Objectification

Terry Halpin

Neumont University
Salt Lake City, Utah, USA.
e-mail: terry@neumont.edu

Abstract: Some information modeling approaches allow instances of relationships to be treated as entities in their own right. In the Unified Modeling Language (UML), this is called "reification", and is mediated by association classes. In Object-Role Modeling (ORM), this is called "objectification" or "nesting". While this modeling option is rarely supported by industrial versions of Entity-Relationship Modeling (ER), some academic ER versions do support it. Objectification is related to the linguistic activity of nominalization, of which two flavors may be distinguished: situational; and propositional. In practice, objectification is prone to misuse, and some modeling approaches provide incomplete or flawed support for it. This paper analyzes objectification in-depth, shedding new light on its fundamental nature, and providing practical guidelines on using objectification to model information. Because of its richer semantics, the main graphic notation used is that of ORM. However, the main ideas are relevant to UML and ER as well.

1 Introduction

In this paper, the terms "relationship type", "association", and "fact type" all denote typed predicates (e.g. Country plays Sport). In many business domains, it is natural to think of certain relationship instances as objects about which we wish to talk. For example, Australia's playing of cricket is rated world class. In Object-Role Modeling (*ORM*) dialects, this process of making an object out of a relationship is called "*objectification*" or "*nesting*" [13, 14, 7, 24, 2]. In the Unified Modeling Language (*UML*), this modeling technique is often called "*reification*", and is mediated by means of association classes [27, 28, 30]. Although industrial versions of Entity-Relationship Modeling (*ER*) typically do not support this modeling option [14, ch. 8; 16], in principle they could be extended to do so, and some academic versions of ER do provide limited support for it (e.g. [3]). As an example of partial support, some ER versions allow objectified relationships to have attributes but not to play in other relationships.

In practice, objectification needs to be used judiciously, as its misuse can lead to implementation anomalies, and those modeling approaches that do permit objectification often provide only incomplete or even flawed support for it. This paper provides an in-depth analysis of the modeling activity of objectification, shedding new light on its fundamental nature, and providing practical guidelines on how to use the technique when modeling information. Because of its richer semantics, the main graphic notation used is that of ORM 2 (the latest generation of ORM), with some examples being recast in UML. However, the main ideas are also relevant to extended ER.

Objectification is related to the linguistic activity of *nominalization*. Section 2 distinguishes between situational and propositional nominalization, and argues that objectification in information models typically corresponds to situational nominalization. Section 3 explains situational nominalization of binary and longer facts in terms of equivalences and composite reference schemes. Section 4 extends this treatment to unary facts, and discusses other issues for the objectification of unaries. Section 5 considers what restrictions (if any) should be placed on uniqueness constraints over associations that are to be objectified, and proposes heuristics to help make such choices. Section 6 discusses modeling support to cater for facts or business rules that involve propositional nominalization or communication acts. Section 7 summarizes the main results, suggests topics for future research, and lists references.

2 Two kinds of nominalization

This paper treats nominalization as the recasting of a declarative sentence as a noun phrase morphologically related to a verb in the original sentence. Declarative sentences may be nominalized in various ways. One way uses a gerund (verbal noun) derived from the original verb or verb phrase. For example, "Elvis *sang* the song 'Hound Dog'" may be nominalized as 'Elvis's *singing* of the song 'Hound Dog'". Another way uses a pronoun or description to refer back to the original (e.g. "*that* Elvis sang the song 'Hound Dog", or "*the fact that* Elvis sang the song 'Hound Dog").

In philosophy, it is usual to interpret the resulting nominalizations as naming either corresponding *states of affairs* or corresponding *propositions* [1]. In linguistics, states of affairs are sometimes distinguished into events and situations [10]). For information modeling, we adopt the philosophical approach, ignoring finer linguistic distinctions, and thus treat nominalizations as either *situational* (referring to a state of affairs or situation in the world or business domain being modeled) or *propositional* (referring to a proposition). We treat events (instantaneous) and activities (of short or long duration) as special cases of a state of affairs.

The relationships between states of affairs, propositions, sentences, and communication acts have long been matters of philosophical dispute [9], with no definitive agreement on these issues. At one extreme, states of affairs and propositions are argued to be identical. Some view logic as essentially concerned with connecting sentences to states of affairs (*Sachverhalte*) [31], while others view its focus to be propositions as abstract structures. Our viewpoint on some of these issues is pragmatically motivated by the need to model information systems, and is now summarized.

