
ORM 2 Constraint Verbalization Part 1

Technical Report ORM2-02, June 2006

*Terry Halpin, Matt Curland
and the CS445 Class¹*



www.neumont.edu

¹ Baldwin, S., Briggs, J., Burt, A., Dornseif, K., Flynn, D., Kaylor, C., Maly, N., Ouzts, M., Owen, K., Wilde, J., & Young, T.

Contents

1	Introduction	3	
1.1	Formal Items and Text Style		3
1.2	Hyphen Binding		3
1.3	Predicate Reading Choices		4
1.4	Object Variable Names		4
1.5	Personal and Impersonal Pronouns		4
1.6	Positive, Negative, and Default Form		4
1.7	Modality		4
1.8	Relational, Attribute, and Mixed Styles		4
1.9	Context		4
2	Uniqueness Constraints	5	
2.1	Internal Uniqueness Constraint (UCI)		5
2.1.1	Unary predicate		5
2.1.2	Binary predicate		5
2.1.3	N-ary predicate		10
2.2	External Uniqueness Constraint (UCE)		12
2.2.1	Binaries with predicate readings from the unconstrained roles		12
2.2.2	Binaries without predicate readings from the unconstrained roles		13
3	Mandatory Constraints	14	
3.1	Simple Mandatory Constraint (SMaC)		14
3.1.1	Unary predicate		14
3.1.2	Binary predicate		14
3.1.3	N-ary predicate		16
3.2	Disjunctive Mandatory (Inclusive-Or) Constraint (DMaC)		17
3.2.1	Unary predicates		17
3.2.2	Unary and Binary predicates		17
3.2.3	Unary and/or Binary and/or N-ary predicates		19
3.3	Combined Mandatory/Unique (Exactly-one) Constraint (CMU)		20
4	Implementation	22	
4.1	Dynamic Verbalization Requirements		22
4.2	Constructing a Verbalization Phase		22
4.2.1	Concatenation Approach to Verbalization		22
4.2.2	Field Replacement Approach to verbalization		23
4.3	Combining Verbalization Snippets		23
	References	25	

1 Introduction

Automated verbalization of constraints in Object-Role Modeling (ORM) was proposed long ago [e.g. 1], and a basic implementation is provided by Microsoft's ORM source model solution [2]. More recently, several improvements to the verbalization procedure have been proposed [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]. Currently, an open source tool called NORMA (Neumont ORM Architect)² is being developed to support the next generation of ORM (ORM 2) [19]. As part of this development, enhanced verbalization support is being included. This document summarizes the main aspects of the verbalization procedures for some ORM constraints that are implemented in this new tool. The discussion is restricted to the English language, but support for other languages is planned for later.

The verbalization is generated from the model, and not used to input models (for that, a separate and typically more concise input language is being devised). Apart from object-placeholder positioning, predicates are treated as unstructured character strings. For example, with the predicate "... originated from ..." no attempt is made to parse the text into a verb ("originated") and preposition ("from"). The English verbalization assumes that object type names are singular (not plural). For example, "Person" should be used instead of "Persons". Although this is common practice, modelers should be aware that use of plurals may lead to awkward verbalizations.

This technical report covers only uniqueness and mandatory constraints. The following technical report covers the other main categories of graphical constraints. For a future release, it is planned to support verbalization of additional constraints, such as relevant constraints over unrestricted join-paths (e.g. complex external-uniqueness and set-comparison constraints).

1.1 Formal Items and Text Style

Constraint verbalizations are comprised of a sequence of textual items that are object type names, full or partial predicate readings (used in relational-style or mixed-style verbalizations), role names (used in attribute-style or mixed style verbalizations), or *formal items* (quantifiers, pronouns, numbers, predefined text etc.). Formal items are comprised of pseudo-reserved words, and should be displayed in a different *text style* from the rest of the verbalization sentence. This text style should be globally selectable by the user as a configuration option. In this document, the **bold** style is used. Words within formal items are only pseudo-reserved, not fully reserved, since some of them may be used within a predicate reading. The text style indicates whether the word is being treated as a formal item. For example, in the following verbalization, only the second "a" is a formal item; the first "a" is just part of the predicate reading "is a resident of". While both instances of "a" have the meaning of an existential quantifier, only the second "a" formally communicates this meaning to the system.

Each Person is a resident of a Country

1.2 Hyphen Binding

If a hyphen within a predicate reading immediately precedes an object-placeholder, the word before it should be treated as an adjective that is bound to the object-type name for purposes of constraint verbalization (and the hyphen is elided in the verbalization). If a hyphen is followed by a space, some words, and an object place-holder, then the word just before it, as well as the words separating it from the object-placeholder, are bound to the object-type name for constraint verbalization. If the hyphen is followed by a non-space character, it is treated simply as part of the predicate (like any other character). Hyphen binding is left/right symmetric, so "most- influential Person" and "Person of highest -influence" would behave similarly, keeping the modifiers with Person irrespective of quantifiers. Also Student has Preference -1 could be used instead of Student has first- Preference.

For example, uniqueness constraints on the first role of the following fact types are verbalized as shown.

Person has first- GivenName.	→	Each Person has at most one first GivenName.
Person has very- high IQ	→	Each Person has at most one very high IQ.
Person drives a semi-trailer for Company	→	Each Person drives a semi-trailer for at most one Company.

² <https://sourceforge.net/projects/orm>

1.3 Predicate Reading Choices

For many constraints, more elegant verbalizations are obtained if predicate readings are supplied that start from a constrained role. If multiple predicate readings are available for the same fact type, then the verbalization of a constraint on a role will use a reading from that role if available.

1.4 Object Variable Names

If an object type is referenced only once in the body of a constraint verbalization, its name is left unaltered. If an object type is referenced two or more times, instances that may be different may be distinguished by subscripting (e.g. If Person₁ is a parent of Person₂ then ...). Multiple occurrences of a single object variable may also be distinguished by introducing the variable with a universal quantifier (e.g. For each), and then back-referencing by the pronoun (e.g. that), so long as no other variable of that type occurs in the reference path (e.g. For each Person: if that Person smokes then that Person is cancer-prone).

For localization to some languages (e.g. Chinese), pronouns (e.g. “that”, “who”, “itself”) which fit well in English do not always have natural equivalents. In such cases, subscripted variables should always be used.

1.5 Personal and Impersonal Pronouns

The modeler may declare any object type to be personal (it is impersonal by default). In English, the pronoun “who” is used in certain contexts for personal object types, and “that” is used for impersonal object types. For example, the following verbalizations capture two subset constraints between unaries:

Each Person **who** smokes is cancer-prone.
Each Car **that** smokes is due for a service.

1.6 Positive, Negative, and Default Form

Many, but not all, constraints may be verbalized in different forms (positive, negative, or default). The *positive form* is generally used first, and emphasizes what must hold for a satisfying population (e.g. Each Person has **at most one** Gender). The *negative form* indicates how the constraint may be violated, and is useful for discussing counterexamples (e.g. It is **impossible that** the same Person has **more than one** Gender). The *default form* is used to indicate what is implied by the absence of a constraint. For example, the absence of a uniqueness constraint on the second role of Person has Gender may be verbalized as It is possible that more than one Person has **the same** Gender.

1.7 Modality

By default, constraint verbalizations in positive form are assumed to be alethic in nature (logical or physical necessities). As a user option, this may be made explicit in the positive form of a verbalization by prepending “It is necessary that”, or in some cases appending “It is necessary that” to the For-list. If a constraint is declared to be deontic (indicated graphically by a different color and a mark such as “o” for “obligatory”), its positive verbalization uses “It is obligatory that” instead of “It is necessary that”.

Negative and default verbalizations include the relevant modality indicator: “possible” and “impossible” for alethic; “permitted” and “forbidden” for deontic.

1.8 Relational, Attribute, and Mixed Styles

Verbalizations in relational style use predicate readings (e.g. Each Person was born in **at most one** Country). Verbalizations in attribute style use role names (e.g. Each Person has **at most one** birthCountry). Verbalizations in mixed style use both predicate readings and role names. Relational style is used by default.

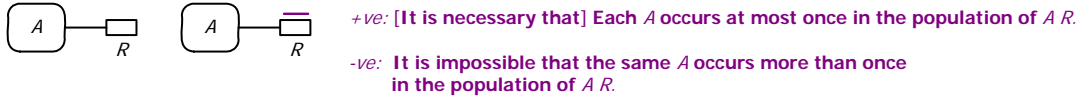
1.9 Context

Some constraints may be conveniently verbalized by declaring the *local context* (the fact types that contain the constrained elements) and then referring to that context. This is especially useful for some external constraints (e.g. Context: Room is in Building; Room has RoomNr. **In this context**, each Building, RoomNr combination is associated with **at most one** Room). Otherwise the constraint context is *global*.

