

Business Rule Modality

Terry Halpin

Neumont University
Salt Lake City, USA
terry@neumont.edu

Abstract. A business domain is typically constrained by business rules. In practice, these rules often include constraints of different modalities (e.g. alethic and deontic). Alethic rules impose necessities, which cannot, even in principle, be violated by the business. Deontic rules impose obligations, which may be violated, even though they ought not. Conceptual modeling approaches typically confine their specification of rules to alethic rules. This paper discusses one way to model deontic rules, especially those of a static nature. A formalization based on modal operators is provided, and some challenging semantic issues are examined from both logical and pragmatic perspectives. Because of its richer semantics, the main graphic notation used is that of Object-Role Modeling (ORM). However, the main ideas could be adapted for UML and ER as well. A basic implementation of the proposed approach has been prototyped in a tool that supports automated verbalization of both alethic and deontic rules.

1 Introduction

An information system in the wider sense corresponds to a business domain or universe of discourse rather than an automated system. Business domains are constrained by various *business rules*, which specify required or desirable states of affairs or behavior. Business rules may be of different *modalities* (e.g. alethic and deontic). *Alethic* rules impose necessities, which cannot, even in principle, be violated by the business, typically because of some physical or logical law. For example: each employee was born on at most one date; no product is a component of itself. *Deontic* rules impose obligations, which may be violated, even though they ought not. For example: it is obligatory that each employee is married to at most one person; no smoking is permitted in any office. Using “constraint” in a liberal sense to include soft as well as hard constraints, deontic rules may also be called deontic *constraints*.

Various information modeling approaches exist for modeling business domains at a high level, for example Entity-Relationship Modeling (ER) [1], the Unified Modeling Language (UML) [14, 15, 17], and Object-Role Modeling (ORM) [11, 4]. However, these modeling approaches typically confine their specification of constraints to alethic rules. It is important for a business to have a clear understanding of all its rules, including deontic constraints, whether or not the business chooses to enforce these rules, or monitor violations of them, by means of an automated system. In recognition of this need, as well as to facilitate exchange of semantics between businesses, the Object Management Group (OMG) is currently finalizing a proposal to specify a business semantics layer on top of its software-specific layers [16].

The proposal that was accepted by the OMG for finalization is the Semantics of Business Vocabulary and Rules (SBVR) submission. As a contributor to this submission, the author focused on the formal logic underpinnings of SBVR. This paper relates in part to that fragment of his contribution that is concerned with modeling of deontic constraints, especially those of a static nature. Because of its richer semantics, the main graphic notation used is that of ORM 2 (the next generation of Object-Role Modeling). However, the main ideas could be adapted for UML and ER as well.

Section 2 provides a simple overview of the use of modal operators in expressing business constraints of alethic and deontic modalities, and illustrates the automated verbalization of these constraints as implemented in NORMA, an open source ORM 2 tool. Section 3 discusses the formal underpinnings of static, alethic constraints. Section 4 does likewise for static, deontic constraints, and examines some challenging semantic issues from both logical and pragmatic perspectives. Section 5 briefly raises some issues relating to dynamic constraints. Section 6 summarizes the main results, suggests topics for future research, and lists references.

2 Modal Operators and Constraint Verbalization

Business constraint formulations may use any of the basic alethic or deontic modal operators from modal logic¹, as shown in Table 1. These modal operators are treated as proposition-forming operators on propositions (rather than actions). Other equivalent readings may be used in whatever concrete syntax is used to originally declare the rule (e.g. “necessary” might be replaced by “required”, and “obligatory” might be replaced by “ought to be the case”). Derived modal operators may also be used in the surface syntax, but are translated into the basic modal operators plus negation (\sim). For example, “It is impossible that p ” is defined as “It is not possible that p ” ($\sim\Diamond p$), and “It is forbidden that p ” is defined as “It is not permitted that p ” ($Fp \equiv_{df} \sim Pp$).

