Enhanced Verbalization of ORM Models

Matthew Curland[1] and Terry Halpin[2]

[1] ORM Solutions, USA
[2] INTI International University, Malaysia and LogicBlox, Australia
mcurland@live.com, terry.halpin@logicblox.com

Abstract. Fact-oriented modeling approaches such as Object-Role Modeling (ORM) validate their models with domain experts by verbalizing the models in natural language, and by populating the relevant fact types with concrete examples. This paper extends previous work on verbalization of ORM models in a number of ways. Firstly, it considers some ways to better ensure that generated verbalizations are unambiguous, including occasional use of lengthier verbalizations that are tied more closely to the underlying logical form. Secondly, it provides improved verbalization patterns for common types of ORM constraints, such as uniqueness and mandatory role constraints. Thirdly, it provides an algorithm for verbalizing external uniqueness and frequency constraints over roles projected from join paths of arbitrary complexity. The paper also includes some discussion of how such verbalization enhancements were recently implemented in the Natural ORM Architect (NORMA) tool.

1 Introduction

When designing an information system, it is important to ensure that the data model accurately reflects the data requirements for the relevant business domain by validating the model with someone who understands the business domain, i.e. a domain expert. Since domain experts are often nontechnical, the data model should be communicated to them in language that is intelligible to them, without requiring mastery of a complex, technical syntax such as that of the Object Constraint Language (OCL) [17]. Fact-oriented modeling approaches are specifically designed to facilitate communication between modelers and domain experts by verbalizing the data model in natural sentences and populating the model's fact types with concrete examples of fact instances. A fact type corresponds to a set of typed predicates of arity one (e.g. Person smokes), two (e.g. Person was born on Date), or higher (e.g. Item contains Item in Quantity). This approach differs from Entity relationship (ER) modeling [3] and class diagramming within the Unified Modeling Language (UML) [19], which encode some facts in attributes (e.g. Person.isSmoker, Person.birthdate).

Fact-orientation's attribute-free nature promotes semantic stability (e.g. no remodeling is needed to talk about an attribute) and its graphical constraint notation for data modeling is much richer than that of industrial ER or UML. The family of fact-oriented modeling approaches include various dialects such as Object-Role Modeling (ORM), Natural-language Information Analysis Method (NIAM) [20],

P. Herrero et al. (Eds.): OTM 2012 Workshops, LNCS 7567, pp. 399–408, 2012.

Fully-Communication Oriented Information Modeling (FCO-IM) [1] and the Predicator Set Model (PSM) [16]. This paper focuses on recent verbalization work within second generation ORM (ORM 2) [6]. Overviews of ORM may be found in [8, 9], and a detailed coverage in [14].

Our previous work on ORM verbalization and its automation is discussed in [13, 5, 10]. The rest of this paper discusses extensions to this work, and is structured as follows. Section 2 considers various ways to help ensure that generated verbalizations are unambiguous, noting some cases where lengthier verbalizations based more closely on logical formalization may be needed. Section 3 provides improved verbalization patterns for some common types of ORM constraints, such as uniqueness and mandatory role constraints. Section 4 provides an algorithm for verbalizing external uniqueness and frequency constraints over roles projected from join paths of arbitrary complexity, and discusses how such verbalization enhancements were implemented in the Natural ORM Architect (NORMA) tool.

2 Avoiding Ambiguity

In Figure 1(a), the binary fact types Person was born in Year and Person speaks Language are displayed with infix predicate readings. Readings are left-to-right unless reversed by an arrow-tip. The bar over the left-hand role of the birth predicate depicts a uniqueness constraint that we verbalize in ORM as "**Each** Person was born in **at most one** Year", and the dot on the line connected to the left-hand role of the speaks predicate depicts a mandatory role constraint that verbalizes as "**Each** Person speaks **some** Language". Using sorted logic with mixfix predicates, these constraints formalize respectively as $\forall p$:Person $\exists^{0..1} y$:Year p was born in y, and $\forall p$:Person $\exists l$:Language p speaks l.