2 Uniqueness Constraints

2.1 Internal Uniqueness Constraint (UCI)

2.1.1 Unary predicate



The top verbalization pattern above (by default without the necessity operator) is used for the *positive, relational, alethic* form of the constraint verbalization. *A* may be any object type (entity type or value type). *R* is a unary predicate. If the necessity operator is included, “Each” is replaced by “each”. For subsequent patterns, we will typically not bother stating this option. The bottom pattern above may be used for the *negative form* of the constraint verbalization.

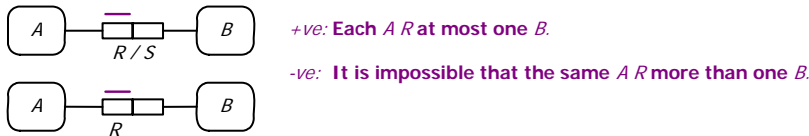
By default, fact types are assumed to set-based rather than bag-based, so the left-hand diagram above is interpreted as equivalent to the right-hand diagram. If we add support for bag populations, a new notation and verbalization will be introduced for “bag fact types”. The unary case is a special case of a predicate with a spanning uniqueness constraint. All such cases include a verbalization for the set-based nature (no duplicate instances in the population of the fact type).

Examples: (+ve) Each Person occurs at most once in the population of Person smokes.
 (-ve) It is impossible that the same Person occurs more than once in the population of Person smokes.

The unary predicate case has no deontic, attribute-style, or contextual form.

2.1.2 Binary predicate

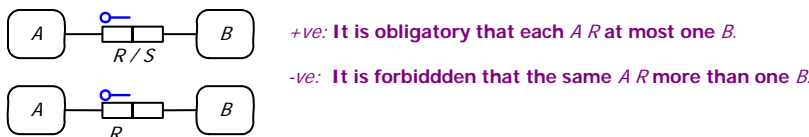
UC on a single role that starts a predicate reading



In the above pattern, *A* and *B* may be any object types (entity or value), not necessarily distinct. *R* is a predicate reading starting from *A*’s role (where no predicate text precedes the object type name in the fact type reading). A predicate reading *S* in the other direction may or may not be present. The *positive and negative forms of the relational, alethic* constraint verbalization patterns are shown.

Examples: (+ve) Each Person was born in at most one Country.
 (-ve) It is impossible that the same Person was born in more than one Country.

Deontic readings use these patterns with the relevant substitution of modality operators.



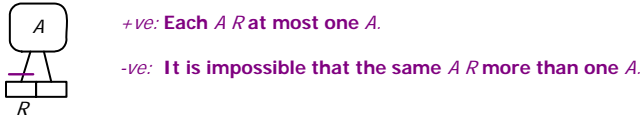
Examples: (+ve) It is obligatory that each Person has at most one SocialSecurityNumber.
 (-ve) It is forbidden that the same Person has more than one SocialSecurityNumber.

From now on, we will typically omit diagrams for deontic cases.

The *absence* of a UC on the role played by B may be indicated by the following *default* verbalization. There is no deontic version of this default, as the absence of a constraint is always interpreted alethically.

It is possible that the same B S more than one A. -- if S is available.
 It is possible that more than one A R the same B. -- if S is unavailable

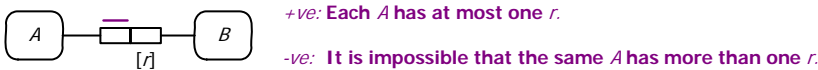
Examples: It is possible that the same Country is the birthplace of **more than one** Person.
 It is possible that **more than one** Person was born in the same Country.



The case when $A = B$ (shown above) is simply a special case of the previous pattern.

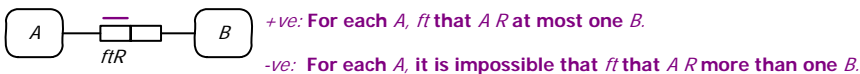
Examples: (+ve, alethic) Each Person has at most one Person as father.
 (-ve, alethic) It is impossible that the same Person has more than one Person as father.
 (+ve, deontic) It is obligatory that each Person is a husband of at most one Person.
 (-ve, deontic) It is forbidden that the same Person is a husband of more than one Person.

There is no contextual form for this constraint pattern, but an *attribute style* is available if and only if a role name is provided for the co-role of the constrained role, as shown below. This relies on the schema being well-formed (for each object type, its far role names are distinct, with the exception that symmetric predicates may have roles with the same name). The deontic versions prepend “It is obligatory that” to the positive form, and replace “impossible” by “forbidden” in the negative form.



Examples: Each Person has at most one birthCountry.
 It is impossible that the same Person has more than one birthCountry.

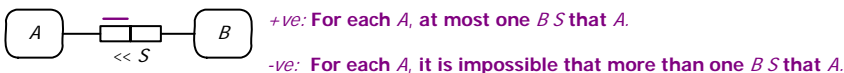
UC on a single role with a predicate reading including some front text (before the role’s object type)



In this case, the predicate ftR includes some *front text* (ft) that precedes the name of the role’s object type in the fact type reading. For example, the previous birth fact type may be rephrased as “the birth of Person occurred in Country”, using the front text “the birth of”. Such front text wordings are rare in practice, at least in English. The patterns for the alethic, relational verbalizations are shown above. These include the case where $A = B$. The deontic versions are obtained in the usual way. We do not support attribute-style for this case.

Examples: For each Person, the birth of that Person occurred in at most one Country.
 For each Person, it is impossible that the birth of that Person occurred in more than one Country.

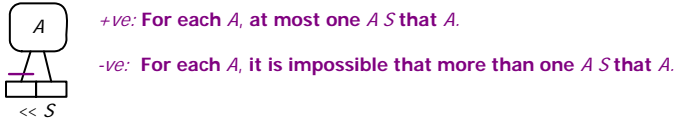
UC on a single role that does not start a predicate reading



The above pattern may be used for the positive and negative, relational, alethic verbalizations. The readings may be adapted for deontic versions as discussed earlier. Optionally, the positive alethic form may append “it is necessary that” to “For each A”. The “For each A” should be separated from the remainder of the verbalization by a comma (as shown) and/or cursor-return-linefeed. This comment also applies to later cases (where a list of one or more items follows “For each”).

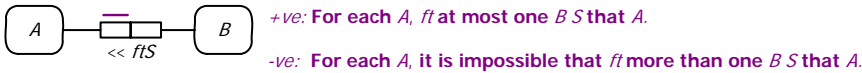
Examples: For each Person, at most one Country is the birthplace of that Person.
 For each Person, it is impossible that more than one Country is the birthplace of that Person.

The above pattern is used whether or not *A* is identical to *B*. If *A* is identical to *B*, there is no need to distinguish the instances of *A* by subscripting (see below).




Examples: For each Person, at most one Person is the father of that Person.
 For each Person, it is impossible that more than one Person is the father of that Person.

If the *inverse predicate has front text*, this front text (ft) precedes the numeric quantifier in the above patterns, as shown in the alethic patterns below. It is possible that *A* = *B*. The deontic versions are adapted as usual. The predicate text always maintains its placeholder places, and may even include trailing text (the examples below are based on the fact type readings “in Country was born Person” and “in Country Person was born”). Such wordings are rare in English.

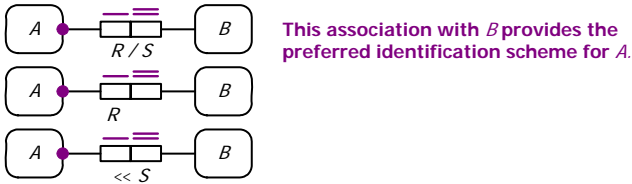


Examples: For each Person, in at most one Country was born that Person.
 For each Person, it is impossible that in more than one Country was born that Person.
 For each Person, in at most one Country that Person was born.
 For each Person, it is impossible that in more than one Country that Person was born.

1:1 predicates

 This case is handled by verbalizing the two uniqueness constraints individually according to the patterns discussed earlier, so introduces no new work.

Preferred identification



If a simple, preferred identification scheme is depicted explicitly, as shown above, where *A* is an entity type and *B* is any object type, the mandatory (see later) and uniqueness constraints are verbalized followed by the preferred identification declaration.