Table 1. Alethic and deontic modal operators

<i>Alethic</i>		<i>Deontic</i>	
<i>Reading</i>	<i>Symbol</i>	<i>Reading</i>	<i>Symbol</i>
It is necessary that	\Box	It is obligatory that	O
It is possible that	\Diamond	It is permitted that	P

The following *modal negation rules* apply: it is not necessary that \equiv it is possible that not ($\sim\Box p \equiv \Diamond\sim p$); it is not possible that \equiv it is necessary that not ($\sim\Diamond p \equiv \Box\sim p$); it is not obligatory that \equiv it is permitted that it is not the case that ($\sim Op \equiv P\sim p$); it is not permitted that \equiv it is obligatory that it is not the case that ($\sim Pp \equiv O\sim p$). In principle, these rules could be used with double negation to get by with just one alethic modal operator (e.g. $\Diamond p$ could be defined as $\sim\Box\sim p$, and Pp could be defined as $\sim O\sim p$).

ORM is a conceptual modeling approach that models any business domain in terms of objects (entities or values) that play roles in relationships (unary, binary, or

¹ http://en.wikipedia.org/wiki/Modal_logic

longer), also known as facts, relegating the attribute construct merely to derived views, and hence offering greater semantic stability than attribute-based approaches like ER and UML [2, 5, 6, 12]. ORM also has a rich graphic notation for capturing constraints, which ORM tools can transform into implementation code for enforcement. In ORM 2 [10], the latest version of ORM, each constraint has an associated *modality*, determined by the logical modal operator that functions explicitly or implicitly as its main operator. ORM 2 distinguishes between positive, negative, and default verbalizations of constraints [7]. In positive verbalizations, an alethic modality of necessity is often assumed (if no modality is explicitly specified), but may be explicitly prepended. For example, the following static constraint

C1 Each Person was born in at most one Country.

may be explicitly verbalized with an alethic modality thus:

C1' It is necessary that each Person was born in at most one Country.

We interpret this in terms of *possible world semantics*, as introduced by Saul Kripke and other logicians in the 1950s. A proposition is necessarily true if and only if it is true in all possible worlds. With respect to a *static constraint* declared for a given business domain, a possible world corresponds to a state of the fact model that might exist at some point in time. The constraint C1 means that for each state of the fact model, each instance in the population of Person is born in at most one country.

A proposition is possible if and only if it is true in at least one possible world. Impossible propositions are true in no possible world (i.e. false in all possible worlds). In ORM, constraint C1 may be reformulated as the following negative verbalization:

C1'' It is impossible that the same Person was born in more than one Country

In practice, both positive and negative verbalizations are useful for validating constraints with domain experts, especially when illustrated with sample populations that provide satisfying examples or counter-examples respectively.

Many business constraints are deontic rather than alethic in nature. To avoid confusion, when declaring a deontic constraint, the deontic modality should always be explicitly included. Consider the following static, deontic constraint.

C2 It is obligatory that each Person is a husband of at most one Person.

If this rule were instead expressed simply as “Each Person is a husband of at most one Person”, it would not be obvious that a deontic interpretation was intended. The deontic version indicates a condition that *ought* to be satisfied, while recognizing that the condition *might* not be satisfied. Including the obligation operator makes the rule much weaker than a necessity claim, since it allows that there could be some states of the fact model where a person is a husband of more than one wife (excluding same-sex unions from instances of the husband relationship). For such cases of polygamy, it is important to know the facts indicating that the person has multiple wives. Rather than reject this possibility, we allow it and then typically perform an action that is designed to minimize the chance of such a situation arising again (e.g. send a message to inform legal authorities about the situation). In ORM, constraint C2 may be reformulated as the following negative verbalization:

C2' It is forbidden that the same Person is a husband of more than one Person.

