
FattyAcidType (Code)
{ 'S', 'M', 'P' }
3
Concentration (%)
...contains...in...
Derivation Rule:
* For each NonAlcoholicDrink:
water% = 100 * serveWaterMass / serveMass

Page 3:


is advised by / advises
as
<=3
Date (ymd)
<< was born on
Athlete Name
<< has
E-mail Address
<< has
Sex (Code)
<< is of
{ 'M', 'F' }
Athlete (Nr)
"Drinking !"
P
drinks
consumes
NrServes PerWeek
Drink
{ 'General endurance', 'Muscular strength', 'Mobility of joints' }
Criterion (Name)
{ 1 .. 3 }
plays
coaches
Sport ! (Name)
3
...on...has...
Rating (Nr)
was awarded
seeks
Certificate (Name)
...has...won by...
Country (Code)
Rank (Nr)


« was born on
is birthdate of
Date (ymd)
is advised by
<=3
« advises
Athlete Name
« has
E-mail Address
« has
Sex (Code)
« is of
{ 'M', 'F' }
Athlete (Nr)
"Drinking !"
consumes
NrServes PerWeek
drinks
Drink
{ 'General endurance', 'Muscular strength', 'Mobility of joints' }
plays
coaches
Criterion (Name)
{ 1 .. 3 }
Sport ! (Name)
3
...on...has...
Rating (Nr)
...has...won by...
Country (Code)
was awarded
seeks
Certificate (Name)
Rank (Nr)