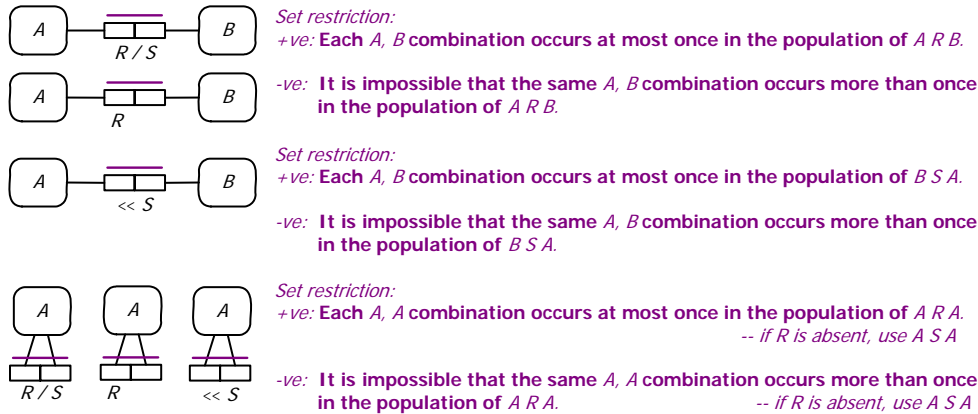
Example: This association with CountryCode provides the preferred identification scheme for Country.

If a simple, preferred identification scheme is depicted implicitly (using a parenthesized reference mode as shown in the left-hand diagram below), the below pattern is used. In this case, the object type *Aref* must be a value type. There are no deontic, negative, or attribute versions of a preferred identification verbalization.



Example: This Code value provides the preferred identifier for Country.

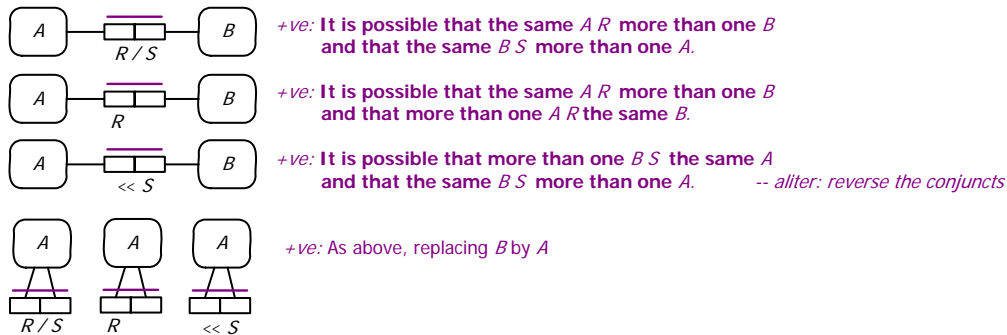
Spanning UC



The five cases for a spanning UC on a binary predicate are shown above. There are two aspects of the constraint verbalization: (a) restriction of the population to a set rather than a bag; (b) many:many nature. The set restriction patterns are shown above. The first +ve/-ve pattern pair (for the case when a forward predicate reading R is available) may be used for all cases: If only the inverse predicate S is supplied, the object types could be ordered B, A if this is easier (effectively reusing the first pattern); the ring binary case merely makes $A = B$. There is no deontic version of the set restriction.

Examples: (+ve) Each Person, Language combination occurs at most once in the population of Person speaks Language.
(-ve) It is impossible that the same Person, Language occurs more than once in the population of Person speaks Language.

The many-to-many nature of the relationship is verbalized alethically by applying the default verbalization for the lack of a UC on each of the two roles. This gives the following *positive, alethic, relational* patterns.



Examples: It is possible that the same Person speaks more than one Language and that the same Language is spoken by more than one Person.
It is possible that the same Person speaks more than one Language and that more than one Person speaks the same Language.
It is possible that more than one Language is spoken by the same Person and that the same Language is spoken by more than one Person. -- aliter: reverse the conjuncts
It is possible that the same Person likes more than one Person and that the same Person is liked by more than one Person.

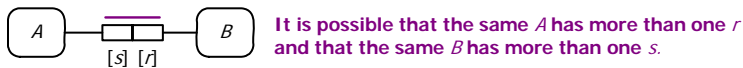
Note: For the ring binary case with only one predicate reading provided, there is a mild ambiguity for the conjunct with no lead reading (see second conjunct in the following example), but this is considered acceptable as the chance of misinterpretation is low, e.g.

It is possible that the same Person likes more than one Person
and that more than one Person likes the same Person.

The spanning UC constraint has *no negative verbalization*. Verbalization of its *deontic* version is obtained by replacing the “possible” modality in the above patterns by “permitted”. For example, in an economically deprived country the following deontic rule might not apply, even though its alethic version does apply:

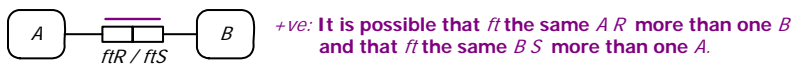
It is permitted that the same Person owns more than one Vehicle
and that the same Vehicle is owned by more than one Person.

An *attribute* style version may be supported using the following alethic pattern (use “permitted” if deontic). If only one role name is supplied, a *mixed style* version may be supported, using the relational style for the unnamed role. If no role names are supplied, there are no attribute or mixed style readings. Support for this is low priority.



Example: It is possible that the same Person has more than one carDriven
and that the same Car has more than one driver.

For those rare cases where the *predicate has front text*, the above patterns are adapted by placing the front text before the first quantifier. We do not support attribute style for such rare cases. For example:



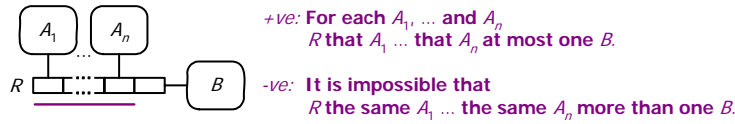
The following example is based on the fact type readings “when young Person played Sport / during youth-hood Sport was played by Person”.

Example: It is possible that when young the same Person played more than one Sport
and that during youth-hood the same Sport was played by more than one Person.

2.1.3 *N*-ary predicate (3 or more roles)

We restrict our attention to uniqueness constraints that are legal for elementary fact types. Any attempt to verbalize an illegal case returns a relevant model error. There are two legal cases: the UC spans $n-1$ roles; the UC spans all n roles ($n \geq 3$). We consider these cases in turn.

UC over all but one role



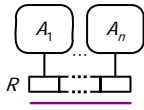
The $A_1 \dots A_n$ ($n \geq 2$) and B object types are entity or value types, are not necessarily contiguous, and are not necessarily distinct. The UC spans just the roles played by $A_1 \dots A_n$. The *alethic positive and negative relational* verbalization patterns are shown above. For the positive, alethic case, “it is necessary that” may optionally be appended after the For-list. It doesn’t matter which role starts the predicate reading R , but the primary predicate reading is selected by default. The object type names are placed in their normal positions within the mixfix predicate, prepended by the formal items shown. If the predicate R has front text, the pattern is still the same (see fourth example below). The universal quantifier reading “for each” is used by default, but the alternative reading “given any” could be supported as a user chosen alternative. If the same object type plays more than one role in the predicate, its instances must be distinguished by subscripting. For n -ary predicates, *no attribute style* version of constraints is supported.

- Examples:
- (+ve) For each Student and Course
that Student in that Course obtained at most one Rating.
 - (-ve) It is impossible that the same Student in the same Course obtained more than one Rating.
 - (+ve) For each Person₁, Person₂ and Year
that Person₁ supervised that Person₂ in that Year for at most one Period.
 - (-ve) It is impossible that
the same Person₁ supervised the same Person₂ in the same Year for more than one Period.
 - (ft) For each Student and Course
in normal circumstances that Student in that Course obtained at most one Rating.

The *deontic* versions are obtained by the rules discussed earlier.

- Examples:
- (+ve) It is obligatory that for each Room and HourSlot
that Room at that HourSlot is booked for at most one Course.
 - (-ve) It is forbidden that
the same Room at the same HourSlot is booked for more than one Course.

Spanning UC

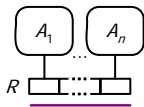


Set restriction:
 +ve: Each A_1, \dots, A_n combination occurs at most once in the population of $R A_1 \dots A_n$.
 -ve: It is impossible that the same A_1, \dots, A_n combination occurs more than once in the population of $R A_1 \dots A_n$.