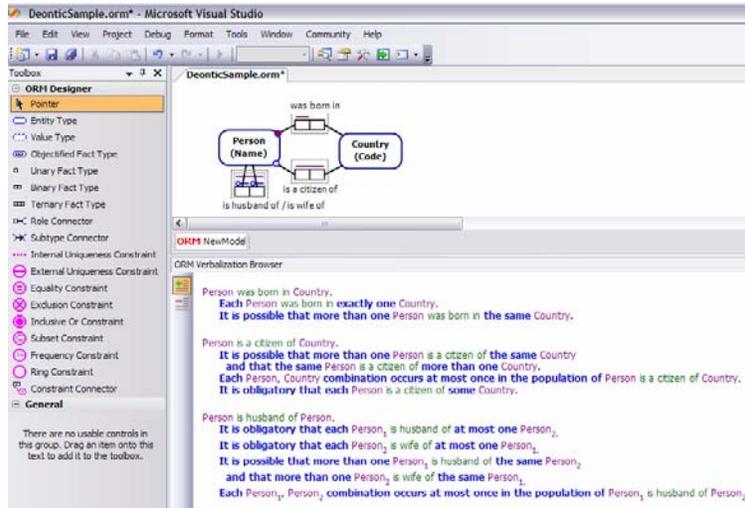


Fig. 1. Screenshot from NORMA, showing positive verbalization of some constraints

Fig. 1 shows a screenshot from NORMA (Neumont ORM Architect), illustrating positive verbalization of some alethic and deontic constraints in ORM 2. *Object types* (e.g. Person, Country) are depicted as named, soft rectangles (ORM 1 used ellipses). A logical *predicate* is depicted as a named sequence of *role* boxes, each 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*—the only data structure in ORM.

A bar spanning one or more roles depicts a *uniqueness constraint* over those roles (e.g. Each Person was born in at most one Country). A constraint over multiple roles applies to the combination of those roles (e.g. the citizenship fact type is many:many). A small “o” (for “obligatory”) at the end of a uniqueness bar indicates the constraint is deontic (e.g. It is obligatory that each Person₁ is husband of at most one Person₂). Subscripts distinguish object variables of the same type. A solid dot on a role indicates a *mandatory constraint* (e.g. Each Person was born in some Country). If the dot is open, the constraint is deontic (e.g. It is obligatory that each Person is a citizen of some Country).

Fig. 2 displays a screenshot from NORMA, illustrating negative verbalization of a deontic uniqueness constraint spanning the first two roles of the ternary fact type: Room at HourSlot was booked for Course. The constraint verbalization (It is forbidden that the same Room at the same HourSlot is booked for more than one Course) uses the deontic $F(\sim P)$ operator. All verbalizations in NORMA are performed automatically via XSLT transforms, and hence may be readily adapted for different native languages. NORMA itself is an open source plug-in to Visual Studio .NET 2005.

In practice, most business rules include only one modal operator, where this is the main operator of the whole rule expression. For these cases, we simply tag the constraint as being of the modality corresponding to its main operator, without committing to any particular modal logic. Apart from this modality tag, some basic modal properties may be used to transform the original high level expression of the rule into a standard logical formulation. These properties include the modal negation rules.

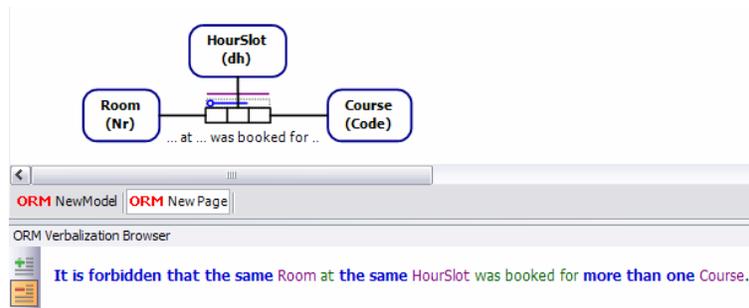


Fig. 2. NORMA screen shot illustrating negative verbalization of a deontic constraint

We also make use of equivalences that allow one to move the modal operator to the front of the formula. For example, suppose the user formulates rule C1 instead as:

For each Person, it is necessary that that Person was born in at most one Country.

The modal operator is now embedded in the scope of a universal quantifier. To transform this rule to a standard logical formulation that classifies the rule as an alethic necessity, we move the modal operator before the universal quantifier, to give:

It is necessary that each Person was born in at most one Country.