We define a proposition as that which is asserted when a sentence is uttered or inscribed. A proposition (e.g. Elvis sang Hound Dog) must be true or false (and thus is a truth-bearer). Intuitively it seems wrong to say that a state of affairs (e.g. Elvis's singing of Hound Dog) is true or false. Rather, a state of affairs is actual (occurs or exists in the actual world) or not. A state of affairs may be possible or impossible. Some possible states of affairs may be actual (occur in the actual world). States of affairs are thus truth-makers: true propositions are about actual states of affairs. As in the correspondence theory of truth, we treat the relationship between propositions and states of affairs as one of *correspondence* rather than identity.

Although natural language may be ambiguous as to what a given usage of a nominalization phrase denotes (a state of affairs or a proposition), the intended meaning can usually be determined from the context of the nominalization use (i.e. the logical predicate applied to talk about it). In the below examples, the first three uses of the demonstrative pronoun “that” result in propositional nominalization. In the final example, “that” is used in combination with the gerund “snowing” to refer a state of affairs (propositions aren’t beautiful). In the previous two sentences, “snowing’ is a present participle, not a gerund. For further discussion of related issues, see [10, 23].

Elvis sang the song ‘Hound Dog’.	-- original proposition
Elvis’s *singing* of the song ‘Hound Dog’ is popular.	-- actual state of affairs
That Elvis sang the song ‘Hound Dog’ is well known.	-- true proposition
That Elvis sang the song ‘Hound Dog’ is a false belief.	-- false proposition

It’s snowing outside.	-- original proposition
It’s true *that* it’s snowing outside.	-- proposition
That snowing is beautiful.	-- state of affairs

Object-Role Modeling is also called *fact-oriented modeling*, because it models all the information in the business domain directly as “facts”, using logical predicates, rather than introducing attributes. For example, the fact that Governor Arnold Schwarzenegger smokes may be declared by applying the unary smokes predicate to the governor, rather than assigning “true” to a Boolean isSmoker attribute of the governor (as in UML). States of affairs may be actual or not, and propositions may be true or false. In ordinary speech, “fact” often means a true proposition, but when modeling information in ORM, the term “*fact*” means “*proposition taken to be true*” in the sense of *epistemic commitment* [26, p. 254]. Model facts (committed propositions) are treated by the business as actual facts (true propositions) even if they might not be known with certainty by the business to be true. In the rest of this paper, the terms “fact” (i.e. fact instance) and “fact type” should be understood in this sense.

As a typical case of objectification in information modeling, Fig. 1(a) displays a simple model in the graphic notation of ORM 2 (the latest version of ORM). *Object types* (e.g. Country) are depicted as named, soft rectangles (earlier versions of ORM used ellipses instead). A logical *predicate* is depicted as a named sequence of *role* boxes, each of which is connected by a line segment to the object type whose instances may play that role. The combination of a predicate and its object types is a *fact type*, which is the only data structure in ORM.

If an entity type has a simple, preferred reference scheme, this may be abbreviated by a reference mode in parentheses. In this example, countries are identified by country codes, based on the injective (1:1 into) fact type Country has CountryCode, whose explicit display here is suppressed and replaced by the parenthesized reference mode (Code) that simply provides a compact view of the underlying fact type.

Here the fact type Country plays Sport is objectified as the object type Playing, which itself plays in another fact type Playing is at Rank. The latter fact type is said to be nested, as it nests another fact type inside it. The exclamation mark “!” appended to “Playing” indicates that Playing is *independent*, so instances of Playing may exist without participating in other fact types. This is consistent with the optional nature of the first role of Playing is at Rank. Gerunds are often used to verbalize objectifications in both ORM and the KISS method [25].

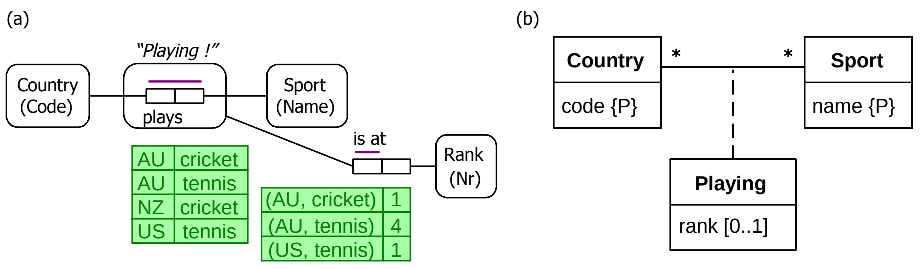


Fig. 1. Objectification of Country plays Sport as Playing in (a) ORM and (b) UML notation

In ORM 2, a bar spanning one or more roles indicates a *uniqueness constraint* over those roles (previously, ORM added arrow tips to the bars). Each role may be populated by a column of object instances, displayed in a *fact table* besides its fact type, as shown. A uniqueness constraint over just a single role ensures that each entry in its fact role column must be unique. In the fact table for Playing is at Rank, the entries for Playing are unique, but some entries for Rank appear more than once, thus illustrating the *n*:1 nature of this fact type. A uniqueness constraint over multiple roles applies to the combination of those roles. In the fact table for Country plays Sport, the entries for the whole row are unique, but entries for Country and Sport may appear on more than one row. Thus illustrates both the uniqueness over the role pair (the table contains a set of facts, not a bag of facts) and the *m:n* nature of this fact type.