The object types $A_1 \dots A_n$ ($n \geq 3$) are entity or value types, and are not necessarily distinct. If the same object type plays more than one role in the predicate, its instances must be distinguished by subscripting. The alethic positive reading of the set restriction aspect is shown above. Front text makes no difference. No deontic version is supported.

Examples: Each Person, Sport, Country combination occurs at most once in the population of Person played Sport for Country.
 It is impossible that the same Person, Sport, Country combination occurs more than once in the population of Person played Sport for Country.

The many:many: ... many nature of the relationship is verbalized by conjoining the default readings for the lack of an $n-1$ uniqueness constraint on each $n-1$ role combination (i.e. for an n -ary predicate, we indicate for each of its n roles the consequence of not uniquely constraining the other $n-1$ roles). This leads to a list of n possibilities, with exactly one object type occurrence preceded by “more than one” and the other object type occurrences preceded by “the same”. This generates a sentence pattern where “more than one” begins in the last object type position, then moves progressively to the first object type position, so the “more than one” instances line up on a diagonal, with “the same” in all other positions (italics are used here to display the diagonal, but are not used in the actual verbalization).



+ve: It is possible that *R the same A_1 the same A_2 ... more than one A_n*
 and that *R the same A_1 the same A_2 ... more than one A_{n-1} the same A_n*
 ...
 and that *R more than one A_1 the same A_2 ... the same A_n* .

The alethic positive pattern is shown above. If the same object type plays more than one role in the predicate, its instances must be distinguished by subscripting.

Example: It is possible that the same Person played the same Sport for more than one Country
 and that the same Person played more than one Sport for the same Country
 and that more than one Person played the same Sport for the same Country.

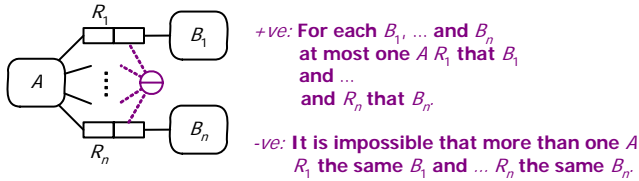
The binary case considered earlier is simply a special case of this pattern, except that subscripts are not required for that case. Note that it is incorrect to simplify the verbalization to a single pass through the predicate, prepending each object type name by “more than one”.

The *deontic* version is obtained by replacing “possible” by “permitted” in the above pattern. No negative version is supported.

2.2 External Uniqueness Constraint (UCE)

For now, we support just basic and nested cases involving binary predicates, where the join path of the constraint is simple. Complex join paths (e.g. [6] pp. 2-3) will be supported at a later stage.

2.2.1 Binaries with predicate readings from the unconstrained roles (no nesting)

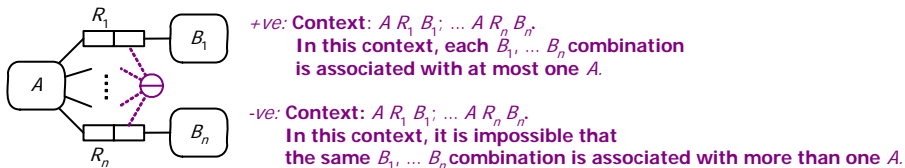


The object types may be entity or value types. The object types $B_1 \dots B_n$ are not necessarily distinct. If two or more of the B_i are identical, their instances in the verbalization must be distinguished by subscripting. The *positive and negative, alethic, relational* verbalization patterns are shown above. These patterns assume we have predicate readings starting (without front text) from each of the n roles of A ($n \geq 2$) shown here that are not spanned by the external uniqueness constraint. The *deontic* versions prepend the positive reading by “It is obligatory that” and replace “impossible” in the negative reading by “forbidden”.

- Examples:
- +ve: For each Building and RoomNr
at most one Room is in that Building and has that RoomNr.
 - ve: It is impossible that more than one Room
is in the same Building and has the same RoomNr.
 - +ve: For each Node₁ and Node₂
at most one Arrow is from that Node₁ and is to that Node₂. -- e.g. subtype-supertype connections
 - ve: It is impossible that more than one Arrow
is from the same Node₁ and is to the same Node₂.

The above patterns do not work if any of the predicates have *front text*. For example, consider the fact types: “the location of Room is in Building; the local number for Room is RoomNr”. For *front text* cases we instead use the contextual verbalization pattern discussed next (without context, general patterns that cater for front text cases where A may be identical to some of the B object types are awkward).

Contextual verbalization is possible, by predeclaring the fact types as the local context for the constraint. The positive and negative, alethic versions are shown below. The deontic versions are obtained by adapting these as discussed as above. These patterns work whether or not the predicates have front text.

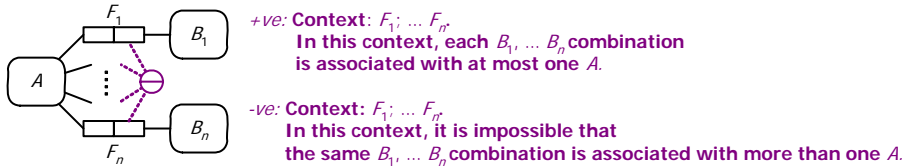


- Examples:
- +ve: Context: Room is in Building; Room has RoomNr.
In this context, each Building, RoomNr combination is associated with at most one Room.
 - ve: Context: Room is in Building; Room has RoomNr.
In this context, it is impossible that
the same Building, RoomNr combination is associated with more than one Room.
 - (ft) Context: the location of Room is in Building; the local number for Room is RoomNr.
In this context, each Building, RoomNr combination is associated with at most one Room

If role names exist for all the constrained roles, an attribute-style reading is possible, but we do not support it at this stage.

2.2.2 Binaries without predicate readings from the unconstrained roles (no nesting)

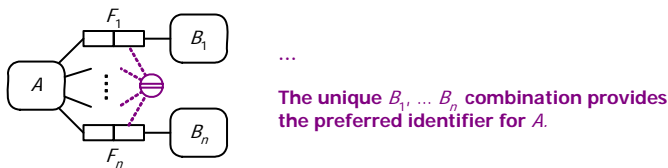
If one or more of the unconstrained roles does not start a predicate reading, then contextual verbalization must be used for relational style. The pattern is as for the case just considered except that in declaring the context the primary readings of the fact types $F_1 \dots F_n$ are used. Front text makes no difference. Deontic readings are adapted as before.



Examples: **+ve: Context: Building includes Room; RoomNr is of Room.**
In this context, each Building, RoomNr combination is associated with at most one Room.

-ve: Context: Building includes Room; RoomNr is of Room.
In this context, it is impossible that the same Building, RoomNr combination is associated with more than one Room.

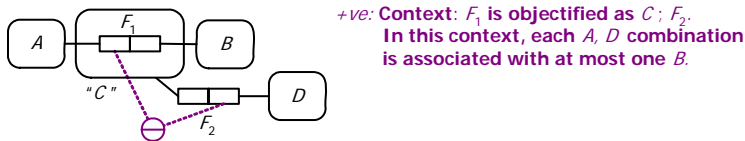
Preferred identification



If the external uniqueness constraint is a declared preferred (double line), the constraint is verbalized as before (for the relevant pattern), followed by indication of the preferred identifier, as shown above.

Example: The unique Building, RoomNr combination provides the preferred identifier for Room.

2.2.3 Nested binary



The nested object type C is an entity type, but all other object types may be entity or value types, not necessarily distinct. For the two fact types F_1 and F_2 , the primary reading is used; it does not matter which roles start a predicate reading, or whether there is front text. It is an error to declare the external uniqueness constraint as the preferred identifier. The positive, alethic, relational verbalization pattern is shown.

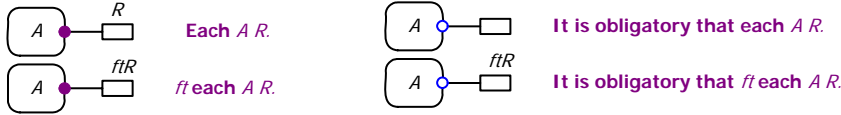
Example: **Context: Country plays Sport is objectified as Play; Play achieved Rank.**
In this context, each Sport, Rank combination is associated with at most one Country.

The deontic version is obtained by prepending “it is obligatory that”. No negative or attribute-style verbalizations are supported.