For such tasks, we assume that the Barcan formulae and their converses apply, so that \Box and \forall are commutative, as are \Diamond and \exists . In other words: $\forall x\Box Fx \equiv \Box\forall xFx$ and $\exists x\Diamond Fx \equiv \Diamond\exists xFx$. While these commutativity results are valid for all normal, alethic modal logics, some philosophical concerns have been raised about these equivalences (e.g. see sections 4.6-4.8 of [3]). As a deontic example, suppose the user formulates rule C2 instead as:

For each Person, it is obligatory that that Person is a husband of at most one Person.

Using a deontic variant of the Barcan equivalences, we commute the \forall and O operators, thus transforming the rule to the deontic obligation:

It is obligatory that each Person is a husband of at most one Person.

So far, our rule examples have included just one modal operator, which (perhaps after transformation) is the main operator. Ignoring dynamic aspects, we may handle such cases without committing to the formal semantics of any specific modal logic. The only impact of tagging a rule as a necessity or obligation is on rule enforcement. Enforcement of a necessity rule should never allow the rule to be violated. Enforcement of an obligation rule should allow states that do not satisfy the rule condition, and take some remedial action (the tool's default action is to generate a message when an update violates the rule). A business person ought to be able to specify a deontic rule first at a high level, without committing at that time to the precise action to be taken if the condition is not satisfied; the action still needs to be specified later in refining the rule to make it fully operational.

3 Static, Alethic Constraints

Rule formulations may use two alethic modal operators: \Box = it is necessary that; \Diamond = it is possible that. Static constraints are treated as necessities by default, where each state of the fact model corresponds to a possible world. On the fact type Person was born in Country, the constraint “Each Person was born in at most one Country” is equivalent to the logical formula $\forall x:\text{Person} \exists^{0..1}y:\text{Country } x \text{ was born in } y$. This formula is understood to be true for each state of the knowledgebase. Pragmatically, the rule is understood to apply to all future states of the fact model, until the rule is revoked or changed. This understanding may be made explicit by prepending the formula with \Box to yield $\Box\forall x:\text{Person} \exists^{0..1}y:\text{Country } x \text{ was born in } y$. For compliance with Common Logic [13], such formulae could then be treated as irregular expressions, with the modal operator treated as an uninterpreted symbol (e.g. using “[N]” for \Box). However we leave this understanding as implicit, and do *not* commit to any particular modal logic.

For the model theory, we omit the necessity operator from the formula. Instead, we merely tag the rule as a necessity. The implementation impact of the necessity tag is that any attempted change that would cause the model of the business domain to violate the constraint must be dealt with in a way that ensures the constraint is still satisfied (e.g. reject the change, or take some compensatory action).

Typically, the only alethic modal operator in an explicit rule formulation is \Box , at the front of the rule. This common case was covered earlier. If an alethic modal operator is placed later in the rule, we first try to “normalize” it by moving the modal operator to the front, using transformation rules such as the modal negation rules ($\sim\Box p \equiv \Diamond\sim p$; $\sim\Diamond p \equiv \Box\sim p$) and/or the Barcan formulae and their converses ($\forall x\Box\Phi x \equiv \Box\forall x\Phi x$ and $\exists x\Diamond\Phi x \equiv \Diamond\exists x\Phi x$, i.e. \Box and \forall are commutative, as are \Diamond and \exists). For example, the formula “ $\forall x:\text{Person} \Box \exists^{0..1}y:\text{Country } x \text{ was born in } y$ ” (For each Person, it is necessary that that Person was born in at most one Country.) transforms to “ $\Box\forall x:\text{Person} \exists^{0..1}y:\text{Country } x \text{ was born in } y$ ” (It is necessary that each Person was born in at most one Country.). We also allow use of the following equivalences: $\Box\Box p \equiv \Box p$; $\Diamond\Diamond p \equiv \Diamond p$; $\Box\Diamond\Box p \equiv \Box\Diamond p$; $\Diamond\Box\Diamond p \equiv \Diamond\Box p$. These hold in S4, but not in some modal logics, e.g. K or T [3, p. 35].