Fig. 1(b) depicts the example in UML. *Classes* are depicted as named rectangles, and *associations* as optionally named line segments with their *association roles* (association ends) connected to the classes whose object instances may play those roles. By default, association ends have role names the same as their classes (renaming may be required to disambiguate). UML encodes facts using either associations or attributes. The ORM fact type Country plays Sport is modeled by the association between Country and Sport, which is reified into the *association class* Playing. A "*" indicates a *multiplicity* of 0 or more, so the Playing association is *m:n*. UML treats the association class Playing as identical to the association, and permits only one name for it, so excludes linguistic nominalization. The fact type Playing is at Rank is represented as an optional attribute ([0..1] denotes a *multiplicity* of 0 or 1) on the association class Playing.

Now consider the question: *are the objects resulting from objectification identical to the relationships that they objectify?* In earlier work, we discussed two alternative ORM metamodels, allowing this question to be answered Yes or No [6]. The UML metamodel answers Yes to this question, by treating AssociationClass as a subclass of both Association and Class [27]. Since relationships are typically formalized in terms of propositions, this affirmative choice may be appropriate for propositional nominalization. However, we believe that the objectification process used in modeling information systems is typically situational nominalization, and for such cases we answer this question in the negative, treating fact instances and the object instances resulting from their objectification as non-identical. An intuitive argument for this position follows, based on the model in Fig. 1.

The relationship instance expressed by the sentence: "Australia plays Cricket" is clearly a proposition, which is either true or false. Now consider the object described by the definite description: "The Playing by Australia of Cricket", or more strictly

"The Playing by the Country that has CountryCode 'AU' of the Sport named 'Cricket'". Clearly, this Playing object is a state of affairs (e.g. an activity). It makes sense to say that Australia's playing of cricket is at rank 1, but it makes no sense to say that Australia's playing of cricket is true or false. So the Playing instance (The Playing by Australia of Cricket) is ontologically distinct from the fact/relationship that Australia plays Cricket. Our experience suggests this is typical for objectification examples in information models. In this case, "objectified relationships" are in 1:1 correspondence with the relationships they objectify, but are not identical to those relationships. Compare this with first order logic, where predicate formulae are often tested for equivalence (≡) but not identity (=). Terms or individuals may be identical, but not equivalent.

In information models, one may encounter propositional nominalizations, where the noun phrase refers to a proposition (e.g. [the fact] that Australia plays cricket is well known). A related though different case is where the noun phrase refers to a communication act (e.g. the assertion that Australia plays cricket was made by Don Bradman). We delay discussion of such cases till Section 6.

3 Objectification, and Composite Reference Schemes

Years ago, we formalized ORM in first-order logic (plus some mathematics) [11]. That analysis treated facts as distinct from the objects resulting from objectification, which were formalized in terms of (typically unnamed) ordered pairs; it also assumed that the facts being objectified are not unary, and that each objectified fact type has only one uniqueness constraint, and this spans all its roles. The formalization of objectification outlined in this paper differs in several ways: it makes no use of ordered pairs, instead relying on intuitive equivalences that may be visualized graphically; it supports objectification of unary predicates and predicates with non-spanning uniqueness constraints; and it supports navigation between facts and their objectifications. This section sketches the main ideas, focusing on binary or longer facts with spanning uniqueness constraints. Later sections discuss objectification of facts that either are unary or have non-spanning uniqueness constraints.

To facilitate high level declaration of business rules [18] and queries [4] on information models that use objectification, we include (implicitly or explicitly) *link fact types*, that link or relate the objectification result to the objects in the relationship that has been objectified. For example, the definite description "**The** Playing **that:** is by **the** Country **that** has CountryCode 'AU'; **and** is of **the** Sport **that** has SportName 'Cricket'" makes use of the linking fact types Playing is by Country and Playing is of Sport (Fig. 2). Such descriptions use ORM2's formal, textual language, and assume its default algorithms for translating between implicit (e.g. reference mode) and explicit (e.g. fact type) readings. The large dots attached to role links depict *mandatory role constraints* (each instance of Playing must play both the linking roles). By default, predicates are read left-to-right and top-down; prepending "<<" to a predicate reading reverses the reading order. The *external uniqueness constraint* depicted as a circled uniqueness bar indicates that each (Country, Sport) pair projected from the attached roles relates to at most one Playing object. Previously ORM used a circled "u" for this kind of constraint. Link fact types have long been used for schema navigation in ORM dialects, including LISA-D [24].