We verbalize the universal quantifier $\forall$ as "**each**" rather than "**all**" or "**every**" because linguistically "**each**" is always a *distributive* quantifier, applying to individuals one at a time [1, p. 352]. In contrast, "**all**" and "**every**" may be used both collectively and distributively, which may lead to ambiguity. For example, "All baby whales weigh more than all baby humans" is true if "all" is interpreted distributively, but false if it is interpreted collectively (because there are many more humans). Another reason for avoiding "all" is that it requires pluralization (e.g. all persons). By default we verbalize the existential quantifier $\exists$ as "**some**", although we do plan to also support "**a**" or "**an**" as appropriate. Linguists regard the pattern "every-some" to be ambiguous [17]. For example, "Every person speaks some language" could mean either (a) For each person, that person speaks some language, or (b) There is some language such that every person speaks that language.

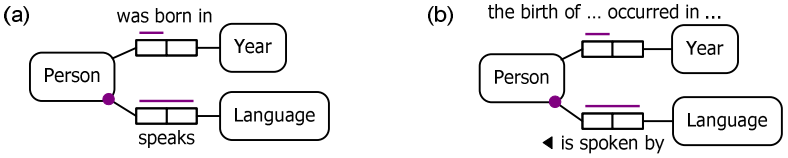

Fig. 1. Some predicate reading choices for ORM fact types

Because ORM supports mixfix predicates, allowing object terms to be placed at any position in a sentence, a predicate reading might include *front text* (before the first term) and/or *trailing text* (after the last term). The presence of front text often requires a different verbalization pattern to avoid ambiguity. For example, applying the simple pattern used earlier, the uniqueness constraint on the birth predicate in Figure 1(b) verbalizes thus "The birth of **each** Person occurred in **at most one** Year" which could be misunderstood by some users to mean that everybody was born in the same year. Hence we instead use the following verbalization: **For each** Person, the birth of **that** Person occurred in **at most one** Year. Our verbalization of a constraint on a role also depends on whether there is a predicate reading that starts at that role. For example, the mandatory role role constraint in Figure 1(b) verbalizes as "**For each** Person, **some** Language is spoken by **that** Person", rather than "**Some** Language is spoken by **each** Person" which linguistically untutored users might take to mean that there is some language that everyone speaks.

In previous work, we often used "**the same**" as a linguistic quantifier for verbalizing some leading existential quantifiers. For example, we verbalized the many-to-many nature of the spanning uniqueness constraint on the speaks predicate in Figure 1(a) thus: **It is possible that the same** Person speaks **more than one** Language **and that more than one** Person speaks **the same** Language. The problem with this is that "the same" can sometimes be misinterpreted as referring back to a previous instance. To avoid such misunderstanding, we now use the following verbalization if forward and converse predicate readings are available: **It is possible that some** Person speaks **more than one** Language **and that some** Language is spoken by **more than one** Person. If only a forward reading is available, we verbalize thus: **It is possible that some** Person speaks **more than one** Language **and that for some** Language **more than one** Person speaks **that** Language. In a few cases where "**the same**" is unambiguous, we still use it to provide a more natural verbalization.

To render natural rather than mathematical verbalizations, we verbalize correlations by using relative pronouns (e.g. "**who**", "**that**"), reflexive pronouns (e.g. "**itself**"), demonstratives (e.g. "**that**"), and typed variables, possibly subscripted (e.g. Person, Person_1), rather than resort to untyped variables (e.g. *x*, *y*) which are used in some controlled natural languages such as Common Logic Controlled English (CLCE) [20]. For further, related details on our Formal ORM Language (FORML) and its use as an input language as well as an output verbalization language, see [15].

Adverbs in predicate readings (e.g. "rarely" in Figure 2(a)) sometimes requires special care to clearly verbalize constraints, even though the meaning of fact instance verbalizations is clear (e.g. The Person named 'Ann Jones' rarely drives the Car that has CarRegNr 'ABC123'). For example, using our earlier pattern, the uniqueness constraint in Figure 2(a) verbalizes thus: **Each** Person rarely drives **at most one** Car.