3 Mandatory Constraints

3.1 Simple Mandatory Constraint (SMaC)

3.1.1 Unary predicate

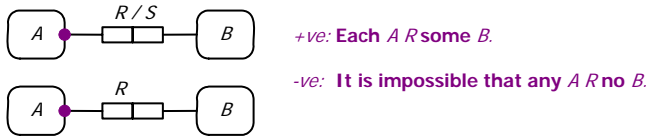


The object type *A* may be an entity type or a value type. The alethic, positive, relational verbalizations are shown above on the left, while the deontic versions are shown on the right. This pattern is rare for the alethic case because for that case the object type could simply be renamed to include the meaning of the predicate *R*. The deontic version is more likely to occur, and is obtained by prepending “It is obligatory that”. No negative or attribute-style verbalizations are supported. Unless the fact type is declared to allow bag populations, the implied UC is also verbalized (ignored here). To save space in later cases, we will typically not show the deontic diagrams.

Examples: Each Square is rectangular in shape.
 It is obligatory that each Doctor is licensed.
 Once a week each Employee has a day off.

3.1.2 Binary predicate

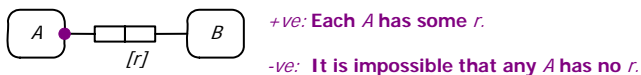
Mandatory role starts a predicate reading



The object types may be entity or value types. The *alethic, positive and negative relational* verbalizations are shown above. The quantifier “any” is used instead of “some” to remove a minor ambiguity in this context. If the user requests an *advanced* version of this verbalization, then “a” or “an” is used instead of “some” depending on whether the name of *B* starts with a consonant or vowel respectively (this feature is low priority, and is too simplistic for those cases where consonants are silent (e.g. “a Hourglass”) and is also locale dependent (e.g. “a Herb”). As usual, the *deontic* versions prepend “it is obligatory that” to the positive form, and replace “impossible” by “forbidden” in the negative form.

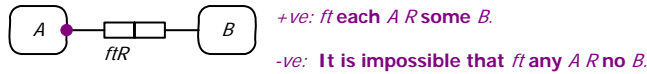
Examples: Each Person was born in some Country.
 It is impossible that any Person was born in no Country.
 It is obligatory that each Immigrant has some Passport
 It is forbidden that any Immigrant has no Passport.
 Each Person likes some Person.

An *attribute-style* reading is available if and only if *B*’s role is named. The positive and negative alethic forms are shown below. The deontic versions prepend “It is obligatory that” to the positive form, and replace “impossible” by “forbidden” in the negative form.



Examples: Each Person has some birthCountry.
 It is impossible that any Person has no birthCountry.

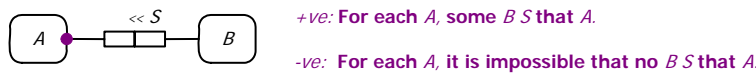
Mandatory role with a predicate reading including some front text (before the role's object type)



In this case, the predicate *ftR* includes some *front text (ft)* that precedes the name of the role's object type in the fact type reading. For example, the fact type reading "the birth of Person occurred in Country" uses the front text "the birth of". Such front text wordings are rare in English. The patterns for the alethic, relational verbalizations, shown above, simply insert the front text before the quantifier in the previous patterns. The ambiguity for the uniqueness constraint case does not occur here. These include the case where $A = B$. The deontic versions are obtained in the usual way. We do not support attribute-style for this case.

Examples: The birth of each Person occurred in some Country.
It is impossible that the birth of any Person occurred in no Country.

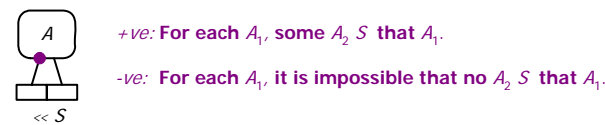
Mandatory role does not start a predicate reading



The positive and negative, alethic, relational verbalization patterns are shown above. For the positive, alethic case, "It is necessary that" may optionally be appended after the For-list. Other comments apply as for the previous case (deontic versions etc.).

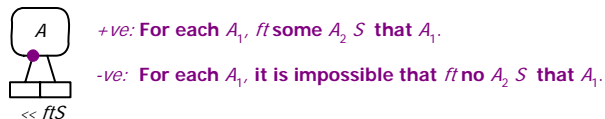
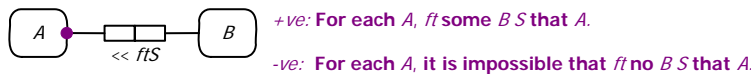
Examples: For each Person, some Country is the birthplace of that Person.
For each Employee, it is impossible that no Country is the birthplace of that Person.

If A and B are identical, their instances in the verbalization must be distinguished by subscripting as shown below (this is just a special case of the above pattern).



Examples: For each Person₁, some Person₂ is liked by that Person₁.
For each Person₁, it is impossible that no Person₂ is liked by that Person₁.

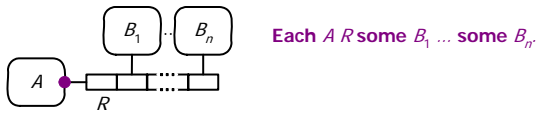
If the *inverse predicate has front text*, this front text (*ft*) precedes the numeric quantifier in the above patterns, as shown in the alethic patterns below. If $A = B$ subscripting is used. The deontic versions are adapted as usual. The predicate text always maintains its placeholder places, and may even include trailing text (the examples below are based on the fact type readings "in Country was born Person", "in Country Person was born" and "fortunately Person is liked by Person"). Such wordings are rare in English.



Examples: For each Person, in some Country was born that Person.
For each Person, it is impossible that in no Country was born that Person.
For each Person, in some Country that Person was born.
For each Person, it is impossible that in no Country that Person was born.
For each Person₁, fortunately some Person₂ is liked by that Person₁.
For each Person₁, it is impossible that fortunately no Person₂ is liked by that Person₁.

3.1.3 N-ary predicate

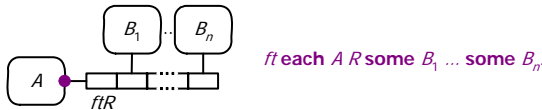
Mandatory role starts a predicate reading



The positive, alethic, relational verbalization pattern is shown above. No negative version is supported (as negative verbalizations are cumbersome for this case). No subscripting is needed to distinguish different instances of the same object type. The deontic version prepends “It is obligatory that”.

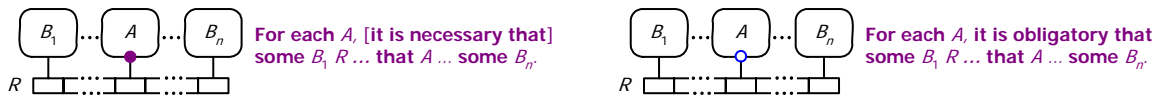
Examples: Each Programmer codes in **some** ProgrammingLanguage at **some** SkillLevel.
 Each Person eats **some** Food cooked by **some** Person.
 It is obligatory that each Translator knows **some** ForeignLanguage at **some** SkillLevel.

If the predicate has *front text*, this is placed in front of the universal quantifier, as shown below.



Example: When hungry **each** Person eats **some** Food cooked by **some** Person.

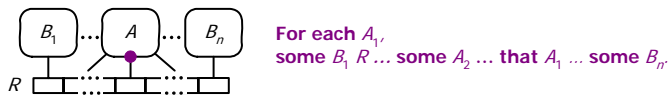
Mandatory role does not start a predicate reading



If A plays only one role in the predicate, the positive, alethic, relational verbalization is as shown above. No negative version is supported (too awkward). The deontic version is obtained by appending “it is obligatory that” to the For-list. Optionally, the alethic versions may append “it is necessary that” to the For-list.

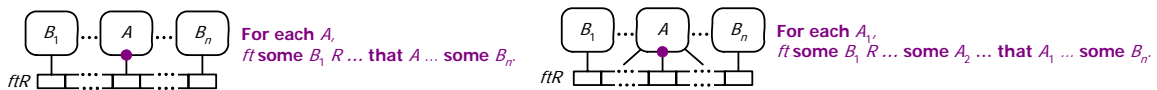
Examples: For each Programmer, **some** ProgrammingLanguage is coded in by **that** Programmer at **some** SkillLevel.
 For each Translator, it is obligatory that **some** ForeignLanguage is known by **that** Translator at **some** SkillLevel.

If A plays more than one role in the predicate, its different instances must be distinguished by subscripting. The positive, alethic, relational verbalization is as shown below. No negative version is supported. The deontic version appends “It is obligatory that” to the For-list.



Example: For each Person₁, **some** Food cooked by **some** Person₂ is eaten by **that** Person₁.