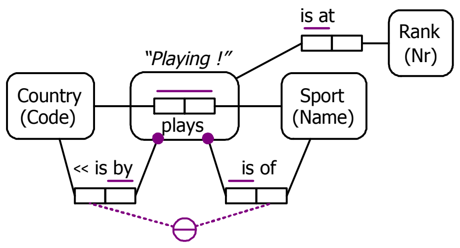
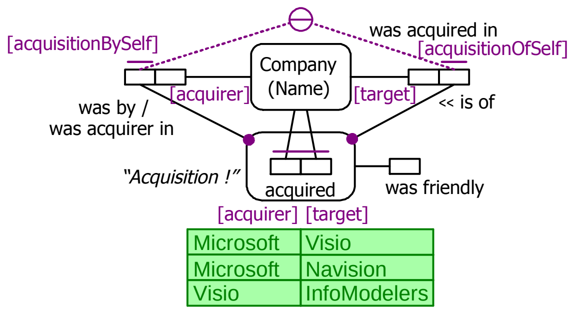


Fig. 2. Objectification in ORM uses linking fact types for relational navigation

If the modeler does not supply readings for the link predicates, default predicate readings are assigned, such as "involves", appended by numbers if needed to distinguish linking fact types that link to the same object type. Fig. 3 adds inverse predicate readings, role names (enclosed in square brackets), and a sample population to the acquisition schema. Display of such model elements on screen and in print may be toggled on/off. The role names (acquirer, target) on the acquisition fact type provide role names for the Company roles in the link fact types—the exact correspondence is derivable if we note the voice (active/passive) of the acquisition predicate reading(s).



[acquirer]	[target]
Microsoft	Visio
Microsoft	Navision
Visio	InfoModelers

Fig. 3. Adding inverse predicate readings and role names supports full navigation

ORM schemas may be *navigated* in *relational-style* (using predicate names) or *attribute-style* (using role names), or a *mixture* of both. From the company Visio we may navigate via the left link to its acquisition of InfoModelers, or via the right link to its acquisition by Microsoft. Navigating via the left link, the schema path may be verbalized in relational style as "Company **that** was acquirer in Acquisition"; navigating via the right link we have "Company **that** was acquired in Acquisition". Here the pronoun "that" performs a conceptual join. Each of the above expressions is a path specification, not a projection on a path. To project on Company and/or Acquisition, we add a projection indicator (e.g. "✓") to the object type occurrence(s) on which we wish to project [4].

To navigate from Acquisition to company, the link paths may be verbalized in relational style as: Acquisition **that** was by Company (navigation via left link); Acquisition **that** is of Company (navigation via right link). Role paths may also be specified in attribute-style, using role names for "attributes". To navigate from Company to Acquisition we have two options: Company.acquisitionBySelf (navigation via left link); Company.acquisitionOfSelf (navigation via right link). To navigate from Acquisition to Company we have two options: Acquisition.acquirer (navigation via left link); Acquisition.target (via right link).

Although such expressions may be used to specify projections, here they simply indicate a path obtained by jumping from an object type to one of its far roles. If the dot notation is replaced by "of-notation", the component order is reversed (e.g. "Company.acquisitionBySelf" becomes "acquisitionBySelf **of** Company). Role path decarations may also mix relational and attribute styles (e.g. Company.acquisitionBySelf **that** occurred on Date).

Fig. 2 is best understood as an abbreviation of Fig. 4(a). Playing is a normal object type with linking fact types to Country and Sport. Playing has a *composite reference scheme* since the external and internal uniqueness, and mandatory constraints on the link fact types ensure an injection (1:1-into mapping) from Playing to (Country, Sport) pairs. This is true even if we add a simple reference scheme for Playing (e.g. PlayingNr).

When an external uniqueness constraint provides a reference scheme, a role sequence obtained by projecting once over each role spanned by that constraint is said to be a *reference projection* for that reference scheme. The order in which the roles are projected is recorded, and its display may be toggled on/off. In Fig. 4(b) the annotation (1.1, 1.2) indicates a role projection formed by projecting respectively on the left and right roles of the fact type Country plays Sport. The annotation (2.1, 2.2) indicates the reference projection for Playing that is formed by projecting respectively on the link roles played by Country and Sport. Role sequence annotations visually disambiguate those rare cases where the role sequences are otherwise ambiguous.