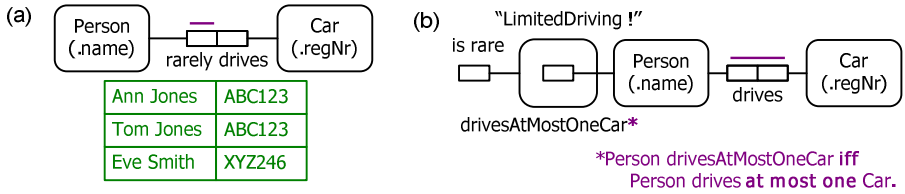

Fig. 2. Adverbs can raise further ambiguity issues with constraint verbalization

This ambiguous verbalization could mean either (a) For each person there is at most one car that he/she rarely drives, or (b) For each person, it is rare that he/she drives at most one car. There are competing linguistic and logical theories for dealing with adverbs that might shed some light on this issue [17, ch. 12], but these theories are complex, and their utilization would seem to require either substantial linguistic parsing machinery or the possession of sophisticated grammatical expertise by the user. We feel that the best way to resolve such ambiguities is to offer an expanded verbalization that more closely matches the underlying logical form of the ORM constraints. Every ORM constraint unambiguously maps to a logical form (e.g. see [4]), and in this example the uniqueness constraint requires interpretation (a), since uniqueness on a role means that entries in the fact column for that role are unique. This interpretation is clearly captured by the logical form of the uniqueness constraint: $\forall p$:Person $\exists^{0..1} c$:Car p rarely drives c, which directly maps to the expanded verbalization: **For each** Person, **there is at most one** Car **such that that** Person rarely drives **that** Car. Interpretation (b) is radically different since it predicates the rareness to the situation of a person driving at most one car, which may be modeled in ORM as in Figure 2(b) by objectifying a derived fact type. ORM formalizes objectification as situational nominalization rather than propositional nominalization [14].

While unambiguously rendering the required meaning, expanded logical form verbalizations are lengthy and awkward in comparison with our typical verbalizations (e.g. compare "**Each** Person was born in **at most one** Year" with "**For each** Person, **there is at most one** Year **such that that** Person was born in **that** Year"). Moreover, the cases where our simple verbalization patterns are ambiguous are rare in practice. As a pragmatic compromise, we generate the simple verbalization patterns by default, but plan to give users the option of seeing the logical form verbalizations whenever they wish (e.g. when the default verbalization provided seems unclear to them). For users who are poor at English grammar or logic, this facility would have the added benefit of improving their grammatical or logical expertise.

Verbalization of constraints and derivation rules may involve use of logical operators. To disambiguate the order in which logical operators are evaluated, we emulate bracketing by careful use of indentation rather than rely on artificial precedence rules (e.g. evaluate **and** before **or**) as is done in some other controlled natural languages. We also avoid well known ambiguous patterns (e.g. "all are not", which could mean either "none are" or "not all are"). Further examples of FORML in this regard may be found in [15].

3 Improved Verbalization Patterns

Because of ORM's rich constraint notation and use of mixfix predicates of any arity, the number of constraint verbalization patterns is substantial. We have space here to discuss only two cases: (a) external uniqueness constraints on simple join paths, and (b) inclusive-or constraints where each role starts a predicate reading with no front text. Verbalizations for other cases may be viewed interactively by invoking the verbalizer in a recent build of NORMA [4], which we implemented as a free plug-in to Microsoft Visual Studio.

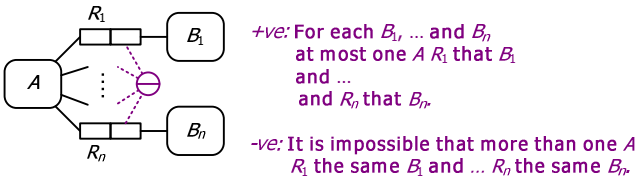

Fig. 3. New verbalization patterns for external uniqueness constraints

Figure 3 shows our new alethic constraint verbalization patterns for an *external uniqueness constraint* where the join object type *A* starts predicate readings with no front text for each of the *n* constrained predicates R_1 .. R_n. The object types B_1 .. B_n may be entity or value types, and are not necessarily distinct. If two or more of the B_i are identical, their instances in the verbalization must be distinguished by subscripting. Previously, our verbalizations for such cases required declaration of a context clause to specify the relevant fact types, followed by another clause that referenced the constrained role combination within that context. Our new verbalizations are closer to the underlying logical form, and are easier for users to understand.