Though not supported by NORMA, the SBVR proposal allows a single rule with multiple occurrences of modal operators, including nesting of a modal operator in the scope of another modal operator. While this expressibility may be needed to capture some rare but real business rules, it significantly complicates the formal semantics.

In very rare cases, a static rule formula might embed an alethic modality that cannot be eliminated by transformation. For such cases, we could retain the modal operator in the rule formulation and adopt the semantics of a specific modal logic. There are many normal modal logics to choose from (e.g. K, K4, KB, K5, DT, DB, D4, D5, T, Br, S4, S5) as well as non-normal modal logics (e.g. C2, ED2, E2, S0.5, S2, S3). For details on these logics, see [3] (esp. pp. 48, 82). For SBVR, if we retain the embedded alethic operator for such cases, we choose S4 for the formal semantics. Schema evolution and changes to necessity constraints may seem to violate S4, where the accessibility relationship between possible worlds is transitive, but we resolve this by treating such evolution as a metametalevel concern. Alternatively, we may handle such very rare cases by moving the embedded alethic operators down to domain-level predicates (e.g. is necessary), similar to our handling of embedded deontics (see later).

4 Static, Deontic Constraints

Constraint formulations may use the standard deontic modal operators (O = it is obligatory that; P = it is permitted that) as well as F = it is forbidden that (defined as $\sim P$, i.e. “It is not permitted that”). If the rule includes exactly one deontic operator, O , and this is at the front, then the rule may be formalized as Op , where p is a first-order formula tagged as obligatory (rather than necessary). NORMA assigns this tag the following informal semantics: it ought to be the case that p (for all future states of the fact model, until the constraint is revoked or changed). The implementation impact is that it is possible to have a state where the rule’s condition is violated (i.e. unsatisfied), in which case some appropriate action (e.g. messaging) ought to be taken to reduce the chance of future violations. Later work will address rule enforcement, including specification of appropriate actions in response to deontic rule “violations”.

From a model-theoretic perspective, a model is an interpretation where each *non-deontic* formula evaluates to true, and the model is classified as a *permitted model* if the p in each deontic formula (of the form Op) evaluates to true, otherwise the model is a *forbidden model* (though it is still a model). Note that this approach removes any need to assign a truth value to expressions of the form Op .

For example, suppose the fact type `Person is a husband of Person` is declared many to many, but that each role has a deontic uniqueness constraint to indicate that the fact type *ought* to be 1:1 (see Fig. 3). The deontic constraint on the husband role verbalizes as: It is obligatory that each Person is a husband of at most one Person. This formalizes as $O\forall x:\text{Person} \exists^{0..1}y:\text{Person } x \text{ is a husband of } y$, which may be captured by entering the rule without the O , then tagging the rule as deontic. The other deontic constraint (each wife should have at most one husband) may be handled similarly. In this example, the combination of alethic and deontic constraints is consistent, but this is not always the case. For example, the argument (role set) of a deontic uniqueness constraint must be a proper subset of the argument of an alethic uniqueness constraint. For instance, if the marriage predicate is alethically 1:1, then no deontic uniqueness constraint may be added (if something is already necessary, it makes no sense to declare it obligatory).

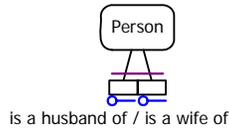


Fig. 3. Deontic constraints obligate the marriage relationship to be 1:1

Some SBVR formulae are illegal in some deontic logics (e.g. iterating modal operators such as OPp is forbidden in von Wright’s deontic logic), and deontic logic itself is “rife with disagreements about what should be the case” [3, p. 173]. If a deontic modal operator is embedded later in the rule formulation, we first try to “normalize” the formula by moving the modal operator to the front, using transformation rules such as $p \supset Oq \equiv O(p \supset q)$ ² or deontic counterparts to the Barcan formulae.

² In contrast to our approach, some versions of deontic logic reject this equivalence on the basis that agent control is restricted to the scope of modal operators.