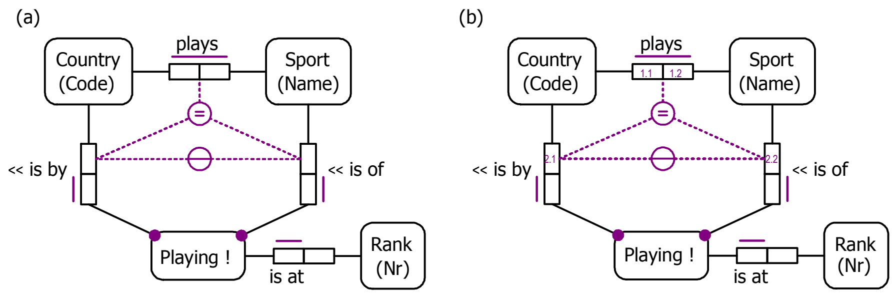


Fig. 4. Explication of the objectification in Fig. 2 of Country plays Sport as Playing

The *equality constraint* depicted by a circled "=" indicates that the (Country, Sport) pairs in the population of the Country plays Sport fact type must be identical to the population of the (Country, Sport) pairs projected from the Country and Sport roles in the join path Playing is by Country **and** is of Sport. In ORM, a set-comparison constraint (subset, equality, or exclusion constraint) applies to two or more sequences of one or more roles. A dotted line connecting a set-comparison constraint to a junction point of two roles includes both the roles in the relevant argument for the constraint. A similar analysis applies to the objectification of ternary and longer facts. For example, we might objectify Country plays Sport in Year as Playing using a third link fact type Playing is in Year whose year role adds a third component to the reference scheme for Playing.

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. This equality constraint may be formalized as an *equivalence*. For our Fig. 4 example, this equivalence might be introduced to the model in three ways: (1) start with the fact type Country plays Sport, and then objectify it as Playing; (2) start with the fact types Playing is by Country and Playing is of Sport, then define Country plays Sport as a fully derived fact type in terms of them; (3) start with the fact types Country plays Sport, Playing is by Country, and Playing is of Sport, then assert the equality constraint between them. These ways may be formalized by the following equivalences: (E1) $\forall x$ [Playing x $\equiv$ $\exists y$:Country $\exists z$:Sport (x is by y & x is of z & y plays z)]; (E2) $\forall x$:Country $\forall y$:Sport [x plays y $\equiv$ $\exists z$:Playing (z is by x & z is of y)]; (E3) $\forall x$:Country $\forall y$:Sport $\forall z$:Playing [x plays y $\equiv$ (z is by x & z is of y)].

ORM 2 includes a formal, high level textual language for declaring its graphical and other business rules (e.g. E2 may be rendered as: Country plays Sport **iff some** Playing is by Country **and** is of Sport). Regardless of which way is used, the model fragment is internally stored in terms of the structure in Fig. 4, and the same mapping procedure is used to transform to the chosen implementation (e.g. a relational database schema).

4 Objectification of unary facts

UML provides no direct support for unary relationships, instead modeling them in terms of attributes or subclasses. ORM supports unary relationships, but typically forbad their objectification. For ORM 2, we removed this restriction, by extending the previous analysis to objectified types with simple reference schemes. Consider the unary fact: The President named ‘Abraham Lincoln’ died. We may objectify this event using the nominalization “that death”, and declare the following additional fact: That death occurred in the Country with country code ‘US’. This natural way of communicating may be supported in a similar way to objectification of non-unary facts. An ORM 2 model for this situation is shown in Fig. 5. Small, sample populations are included for the object types and fact types. Here the unary fact type President died is objectified by the object type Death. If desired, the death entries in the fact table for Death occurred in Country may be expanded by prepending “the death of”.

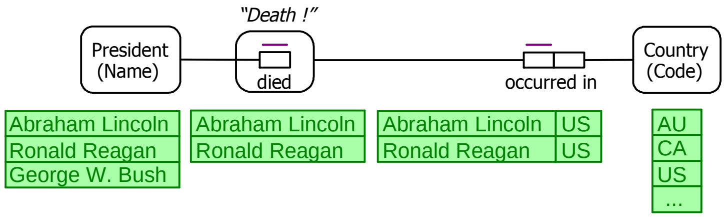


Fig. 5. Objectification of unary facts is allowed in ORM 2

We interpret this unary objectification using the expanded schema shown in Fig. 6. Here, Death is a normal entity type, with a simple reference scheme provided by its injective relationship to President (e.g. Abraham Lincoln’s death may be referenced by the definite description “**The** Death **that** is of **the** President **who** has **the** PresidentName ‘Abraham Lincoln’”). Our previous analysis of objectification may be generalized to include unaries by removing the arity restriction and the composite reference requirement. Hence, *the result of objectifying a relationship type may be viewed as an entity type that has a reference scheme whose reference projection bears an equality constraint to the fact type being objectified*.