We provide positive verbalizations for all constraints, and negative verbalizations as well for some constraints. Positive verbalizations stress what must be satisfied while negative verbalizations indicate what cannot or must not be the case. Positive alethic constraint verbalizations may optionally be prepended by the modal necessity operator □ "**It is necessary that**". The *deontic* versions prepend the positive reading by the obligation operator ***O*** "**It is obligatory that**" and replace ~◊ "**impossible**" in the negative reading by ***F*** "**forbidden**". For further discussion of constraint modalities and the use of modal operators see [8]. Here are two examples:

+ve, alethic: **For each** Building **and** RoomNr,
at most one Room is in **that** Building **and** has **that** RoomNr.

-ve, deontic: **It is impossible that**
more than one Arrow is from **the same** $Node_1$ **and** is to **the same** $Node_2$.

The above patterns do not work if any predicate has *front text, or a predicate does not start at A*. For example, consider the fact types: "the location of Room is in Building; RoomNr is of Room". For such cases we use another pattern leading to the following verbalization for the positive alethic case: **For each** Building **and** RoomNr, **there is at most one** Room **such that** the location of **that** Room is in **that** Building **and that** RoomNr is of **that** Room.

For all cases, the constraint has the following logical forms. The initial formula in each case assumes there are predicates $R_1..R_n$ starting at an *A* role. For each fact type F_i where this is not the case, replace xR_iy_i by y_iS_ix where *S* is the preferred predicate starting at B_i and $1 \le i \le n$.

+ve alethic: $\Box\ \forall y_1{:}B_1..y_n{:}B_n\ \exists^{0..1}x{:}A(xR_1y_1\ \&\ ..\ xR_ny_n)$
+ve deontic: $\boldsymbol{O}\ \forall y_1{:}B_1..y_n{:}B_n\ \exists^{0..1}x{:}A(xR_1y_1\ \&\ ..\ xR_ny_n)$
-ve alethic: $\sim\Diamond\ \exists y_1{:}B_1..y_n{:}B_n\ \exists^{2..}x{:}A(xR_1y_1\ \&\ ..\ xR_ny_n)$
-ve deontic: $\boldsymbol{F}\ \exists y_1{:}B_1..y_n{:}B_n\ \exists^{2..}x{:}A(xR_1y_1\ \&\ ..\ xR_ny_n)$

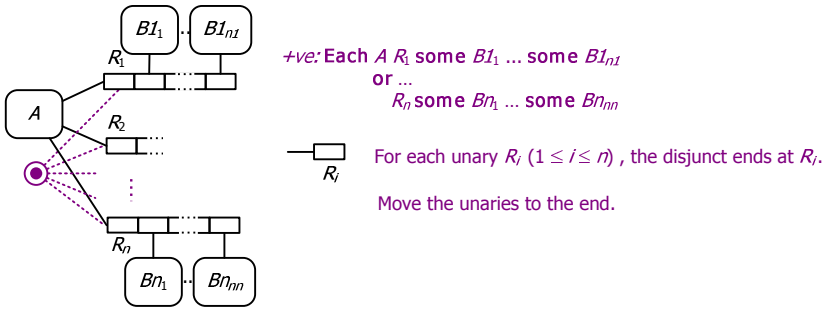

Fig. 4. A basic inclusive-or constraint verbalization pattern

Figure 4 shows the positive alethic verbalization pattern for an *inclusive-or constraint* where each constrained role starts a predicate reading with no front text. The object types are not necessarily distinct. Each predicate R_i may have ni roles ($ni \geq 0$) following the role played by A. So the predicates may all be of different arity (unary upwards). We verbalize all the non-unaries before all the unaries. The *deontic* version prepends "**it is obligatory that**" to the positive form. Here are two examples:

+ve, alethic: **Each** Partner became the husband of **some** Partner on **some** Date
or became the wife of **some** Partner on **some** Date.