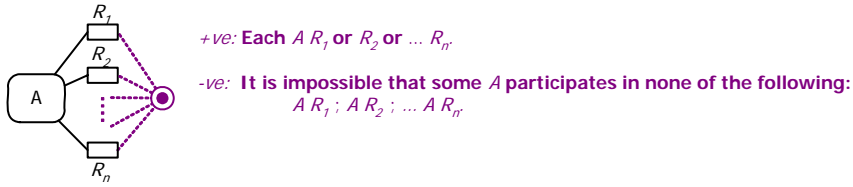
For the two patterns above, if the predicate R has *front text*, this is placed in front of the first existential quantifier, as shown below.



Example: For each Person₁, when ready **some** Food cooked by **some** Person₂ is eaten by **that** Person₁.

3.2 Disjunctive Mandatory (Inclusive-Or) Constraint (DMaC)

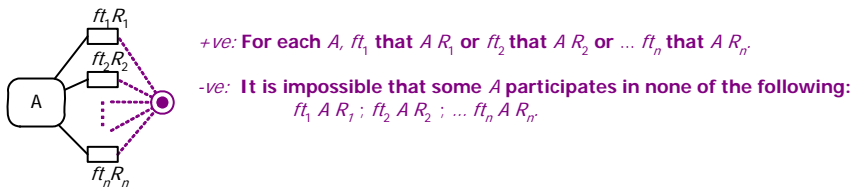
3.2.1 Unary predicates



Here $R_1 .. R_n$ are n unary predicates ($n \geq 2$). The *alethic, positive and negative relational* verbalizations are shown above. Users should understand that “or” in verbalizations always means inclusive-or. The *deontic* versions prepend “it is obligatory that” to the positive form, and replace “impossible” by “forbidden” in the negative form. There is no attribute-style version.

Examples: Each Person is male or is female.
 It is impossible that some Person participates in none of the following:
 Person is male; Person is female.
 Each Plant is male or is female or is hermaphroditic.
 It is obligatory that each Vehicle is purchased or is rented.

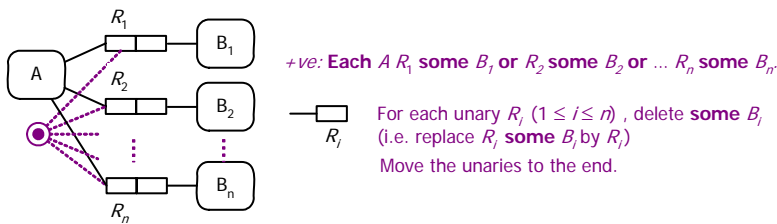
If even one of the predicates has *front text*, the pattern changes to that shown below.



Example: For each Person, when hungry that Person eats or when thirsty that Person drinks.

3.2.2 Unary and/or Binary predicates

Each constrained role starts a predicate reading



The n predicates $R_1 .. R_n$ ($n \geq 2$) are either unary or binary. The B object types are not necessarily distinct. The predicates may be all binary, all unary, or a mix of binary and unaries. Hence this pattern includes the unary-only, positive pattern just discussed as a special case. The *alethic, positive relational* verbalizations are shown above. Remove “some B_i ” for each unary predicate R_i , and *verbalize all the binaries before all the unaries* (if we swap the disjuncts in the first example below to yield “Each Lecturer is tenured or is contracted until some Date”, it is not as obvious that “until some Date” applies only to the contracted lecturers).

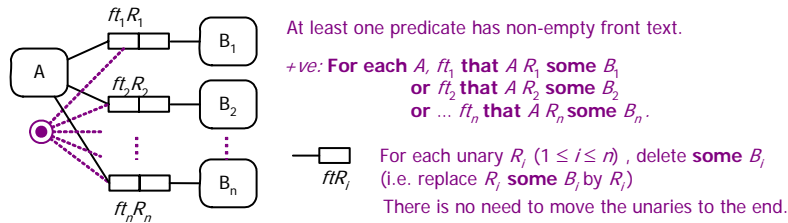
No negative reading is supported. For the all-binary case, a negative reading is possible (It is impossible that some $A R_1$ no B_1 and R_2 no B_2 and ... R_n no B_n) but it is not worth the trouble to support this as it doesn’t cater for

unaries, and the previous pattern for unaries-only doesn't work for binaries if *A* plays more than one role in one of the predicates. No attribute version is supported (one may be easily be devised, but it is no clearer than the relational version). The *deontic* version prepends “it is obligatory that” to the positive form.

Examples: Each Lecturer is contracted until some Date or is tenured.
 It is obligatory that each Vehicle was purchased on some Date or was rented on some Date..

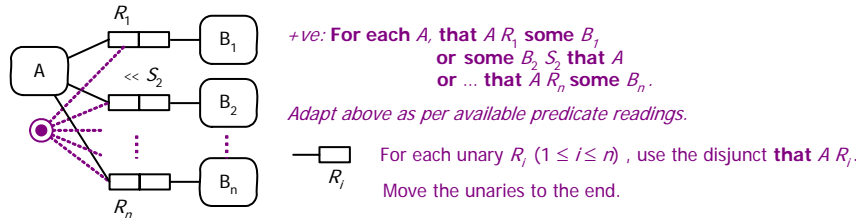
Front text adaptation:

If even one of the predicates in the previous case has *front text*, then the following pattern is used instead. For those R_i predicates without front text, the ft_i term is ignored.



Example: For each Person, when hungry that Person eats some Food
 or when thirsty that Person drinks.

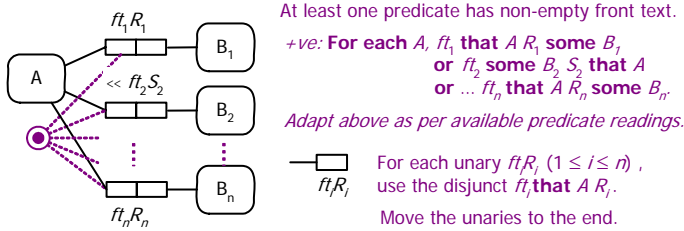
Some constrained roles do not start a predicate reading



This pattern is used so long as at least one of the constrained roles does not start a predicate reading, and no predicate has front text. The diagram illustrates just one case. For binary fact types, if a predicate reading R_i from the constrained role is available, use it to form the disjunct “that $A R_i$ some B_i ”; otherwise use the alternate predicate reading S_i to form the disjunct “some $B_i S_i$ that A ”. For each unary predicate R_i , use the disjunct “that $A R_i$ ”. Verbalize all the binaries before all the unaries. The *positive, alethic, relational* pattern is shown above. No negative or attribute versions are supported, as these are awkward or add no clarity. The *deontic* version prepends “it is obligatory that”, making it clear that the scope of the modal operator includes the whole disjunction (placing the modal operator after the For-list would result in ambiguity, since the scope might be incorrectly be taken to exclude the second disjunct – consider the second example below with the obligation operator placed at the end of the first line instead of the start).

Examples: For each Lecturer,
 some Date is the contract-expiry date for that Lecturer
 or that Lecturer is tenured.
 It is obligatory that for each Vehicle,
 some Date was the purchase-date for that Vehicle
 or some Date was the rental-date for that Vehicle.

If any of the predicates has *front text*, the above pattern is adapted as shown below to place the front text before the “that” (for a predicate reading from the constrained role) and before the “some” for a predicate reading from the unconstrained role. The diagram shows front text for all predicates. If there is no front text for some predicate, the ft term in the pattern is ignored.



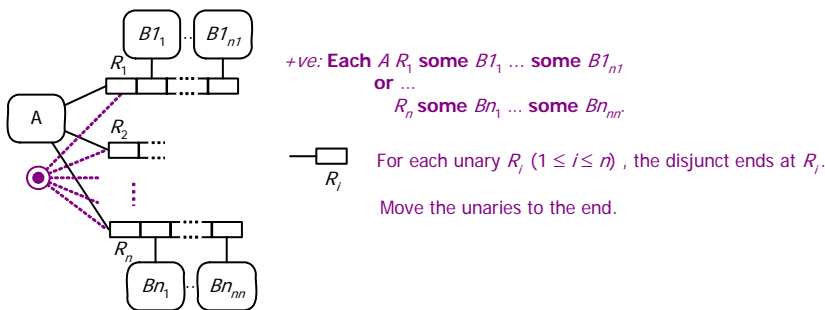
Example: For each Person, when hungry that Person eats some Food
 or to address a thirst some Liquid is drunk by that Person.

3.2.3 Unary and/or Binary and/or N-ary predicates

These patterns simply generalize the previous unary and/or binary patterns, by including “some” before the additional object type instances. No subscripting is needed to distinguish instances of the same type.