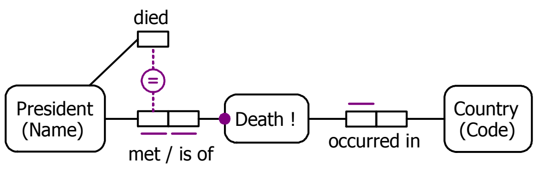


Fig. 6. Objectification of unaries may be explicated as shown

This interpretation does not assume that the Death is of President relationship provides the only, or even primary way of referring to deaths (e.g. we may introduce a death number as an alternative way to reference deaths). ORM 2 allows a reference scheme to be designated as preferred (not the same as primary) if the business treats it as so.

The ORM version known as Fully Communication Oriented Information Modeling (FCO-IM) [2] also supports objectification of unaries, but in a very different manner. To support existential facts such as "**There is a** Country **that** has **the** CountryCode 'AU'", we introduced to ORM the notion of independent entity types (initially called "lazy" entity types) [12]. The FCO-IM approach soon after introduced objectification of unaries to provide an alternative way of supporting existential facts, and to allow models where all base objects are lexical in nature [2]. With this approach, an entity (non-lexical object) is an objectification of a role played by a value (lexical object). In Fig. 7 for example, the entity type Country is derived by objectifying the unary fact type CountryCode refers to a country. While this approach encourages use of natural reference schemes in modeling, and has tool support, we personally find it unintuitive (e.g. it seems to conflate reference with referent), and awkward in dealing with practical modeling issues such as multiple inheritance, context-dependent reference schemes, and changes to reference schemes.

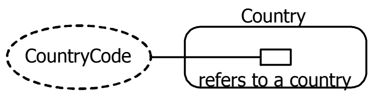


Fig. 7. In FCO-IM, non-lexical types are objectifications of roles of lexical object types

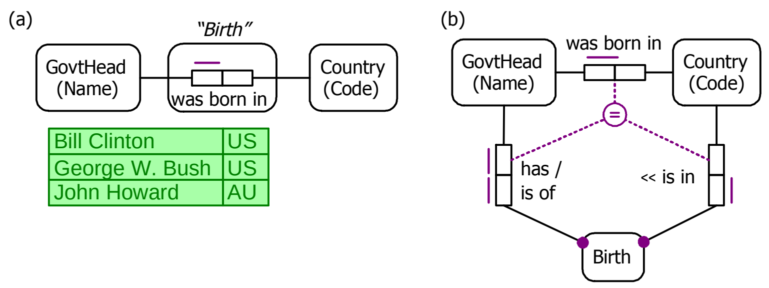
5 Objectification of Fact Types with Non-spanning Uniqueness

Previous versions of ORM allow an association to be objectified only if either it has just one uniqueness constraint, and this spans all its roles, or it is a binary 1:1 association. This restriction forbids the following two kinds of associations to be objectified:

(1) An *n*:1 (or 1:*n*) binary association;
(2) A ternary or longer association whose longest uniqueness constraint spans exactly n-1 roles.

We exclude any *n*-ary association whose longest uniqueness constraint spans fewer than *n*-1 roles, because such an association is compound rather than elementary. Both UML and ER versions that support objectification allow cases (1) and (2) to be objectified. The rest of this section briefly summarizes why ORM 2 has been modified to do likewise, though with modeling guidelines. For a detailed discussion concerning this relaxation, with examples using the ORM 1 and UML notations, see [15].

Fig. 8(a) depicts in ORM 2 the objectification of the *n*:1 fact type GovtHead was born in Country as Birth, together with a sample population. Fig. 8(b) shows how the schema is interpreted. The uniqueness constraint on the "has" role of GovtHead implies, and hence removes the need for, an explicit external uniqueness constraint. The equality constraint may be formalized as an equivalence. The expanded interpretation avoids denormalization when adding other facts or mapping to implementation structures. For example, adding or mapping the fact type Birth was on Date does not require details about birth countries.



Bill Clinton	US
George W. Bush	US
John Howard	AU

Fig. 8. Objectification of an *n*:1 fact type

Mandatory constraints are required on each role played by the objectified type in the link fact types. For example, if it is optional for the birth country to be known for a birth, the fact type GovtHead was born in Country may be defined in terms of the other fact types (by default, a conceptual inner join is performed on the Birth roles), but it does not allow Birth to be defined in terms of GovtHead was born in Country (whose instances always include a country).