+ve, deontic: **It is obligatory that each** Vehicle was purchased from **some** AutoRetailer
or is rented.

No negative verbalization is supplied for this constraint case, as it is awkward and adds no clarity. The logical form of the positive alethic constraint is shown below, where any disjuncts with unary predicates must appear only after all the disjuncts with non-unary predicates (if any). The deontic form simply replaces □ by $\boldsymbol{O}$.

+ve alethic: $\Box\ \forall x{:}A\ (\exists x_1{:}B1_1..x_{n1}{:}B1_{n1}\ Rxx_1..x_{n1}$
$\vee\ \ldots$
$\vee\ \exists x_1{:}Bn_1..x_{nn}{:}Bn_{nn}\ Rxx_1..x_{nn})$

4 Implementing Verbalizations Involving One Join Path

In ORM, an external uniqueness or external frequency constraint applies to roles in more than one fact type, so a join path must be associated with that constraint to formally specify the relationship between the fact types. In most common cases, NORMA is able to automatically determine the join path for common patterns; more complex cases require user entry. For this discussion, we assume that an error-free join path has been implicitly or explicitly created for the verbalized multi-fact type constraints. For cases where the optimized verbalization patterns are not available due

to complexity in the model structure or associated predicate readings, the verbalization engine relies on the join paths to generate verbalizations for the general logic-based forms.

We have previously discussed the use of *role paths* in NORMA to define fact type derivation rules [5]. This same underlying structure is also used to specify join paths for constraints. The primary difference is that fact type derivation rules project nodes from a role path onto roles of the derived fact type, while join paths project these same nodes onto *constraint roles*, meaning a use of role in the ordered list of roles restricted by the given constraint. In terms of predicate logic, the role path forms the body of a logic expression, while the projection targets (fact type roles for derivation and constraint roles for join paths) can be viewed as a list of variables in the head of the logic expression. Continuing with the logic parallels, each variable projected from the body (thus appearing in the head) is assumed to be *free* in the body, meaning that the value of the variable is known in the body and can be unambiguously referenced with a *that X* expression.

By default, the role path verbalization engine assumes that no variables are free in the path and existentially places [13] variables as needed. For example, given the Room/Building uniqueness example with the obvious role path (consisting of an and-split after Room), the path verbalization engine would produce *some Room is in some Building and has some RoomNr* for the simple reading, and *for some Room, the location of that Room is in some Building and that Room has some RoomNr* for the more complicated reading with front text.

Clearly, the path verbalizations with no free variables are of very little use, just as the body of an expression is of little use without a head. Forming full verbalizations, therefore, requires each type of role path use to tell the role path verbalizers which nodes are used in the head, thereby forming a meaningful verbalized statement. The general verbalization for a multi-fact type uniqueness constraint has gone through several stages:

1. The original form used a stated Context and no join path. This is not very readable and does not clearly state the relationship between the different fact types involved. The join information can be inferred for trivial cases, but this approach of simply listing the fact types is inadequate for non-trivial join paths.

   Context: Room is in Building;
   Room has RoomNr.
   In this context, each Building, RoomNr **combination occurs at most once.**

2. The first modification used a verbalized role path with no free variables to form the context. The main problem here is the ambiguity of the term *some* used to introduce variables in the verbalized path. The intent was to convey "any" in a universal sense (e.g. given *any* Room), but it can also be interpreted as applying to some (but not all) of the specified instances.

   Context: some Room is in **some** Building
   and has **some** RoomNr.
   In this context, each Building, RoomNr **combination occurs at most once.**

3. This form is improved by applying the universal quantifier to the projected variables before verbalizing the role path. Here, the items in the *for each* list correspond to the constraint roles, which are shown numbered on a diagram in NORMA when the constraint is selected. This form uses the *combination* suffix after the projection list so that we can use *that combination* without ambiguity in the later list. Note that this structure is used as the general form of a frequency constraint by modifying the quantifier (for example, replacing *occurs exactly once* with *occurs at most 2 times* verbalizes a frequency constraint over the same roles).