Each constrained role starts a predicate reading

The *positive, alethic relational* pattern is shown below. 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), and the object types may or may not be the same. Verbalize all the binaries before all the unaries. No negative or attribute versions are supported, as these are awkward or add no clarity. The *deontic* version prepends “it is obligatory that” to the positive form.

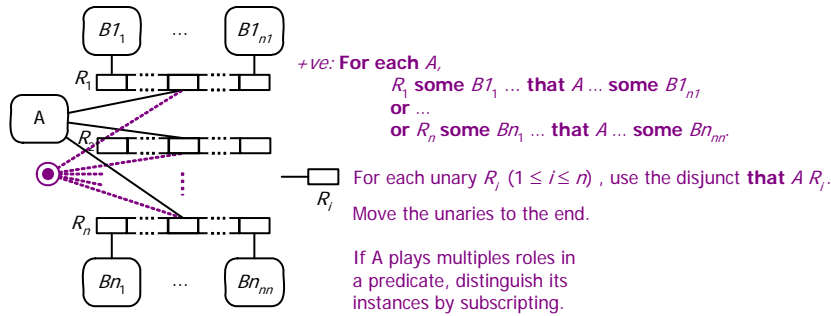


Examples: Each Partner became the husband of some Partner on some Date
 or became the wife of some Partner on some Date.
 It is obligatory that each Vehicle was purchased from some BranchNr of some AutoRetailer or is rented.

Front text adaptation:

If even one of the predicates in the previous case has *front text*, then the following pattern is used instead. Here if any R_i predicate contains front text, this is simply included as part of the predicate reading.

Some constrained roles do not start a predicate reading



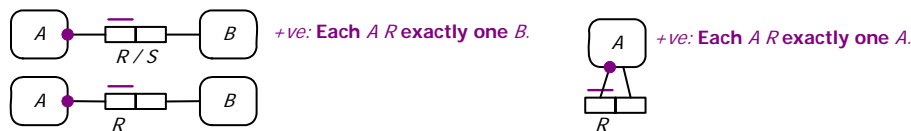
The *positive, alethic relational* pattern is shown above. The object types are not necessarily distinct. If *A* plays more than one role in at least one of the R_i predicates, subscript its instances to distinguish them. Each predicate R_i may have n_i roles ($n_i \geq 0$) plus the role played by *A*. Verbalize all the non-unaries before all the unaries. No negative or attribute versions are supported, as these are awkward or add no clarity. The deontic version prepends “It is obligatory that” to the positive form.

Examples: For each Partner₁,
on some Date that Partner₁ became the husband of some Partner₂
or on some Date that Partner₁ became the wife of some Partner₂.
It is obligatory that for each Vehicle,
some BranchNr of some AutoRetailer sold that Vehicle
or that Vehicle is rented.

3.3 Combined Mandatory/Unique (Exactly-one) Constraint (CMU)

If a role has both a simple uniqueness and a mandatory constraint, and both constraints are selected for verbalization (e.g. by selecting the fact type), then if the configuration option to verbalize constraints in combined form is set to Yes, both constraints may be combined in a single, more compact verbalization as shown below. The main idea to use “exactly one” to abbreviate “some (at least one) and at most one”. The current Visio tool supports this option for constraint input prompts, but does not support it for output verbalization. No unary predicate version is supported (as the set restriction aspect of a UC has no deontic form, but the mandatory constraint does). No n -ary version is supported (as a simple UC on an n -ary violates the $n-1$ rule).

Constrained role starts a predicate reading



The verbalization pattern above is used for the *positive, relational, alethic form* of the constraint verbalization. The object types *A* and *B* are not necessarily distinct (so the right hand-hand diagram is simply a special case of the bottom left-hand diagram). No negative or attribute-style form is supported. The *deontic* version prepends “it is obligatory that” to the positive form.

Examples: Each Person was born in exactly one Country.
It is obligatory that each Immigrant has exactly one Passport
Each Person is identical to exactly one Person.

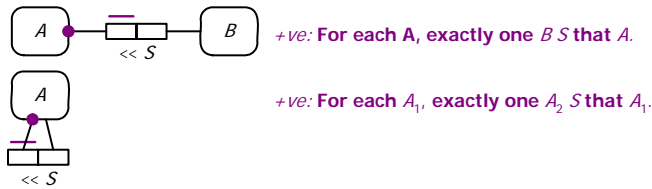
Front text adaptation:

If the predicate contains front text, the verbalization pattern is adapted to that shown below, placing the front text before the first “that”. This includes the case where *A* is identical to *B*.



Example: For each Person, the birth of that Person occurred in exactly one Country.

Constrained role does not start a predicate reading



The verbalization pattern above is used for the *positive, relational, alethic form* of the constraint verbalization. If the object types *A* and *B* are identical (bottom diagram), their instances must be distinguished by subscripting. No negative or attribute-style form is supported. The deontic version is obtained by appending “it is obligatory that” to the For-list. Optionally, the alethic versions may append “it is necessary that” to the For-list.

Examples: For each Person, exactly one Country was the birthplace of that Person.
For each Immigrant, it is obligatory that exactly one Passport belongs to that Immigrant.
For each Person₁, exactly one Person₂ is identical to that Person₁.

Front text adaptation:

If the predicate contains front text, the verbalization pattern is adapted to that shown below, placing the front text before the “exactly one” quantifier. This includes the case where *A* is identical to *B*.



Example: For each Person, in exactly one Country was born that Person.

That completes the verbalization patterns for all uniqueness and mandatory constraints (with the exception of uniqueness constraints over complex join paths). Verbalization of other constraint types will be discussed in the next Technical Report.

4 Implementation

Implementing a verbalization pattern must be approached carefully because of the eventual size of any implementation. As you can tell from this document, which discusses only a subset of the possible constraints that need to be verbalized in an ORM tool, the number of potential variations is very high. A conservative estimate is that a full ORM verbalization implementation coded by hand would have 10,000-15,000 lines of code, or about 6 man months (again a conservative estimate) to create an initial implementation. Incremental maintenance costs on such an implementation would also be extremely high due to the size of the code. To succeed both short term and long term we decided to use a pattern-driven generative approach to implementing the code.

4.1 Dynamic Verbalization Requirements

The rules for verbalization of a given constraint pattern are constant, but the actual text used for different parts of the verbalization depends on a number of environment-specific factors.

1. The most obvious environmental factor is output language. Although our reference implementation uses English verbalization phrases, verbalization in other languages should incur only incremental additional implementation costs.
2. The same verbalization engine should be able to render different output formats. The NORMA tool's verbalization window will display html, but we may want different html for a report view, and plain text in other views. The same verbalization engine should be able to handle outputs in all of these formats.
3. Personal verbalization preferences are also an environment factor. For example, by default we do not show the implied 'it is necessary that' before positive alethic constraints. However, any skilled user should be able to choose to see the explicit form, or phrase it as 'the following condition is necessary: '. Other verbalization text should be equally flexible.
4. A skilled user should be able to easily adapt the verbalization output to the current target audience for the tool. Examples include an ORM instructor who would like to emphasize a specific quantifier in a lecture, or a consultant who would like to replace the default deontic 'it is obligatory that' with 'One ought to ensure that' or even the personalized '*CompanyName* policy requires that'.

The goal of dynamic verbalization is to output verbalization that is easily verified by the reader. If the reader prefers a language other than English, then verbalize in the other language. If the reader wants a full report instead of individual diagram selections, then the same verbalization engine should be able to produce both a standalone printed report and a website of mini-reports for each object type, fact type, and constraint. Similarly, if readers want to see their company name mentioned in company-specific business rules, they should have this flexibility.

4.2 Constructing a Verbalization Phrase

There are two approaches that can be used to generate a verbalization phrase. Both are combinations of user-provided predicate text and text particles provided by the verbalization engine. The first approach uses *concatenation*, where pieces of a phrase are constructed by combining particles in a specific order. The second approach uses *field replacement*, where the particles specify the location and order of the particles surrounding them. Let's break down the verbalized phrase 'Each Person was born in some Country' using both approaches.

4.2.1 Concatenation Approach to Verbalization

To use concatenation to verbalize this phrase requires 7 strings to be combined in the correct order. The first three strings form the predicate text, which requires arity+1 strings for any predicate text. In this case, the leading and trailing strings of the binary predicate are empty, but that information is still required. The three predicate strings are {"", " was born in ", ""}. In addition to the predicate strings, we can round out the user-provided data with the object type names {"Person", "Country"}. The verbalization engine then provides the universal quantifier "each " and the existential quantifier "some ". The specified pattern (a simple mandatory constraint on a binary predicate where predicate text is available such that the mandatory role begins the reading) is now built as follows (using + to mean concatenation). Though not shown here, each verbalization starts with an initial capital letter and ends in a period “.”