As discussed in [15], objectification of *n*:1 associations typically portrays the business domain in an unnecessarily complicated way (why introduce birth countries in order to talk about births?), and may add overhead to certain kinds of model changes. However, such objectifications may better depict the semantic affinity between fact types attached to the objectified type, and they simplify model evolution for those cases where the uniqueness constraint on the objectified association changes over time (e.g. from an *n*:1 to an *m:n* pattern).

The second case is objectification of *n*-ary associations ($n > 2$) whose longest uniqueness constraint spans *n*-1 roles. The *n*-ary association may have overlapping uniqueness constraints. For example, the ternary fact type Country in Sport has Rank may have a uniqueness constraint over its first two roles, and another uniqueness constraint over its last two roles. In such cases, objectifying part of the association based on the roles played by one of the uniqueness constraints typically makes the model harder to understand, and may force an arbitrary decision on which uniqueness constraint to use as the basis for a spanning objectification [15]. In rare cases, it is also possible that the uniqueness constraint pattern on the *n*-ary association may change over time (e.g. to a spanning uniqueness constraint), and in such cases semantic stability may be enhanced by allowing nesting of the original association.

These considerations lead to the following modeling heuristic. A fact type may be objectified only if: (a) it has only a spanning uniqueness constraint; or (b) its uniqueness constraint pattern is likely to evolve over time (e.g. from n:1 to m:n, or m:n:1 to m:n:p); or (c) it has at least two uniqueness constraints spanning n-1 roles(n > 1), and there is no obvious choice as to which of the n-1 role uniqueness constraints is the best basis for a smaller objectification based on a spanning uniqueness constraint; or (d) the objectification significantly improves the display of semantic affinity between fact types attached to the objectified type.

6 Propositional Nominalization and Communication Acts

So far we have discussed objectification in the sense of situational nominalization, where the referenced object is a state of affairs (event, activity etc.). One may also encounter cases where the referenced object is either a proposition (resulting from propositional nominalization) or a communication act (e.g. an utterance by a speaker). In response to an Object Management Group request for proposal to add a business semantics layer [29], the Business Rules Team submission included examples of propositional nominalization as business rules. For example: If a waiter earns an amount of money as a tip from serving a meal, the waiter must report that fact.

While one may interpret this as a case of propositional nominalization (reporting the fact rather than the act), the rule may instead be declared using situational nominalization (reporting the act rather than the fact), as shown in compact form in Fig. 9. If the rule is modified to require reporting after the service is performed, a time limit for reporting must be declared to make the rule operational; in this case, the relevant temporal object type may now be added to the model to cater for the extended rule in an obvious way. For simplicity, we recommend modeling all propositional nominalizations instead by their corresponding situational nominalizations.

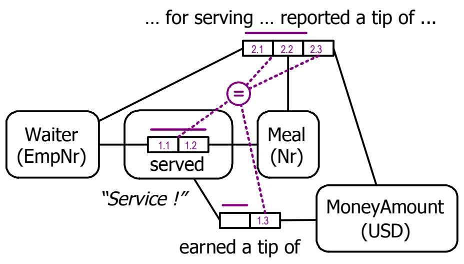


Fig. 9. Propositional nominalization may be replaced by situational nominalization

As regards modeling of *communication acts* [32], when it is of interest to model these acts, they are best modeled directly like any other business domain objects. For example, in a genealogy model we might be interested in not just descriptions of states of affairs, but assertion acts made by researchers about states of affairs. Such a model might include fact types such as: AssertionAct reported Proposition; AssertionAct was made by Researcher with ConfidenceLevel; etc. These comments relate to the information model only. For modeling communication processes, the information model should be supplemented by other kinds of model (e.g. workflow models) that provide a more intuitive and direct way of understanding essential business processes/services. For some initial discussion of how ORM might be extended in this regard, see [8, 22].

7 Conclusion

This paper distinguished two kinds of nominalization (situational and propositional), and argued that objectification used to model information systems may be adequately addressed by situational nominalization alone, where the object referenced by the nominalization is a state of affairs. An underlying theory was then presented that interpreted the objectification of facts of any arity (unary, binary or longer) in terms of normal entity types, their reference schemes, and equality constraints. To cater for objectification over predicates with non-spanning uniqueness constraints, guidelines were proposed to help the modeler decide whether or how to perform the objectification. Finally, it was argued that no additional meta-structures are needed to capture information models for specific business domains that involve propositional nominalization or communication acts.

In previous work, we formalized ORM and worked on the ORM technology currently supported in a Microsoft modeling tool [20]. Currently we are working with a team on the specification of ORM 2 (the next generation of ORM), and an associated open-source modeling tool that supports the refinements to objectification discussed in this paper, as well as many other extensions being added to ORM 2.