   For each RoomNr **and** Building **combination**,
   if some Room has **that** RoomNr
   and is in that Building
   then that combination occurs exactly once in this context.

   It is also important that we can use the stronger *exactly once* phrase here instead of the earlier *at most once*. This is possible because the context is fully defined, so the quantifier applies only to states that are known to exist, implying *at least once*. This differs from the earlier forms, which did not have a complete logical context and could use only the weaker *at most once*.

4. While formally correct, the 3rd verbalization form lacks elegance because it still relies on the abstract notions of *combination* and *context* to express its meaning. The desired general verbalization brings us directly to the pure logical form discussed earlier by applying the at-most-one numeric quantifier to *Room* before verbalizing the role path—with both the universally and numerically quantified variables free in the path body.

   For each Building **and** RoomNr,
   there is at most one Room **such that**
   that Room is in **that** Building
   and has **that** RoomNr.

While the fourth form is clearly the most readable, it adds a non-trivial implementation challenge for the case of the fully generalized role path. The problem is that the projection list (used in the universal *for each* quantifier) is part of the model, as is the full role path, but the list of items to numerically quantify must be programmatically determined. It is common for the list of unique items to be the same as the remaining variables not under universal quantification, but this does not hold when additional conditions are applied to the role path. For example, we could add the many-to-many fact type Building has been certified by Inspector to the model and add this as a condition in the join path. We now want "**for each** Building **and** RoomNr, **there is at most one** Room **such that that** Room is in **that** Building **that** has been certified by **some** Inspector **and that** Room has **that** RoomNr." In this case the number of inspectors is irrelevant, so using "**at most one** Room **and** Inspector" as the numerically quantified list would be incorrect.

The algorithm used to determine the numerically quantified variable list takes advantage of the hierarchical nature of the role path structure, proceeding as follows:

1. Starting from the projected nodes, recursively step towards the top of the role path, record all passed nodes, and add all nodes that form a join to another fact type to the list of variables to be numerically quantified. This means that variables in *n*-ary fact types that are not directly joined to other fact types are not in the list.
2. If any node is passed—including those not added to the list—that is correlated with another node in the tree, then repeat the recursive algorithm beginning at each of the correlated nodes. Repeat until the top of the role path has been reached in all cases.
3. Provide a consistent order for the numerically quantified nodes determined by 1 and 2 by doing a depth-first walk from the top of the role path, ordering nodes in the order in which they are reached.

The point of this algorithm is to discover the nodes that must be unique, while not touching nodes (or the corresponding variables) from the conditional branches. The algorithm also covers cases where projected nodes are above or below each other in the path. For example, if our path starts at Building and steps over Room to RoomNr (instead of the default path that starts at Room and splits to Building and RoomNr), then this algorithm still identifies Room as the numerically quantified variable. This algorithm also correctly identifies variables for uniqueness constraints across longer role paths.

5 Conclusion

This paper extended our previous work on verbalization of ORM models, noting various techniques for avoiding ambiguity, including the use of expanded logic-based verbalizations to deal with problematic cases arising from the presence of adverbs in predicate readings. We also provided improved general verbalization patterns for some common kinds of ORM constraints, and discussed an algorithm for verbalizing external uniqueness and external frequency constraints over role combinations projected from join paths of arbitrary complexity.

We have implemented in NORMA all of the verbalization work described in this paper, as well as many other new verbalization patterns, with the sole exception of the expanded logic-based verbalization option, which we plan to soon support. Other research plans in this area include refining the verbalization of other constraints and derivation rules, fully implementing FORML as a language for inputting and querying ORM models, and providing verbalization in languages other than English.

Acknowledgement. Our verbalization research has benefited from reviewing linguistics publications suggested by our colleague Kurt Stirewalt, who also encouraged more use of logical forms.