In some cases, a formula for a static business rule might contain an embedded deontic modality that cannot be eliminated by transformation. In this case, we still allow the business user to express the rule at a high level using such embedded deontic operators, but *where possible* we transform the formula to a first-order formula without modalities by *replacing the modal operators by predicates at the business domain level*. These predicates (e.g. is forbidden) are treated like any other predicate in the domain, except that their names are reserved, and they are given some basic additional formal semantics to capture the deontic modal negation rules: it is not obligatory that \equiv it is permitted that it is not the case that ($\sim Op \equiv P\sim p$); it is not permitted that \equiv it is obligatory that it is not the case that ($\sim Pp \equiv O\sim p$). For example, these rules entail an exclusion constraint between the predicates is forbidden and is permitted.

This latter approach may also be used as an alternative to tagging a rule as deontic, thereby (where possible) moving deontic aspects out of the metamodel and into the business domain model. For example, consider the following rule: *Car rentals ought not be issued to people who are barred drivers at the time the rental was issued*. This deontic rule may be captured by the following ORM derivation rule for the partly derived domain fact type CarRental is forbidden:

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

The fact type Person is a barred driver at Time is derived from other base fact types (Person was barred at Time, Person was unbarred at Time) using the ORM derivation rule:

Person is a barred driver at Time1 **iff**
 Person was barred at a Time2 \leq Time1 **and**
 Person was **not** unbarred at a Time3 **between** Time2 **and** Time1

The deontic rule may be formalized by the first-order formula: $\forall x:\text{CarRental} \forall y:\text{Person} \forall t:\text{Time} [(x \text{ was issued at } t \ \& \ x \text{ was issued to } y \ \& \ y \text{ is a barred driver at } t) \supset x \text{ is forbidden}]$. This schema (see Fig. 4) allows for the possible existence of forbidden car rentals; if desired, some fact types could be added to describe actions (e.g. sending messages) to be taken in reaction to such an event. Full and partial derivation is denoted by “*” and “*” respectively. Note also the exclusion (circled “X”) constraint (nobody can be simultaneously barred and unbarred) and subset constraint. We allow a person to be barred many times before being unbarred.

For other examples illustrating this approach, including use of derivation rules and objectification, see the SBVR submission to the OMG [16]. Our approach to objectification works for those cases where a fact (proposition taken to be true) is being objectified (which covers the usual cases of nominalization [9]), but it does not handle cases where no factual claim is being made of the proposition.

SBVR is intended to cater for rules that embed possibly non-factual propositions. However, there does not appear to be any simple solution to providing explicit, formal semantics for such rules. As a nasty example, consider the following business rule:

It is not permitted that some department adopts a rule that says it is obligatory that each employee of that department is male.

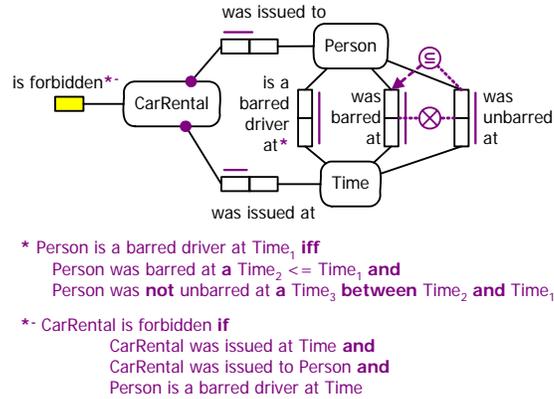


Fig. 4. Forbidding rentals to barred drivers using a domain level predicate

This example includes the mention (rather than use) of an open proposition in the scope of an embedded deontic operator. One possible, though weak, solution is to rely on reserved domain predicates to carry much of the semantics implicitly. For example, the ORM schema in

Fig. 5 uses the special predicates “obligates the actualization of” and “is actual”, as well as an object type “PossibleAllMaleState” which includes all conceivable all-male-states of departments, whether actual or not. The derived fact type PossibleAllMaleState is actual may be defined using the derivation rule:

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

i.e. $\forall x:\text{PossibleAllMaleState} [x \text{ is actual} \equiv \exists y:\text{Department} (x \text{ is of } y \ \& \ \forall z:\text{Person} (z \text{ works for } y \supset z \text{ is male}))]$. The deontic rule may now be captured by the following derivation rule for the partly derived fact type RuleAdoption is forbidden:

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

i.e. $\forall x:\text{RuleAdoption} \ \forall y:\text{Department} \ \forall z:\text{Rule} \ \forall w:\text{PossibleAllMaleState} [(x \text{ is by } y \ \& \ x \text{ is of } z \ \& \ z \text{ obligates the actualization of } w \ \& \ w \text{ is of } y) \supset x \text{ is forbidden}]$

The formalization of the deontic rule works, because the relevant instance of PossibleAllMaleState exists, regardless of whether or not the relevant depart actually is all male. The “obligates the actualization of” and “is actual” predicates embed a lot of semantics, which is left implicit. While the connection between these predicates is left

informal, the derivation rule for PossibleAllMaleState is actual provides enough semantics to enable human readers to understand the intent.

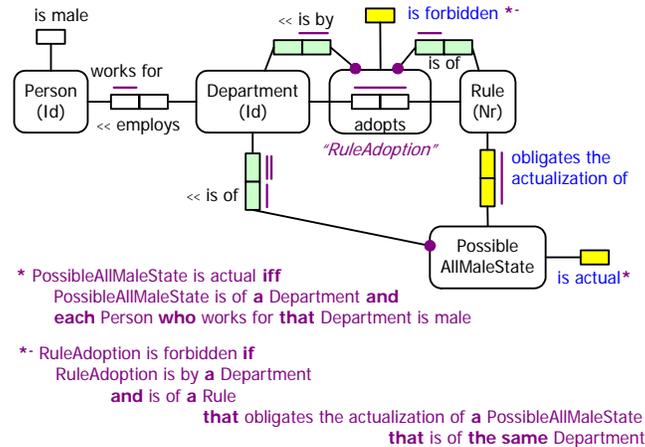


Fig. 5. A complex case involving embedded mention of propositions

Alternatively, we could adopt one of two extremes: (1) treat the rule overall as an uninterpreted sentence, or informal comment, for which humans provide the semantics; (2) translate the semantic formulation directly into higher-order logic, permitting logical formulations (which connote propositions) to be predicated over. The complexity and implementation overhead of option (2) would seem to be very substantial.

We could try to push such cases down to first-order logic by providing the necessary semantic formulation machinery as a predefined package that may be imported into a domain model, and then identifying propositions by means of a structured logical formulation. But that seems unclean, because in order to assign formal semantics to such expressions, we must effectively adopt the higher-order logic proposal mentioned in the previous paragraph.

Support for reification is currently being added by Pat Hayes and Chris Menzel to the IKL language. This support is intended to cater for objectification of propositions that are already being asserted as facts (i.e. propositions being used), as well as propositions for which no factual claim is made (i.e. propositions being mentioned), while still retaining a first-order approach. When available, this may offer a better solution for the problem under consideration.

5 Dynamic Constraints

Dynamic constraints apply restrictions on possible transitions between business states. The constraint may simply compare one state to the next (e.g. salaries should never

decrease); or the constraint may compare states separated by a given period (e.g. invoices ought to be paid within 30 days of being issued). The invoice rule might be formally expressed in a high level rules language thus, assuming the fact types Invoice was issued on Date and Invoice is paid on Date are included in the conceptual schema:

For each Invoice, if that Invoice was issued on Date₁ then it is obligatory that that Invoice is paid on Date₂ where Date₂ <= Date₁ + 30 days

This might now be normalized to the following formulation, moving the deontic operator to the front:

It is obligatory that each Invoice that was issued on Date₁ is paid on Date₂ where Date₂ <= Date₁ + 30 days

There are two issues here. First, what rules did we rely on to license the transformation of the rule? It would seem that we require an equivalence rule such as $p \supset Oq \equiv O(p \supset q)$. While this formula is actually illegal in some deontic logics, it does seem intuitively acceptable. At any rate, the preliminary transformation work in normalizing a business rule formulation might involve more than just the Barcan equivalences or their deontic counterparts. In principle, this issue might be ignored for interoperability purposes, so long as the business domain expert is able to confirm that the final normalized formulation (perhaps produced manually by the business rules modeler) agrees with their intended semantics; it is only the final, normalized formulation that is used for exchange with other software tools.