Acknowledgement: Our presentation of some ideas has benefited from discussion with Andy Carver (Neumont University) and Don Baisley (Unisys Corporation).

References

1. Audi, R. (ed.) 1999, *The Cambridge Dictionary of Philosophy*, 2nd edition, Cambridge University Press, Cambridge, p. 876.
2. 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.
3. Batini, C., Ceri, S. & Navathe, S. 1992, *Conceptual Database Design*, Benjamin/Cummings, Redwood City.
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. Chen, P. P. 1976, 'The entity-relationship model—towards a unified view of data'. *ACM Transactions on Database Systems*, 1(1), pp. 9–36.
6. 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).
7. 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.
8. Dietz, J. & Halpin, T. 2004, 'Using DEMO and ORM in Concert: A Case Study', *Advanced Topics in Database Research, vol. 3*, ed. K. Siau, Idea Publishing Group, Hershey PA, USA, Ch. XI (pp. 218-36).
9. Gale, R. 1967, 'Propositions, Judgments, Sentences, and Statements', *The Encyclopedia of Philosophy*, ed. P. Edwards, vol. 6, Collier-Macmillan, London, pp. 494-505.

10. Gundell, J., Hegarty, M. & Borthen, K., 'Cognitive Status, Information Structure, and Pronominal Reference to Clausally Introduced Entities. Online at: http://www.coli.uni-sb.de/~korbay/esslli01-wsh/Jolli/Final/gundel-etal.pdf.
11. Halpin, T. 1989, 'A Logical Analysis of Information Systems: static aspects of the data-oriented perspective', doctoral dissertation, University of Queensland.
12. Halpin, T. 1993, 'What is an elementary fact?', *Proc. First NIAM-ISDM Conf.*, eds G.M. Nijssen & J. Sharp, Utrecht, (Sep), 11 pp. Online at http://www.orm.net/pdf/ElemFact.pdf.
13. Halpin, T. 1998, 'ORM/NIAM Object-Role Modeling', *Handbook on Inf. Systems Architectures*, eds P. Bernus, K. Mertins & G. Schmidt, Springer, Berlin, pp. 81-101.
14. Halpin, T. 2001, *Information Modeling and Relational Databases*, Morgan Kaufmann, San Francisco.
15. Halpin, T. 2003, 'Uniqueness Constraints on Objectified Associations', Journal of Conceptual Modeling, October 2003. Online at: http://www.orm.net/pdf/JCM2003Oct.pdf.
16. 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).
17. 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.
18. 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.
19. Halpin, T. A. 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).
20. Halpin, T., Evans, K, Hallock, P. & MacLean, W. 2003, *Database Modeling with Microsoft® Visio for Enterprise Architects*, Morgan Kaufmann, San Francisco.
21. Halpin, T. & Proper, H. 1995, 'Database schema transformation and optimization', *Proc. OOER'95: OO. and ER. Modeling*, Springer LNCS, vol. 1021, pp. 191-203.
22. Halpin, T. & Wagner, G. 2003, 'Modeling Reactive Behavior in ORM'. *Conceptual Modeling – ER2003*, Proc. 22nd ER Conference, Chicago, October 2003, Springer LNCS.
23. Hegarty, M., 'Referential Properties of Factive and Interrogative Complements Indicate their Semantics'. Abstract: http://www.linguistics.berkeley.edu/BLS/abstracts/0113.pdf.
24. 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.
25. Kristen, G. 1994, *Object Orientation – The KISS Method: From Information Architecture to Information System*, Addison Wesley, Reading, MA.
26. Lyons, J. 1995, *Linguistic Semantics: An Introduction*, Cambridge University Press: Cambridge, UK.
27. Object Management Group 2003, *UML 2.0 Infrastructure Specification*. Online: www.omg.org/uml.
28. Object Management Group 2003, *UML 2.0 Superstructure Specification*. Online: www.omg.org/uml.
29. Object Management Group 2003, *Business Semantics of Business Rules RFP*. Online at: http://www.omg.org/cgi-bin/doc?br/2003-6-3.
30. Rumbaugh J., Jacobson, I. & Booch, G. 1999, *The Unified Language Reference Manual*, Addison-Wesley, Reading, MA.
31. Smith, B. 1989, 'Logic and the Sachverhalt', *The Monist*, 72:1, pp. 52-69. Online at: http://ontology.buffalo.edu/smith//articles/logsvh.html.
32. Thomas, J. 1995, Meaning in Interaction: An Introduction to Pragmatics, Longman, London.