References

1. Allen, J.: Natural Language Understanding, 2nd edn. Benjamin/Cummings (1994)
2. Bakema, G., Zwart, J., van der Lek, H.: Fully Communication Oriented Information Modelling. Ten Hagen Stam (2000)

3. Chen, P.P.: The entity-relationship model—towards a unified view of data. ACM Transactions on Database Systems 1(1), 9–36 (1976), http://csc.lsu.edu/news/erd.pdf
4. Curland, M., Halpin, T.: The NORMA Software Tool for ORM 2. In: Soffer, P., Proper, E. (eds.) CAiSE Forum 2010. LNBIP, vol. 72, pp. 190–204. Springer, Heidelberg (2011)
5. Curland, M., Halpin, T., Stirewalt, K.: A Role Calculus for ORM. In: Meersman, R., Herrero, P., Dillon, T. (eds.) OTM 2009 Workshops. LNCS, vol. 5872, pp. 692–703. Springer, Heidelberg (2009)
6. Halpin, T.: A Logical Analysis of Information Systems: static aspects of the data-oriented perspective. Doctoral dissertation, University of Queensland (1989), http://www.orm.net/Halpin_PhDthesis.pdf
7. Halpin, T.: Business Rule Verbalization. In: Doroshenko, A., Halpin, T., Liddle, S., Mayr, H. (eds.) Information Systems Technology and its Applications, Proc. ISTA-2004. Lec. Notes in Informatics, vol. P-48, pp. 39–52 (2004)
8. Halpin, T.: ORM 2. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2005. LNCS, vol. 3762, pp. 676–687. Springer, Heidelberg (2005)
9. Halpin, T.: Modality of Business Rules. In: Siau, K. (ed.) Research Issues in Systems Analysis and Design, Databases and Software Development, pp. 206–226. IGI Publishing, Hershey (2007)
10. Halpin, T.: Object-Role Modeling: Principles and Benefits. International Journal of Information Systems Modeling and Design 1(1), 32–54 (2010)
11. Halpin, T.: Fact-Orientation and Conceptual Logic. In: Proc. 15th International EDOC Conference, pp. 14–19. IEEE Computer Society, Helsinki (2011)
12. Halpin, T., Curland, M.: Automated Verbalization for ORM 2. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4278, pp. 1181–1190. Springer, Heidelberg (2006)
13. Halpin, T., Curland, M., Stirewalt, K., Viswanath, N., McGill, M., Beck, S.: Mapping ORM to Datalog: An Overview. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6428, pp. 504–513. Springer, Heidelberg (2010)
14. Halpin, T., Curland, M.: Enriched Support for Ring Constraints. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM-WS 2011. LNCS, vol. 7046, pp. 309–318. Springer, Heidelberg (2011)
15. Halpin, T., Harding, J.: Automated support for verbalization of conceptual schemas. In: Brinkkemper, S., Harmsen, F. (eds.) Proc. 4th Workshop on Next Generation CASE Tools, Univ. Twente Memoranda Informatica 93-32, pp. 151–161 (1993)
16. Halpin, T., Morgan, T.: Information Modeling and Relational Databases, 2nd edn. Morgan Kaufmann, San Francisco (2008)
17. Halpin, T., Wijbenga, J.P.: FORML 2. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) BPMDS 2010 and EMMSAD 2010. LNBIP, vol. 50, pp. 247–260. Springer, Heidelberg (2010)
18. ter Hofstede, A., Proper, H., van der Weide, T.: Formal definition of a conceptual language for the description and manipulation of information models. Information Systems 18(7), 489–523 (1993)
19. Larson, R., Segal, G.: Knowledge of Meaning. MIT Press (1995)
20. Object Management Group: UML OCL 2.0 Specification (2005), http://www.omg.org/docs/ptc/05-06-06.pdf
21. Object Management Group: Unified Modeling Language Specification, version 2.4.1 (2011), http://www.omg.org/spec/UML/2.4.1/
22. Sowa, J.: Common Logic Controlled English (2004), http://www.jfsowa.com/clce/specs.htm
23. Wintraecken, J.: The NIAM Information Analysis Method: Theory and Practice. Kluwer, Deventer (1990)