The second issue concerns the dynamic nature of the rule. While it is obvious how to implement this rule in a database system, capturing the formal semantics in an appropriate logic (e.g. a temporal or dynamic logic) is a harder task. One possibility is to provide a temporal package that may be imported into a domain model, in order to provide a first-order logic solution. Another possibility is to adopt a temporal modal logic (e.g. treat a possible world as a sequence of accessible states of the fact model). For a discussion of why we prefer a first-order solution where possible, see [8].

6 Conclusion

In practice, many business rules are of a deontic rather than alethic nature. This paper discussed an approach for adding formal support for deontic constraints within information models, using ORM 2 to illustrate various examples. NORMA, an open source ORM 2 tool, is being used as a vehicle to implement the suggested approach. Although still at the prototype stage, this tool already provides automated verbalization of alethic and deontic constraints. While the ORM 2 modeling notation was used to illustrate the ideas, the notion of adding support for deontic constraints is just as relevant for other modeling approaches such as ER and UML, and much of the formal discussion in the paper applies equally well to these approaches.

The formalization of static constraints of both alethic and deontic modalities was discussed in some depth. NORMA's modality support is restricted to those modal formulae that include just one modal operator (it is necessary that, it is obligatory that), where that operator is the main operator. Such formulae appear to offer no major implementation difficulties. However, more complex formulae involving either

embedded deontic operators or embedded mention of propositions are far harder to support. While the paper identified some possible approaches to address these complex cases, further research is needed to determine the best solution. The topic of modalities in dynamic constraints also needs further research.

Acknowledgement: Some aspects of the logical formalization presented in this paper have benefited from discussions with Pat Hayes (IHMC, Florida).

References

1. Chen, P. P. 1976, 'The entity-relationship model—towards a unified view of data'. *ACM Transactions on Database Systems*, 1(1), pp. 9–36.
2. 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.
3. Girle, R. 2000, *Modal Logics and Philosophy*, McGill-Queen's University Press.
4. Halpin, T. 1989, 'A Logical Analysis of Information Systems: static aspects of the data-oriented perspective', doctoral dissertation, University of Queensland.
5. Halpin, T. 2001, *Information Modeling and Relational Databases*, Morgan Kaufmann, San Francisco.
6. 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).
7. 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.
8. Halpin, T. 2005, 'Higher-Order Types and Information Modeling', *Advanced Topics in Database Research, vol. 4*, ed. K. Siau, Idea Publishing Group, Hershey PA, USA, Ch. X (pp. 218-237).
9. Halpin, T. 2005. 'Objectification', *Proc. CAiSE'05 Workshops*, vol. 1, eds J. Castro & E. Teniente, FEUP, pp. 519-32.
10. Halpin, T. 2005, 'ORM 2', *On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops*, eds. R. Meersman, Z. Tari, P. Herrero et al., Cyprus. Springer LNCS 3762, pp 676-87.
11. Halpin, T. 2006, 'Object-Role Modeling (ORM/NIAM)', *Handbook on Architectures of Information Systems, 2nd edition*, Springer, Heidelberg, pp. 81-103.
12. 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.
13. ISO 2005, *ISO Common Logic Standard*. (Adoption expected April 2006). Draft online at <http://cl.tamu.edu/docs/cl/32N1377T-FCD24707.pdf>.
14. Object Management Group 2003, *UML 2.0 Infrastructure Specification*. Online: www.omg.org/uml.
15. Object Management Group 2003, *UML 2.0 Superstructure Specification*. Online: www.omg.org/uml.
16. Object Management Group 2006, *Semantics of Business Vocabulary and Rules Interim Specification*. Online at: www.omg.org/cgi-bin/doc?dtc/06-03-02.
17. Rumbaugh J., Jacobson, I. & Booch, G. 1999, *The Unified Language Reference Manual*, Addison-Wesley, Reading, MA.