"" + "each " + "Person" + " was born in " + "some " + "Country" + ""

In practice, generating formatted text makes this significantly more complicated because each particle must include formatting specifications before and after the text, thus tripling the number of text particles necessary to complete the phrase.

4.2.2 Field Replacement Approach to Verbalization

The field replacement approach uses numbered replacement fields. We'll show these in the format required for the .NET System.String.Format function, which uses (regular expression) “\{d+}” to denote a zero-based replacement field in a format string. For example, “{0}” in a format string is the placeholder for the first replacement field. The Format function takes a format string as the first argument, followed by arguments for the replacement fields. For our current phrase, the predicate text is “{0} was born in {1}” and the quantifiers become “each {0}” and “some {0}”. The equation now looks like:

```
REPLACE("{0} was born in {1}", REPLACE("each {0}", "Person"), REPLACE("some {0}", "Country"))
```

Note that there are immediate advantages to this approach.

- 1) The model stores a single predicate text with replacement fields instead of arity+1 strings.
- 2) Adding formatting specifications to verbalization-provided quantifiers does not add any additional snippets.
- 3) Order dependency is eliminated: “each” may come before “Person” in English, but there is a strong likelihood that other languages will have some quantifiers either after or around a given replacement field. Using the concatenation approach would require either new code for each language or before/after specification for every phrase. Using the replacement approach removes all ordering and formatting considerations from the code by placing the onus for ordering specification on the snippet specifications.

Using field replacement exclusively in the ORM2 verbalization engine allows all *snippets* to be specified as user-modifiable data. Concatenation is used for generating lists only. Using field replacements simplifies the verbalization engine and enables data-driven snippet sets to be specified according to both language and user preferences. The snippets are considered dynamic data; how they are combined is specified statically for each verbalized model element.

4.3 Combining Verbalization Snippets

The process of implementing verbalization phrases for a given constraint involves identifying the required snippets and how to combine the snippets with user-provided data. This paper specifies a number of factors that are used to determine how constraints are verbalized. For example, the verbalization pattern used can depend on the availability of a predicate for a given reading order. The patterns used when a forward reading is available are radically different from the pattern used when only the reverse reading is available or when the forward reading contains front text. However, regardless of the pattern, the basic approach is the same: Recursively provide a set of replacement fields to populate either a model-provided (predicate text) or verbalization-engine-provided format string.

As previously discussed, even with all of the formatting work moved to a set of data-driven snippets, the size of a hand-written verbalization implementation is prohibitively large. To keep the implementation costs within reach and to easily verify fidelity with the specification we use an XML/XSLT-based generative approach for snippet creation. The generative approach repeatedly cycles through three steps:

- 1) Schematize: *Determine an XML schema that represents the patterns and conditions expressed in the verbalization specification.*
- 2) Populate: *Populate the XML document with information from the schema.*
- 3) Generate: *Map the patterns and conditions represented in the XML document to code.*

A fragment from our generation document shows the verbalization specification for a single-role mandatory constraint on both unary and binary predicates.

```
<Constraint type="MandatoryConstraint" patternGroup="SetConstraint">
  <!-- 3.1.1 unary predicate -->
  <!-- Each A R -->
  <ConstrainedRoles constraintArity="1" factArity="1" sign="positive">
    <Snippet ref="ImpliedModalNecessityOperator">
      <Snippet ref="UniversalQuantifier">
```

```

        <Fact/>
    </Snippet>
</Snippet>
</ConstrainedRoles>

<!-- 3.1.2 binary predicate -->
<ConstrainedRoles constraintArity="1" factArity="2">
    <ConditionalReading>

        <!-- mandatory constraint starts reading relational style -->
        <!-- Each A R some B -->
        <ReadingChoice match="RequireLeadReading">
            <Snippet ref="ImpliedModalNecessityOperator">
                <Fact readingChoice="Conditional">
                    <PredicateReplacement match="included">
                        <Snippet ref="UniversalQuantifier"/>
                    </PredicateReplacement>
                    <PredicateReplacement>
                        <Snippet ref="ExistentialQuantifier"/>
                    </PredicateReplacement>
                </Fact>
            </Snippet>
        </ReadingChoice>

        <!-- mandatory role does NOT start reading -->
        <!-- For each A, some B S that A -->
        <ReadingChoice>
            <Snippet ref="ForEachCompactQuantifier">
                <IterateRoles match="included" listStyle="SimpleList"/>
                <Snippet ref="ImpliedModalNecessityOperator">
                    <Fact>
                        <PredicateReplacement match="included">
                            <Snippet ref="DefiniteArticle"/>
                        </PredicateReplacement>
                        <PredicateReplacement>
                            <Snippet ref="ExistentialQuantifier"/>
                        </PredicateReplacement>
                    </Fact>
                </Snippet>
            </Snippet>
        </ReadingChoice>
    </ConditionalReading>
</ConstrainedRoles>
</Constraint>

```

This XML, along with similar XML used to generate the default snippet values, provides the full input for simple mandatory constraints on unary and binary predicates. The XML tags fall into four categories:

- 1) Type and pattern identification (Constraint). These tags tell the code generator which class to generate (in this case, MandatoryConstraint) and the type of pattern to recognize (SetConstraint).
- 2) Snippet references (Snippet). Each snippet tag has the same number of child XML elements as replacement fields in the snippet.
- 3) Conditional patterns (ConstraintRoles, ConditionalReading, ReadingChoice). These nodes translate into conditional branching instructions in code and represent conditions such as the availability of a predicate that starts with a specific role, the number of roles affected by the current constraint, or the arity of facts attached to that constraint.

- 4) Other iteration and replacement constructs (IterateRoles, Fact, PredicateReplacement). These nodes create code to extract replacement fields from the current context element for use in the containing snippets. For example, the *Fact* element represents the predicate text (complete with replacement fields) from the ORM model. The default replacement fields are the object type names, but each object type can be wrapped in another snippet. The match="include" attribute refers to the roles from the context constraint object. The PredicateReplacement elements for the last Fact entry in the sample XML indicates that we should wrap all constrained roles with the *DefiniteArticle* snippet, all other roles with the *ExistentialQuantifier* snippet.

References:

1. Halpin, T.A. & Harding, J. 1993, 'Automated support for verbalization of conceptual schemas', *Proc. 4th Workshop on Next Generation CASE Tools*, eds S. Brinkkemper & F. Harmsen, Univ. Twente Memoranda Informatica 93-32, pp. 151-161, Paris (June).
2. Halpin, T. A., Evans, K, Hallock, P. & MacLean, W. 2003, *Database Modeling with Microsoft® Visio for Enterprise Architects*, Morgan Kaufmann, San Francisco.
3. Halpin, T.A. 2003, 'Verbalizing Business Rules: Part 1', *Business Rules Journal*, Vol. 4, No. 4, (April 2003), URL: <http://www.BRCommunity.com/a2003/b138.html>.
4. Halpin, T.A. 2003, 'Verbalizing Business Rules: Part 2', *Business Rules Journal*, Vol. 4, No. 5, (June 2003), URL: <http://www.BRCommunity.com/a2003/b152.html>.
5. Halpin, T. A. 2003, 'Verbalizing Business Rules: Part 3', *Business Rules Journal*, Vol. 4, No. 8 (August 2003), URL: <http://www.BRCommunity.com/a2003/b163.html>.
6. Halpin, T. A. 2003, 'Verbalizing Business Rules: Part 4', *Business Rules Journal*, Vol. 4, No. 10 (October 2003), URL: <http://www.BRCommunity.com/a2003/b172.html>.
7. Halpin, T. A. 2004, 'Verbalizing Business Rules: Part 5', *Business Rules Journal*, Vol. 5, No. 2 (February 2004), URL: <http://www.BRCommunity.com/a2004/b179.html>.
8. Halpin, T. A. 2004, 'Verbalizing Business Rules: Part 6', *Business Rules Journal*, Vol. 5, No. 4 (April, 2004), URL: <http://www.BRCommunity.com/a2004/b183.html>.
9. Halpin, T. A. 2004, 'Verbalizing Business Rules: Part 7', *Business Rules Journal*, Vol. 5, No. 7 (July, 2004), URL: <http://www.BRCommunity.com/a2004/b198.html>.
10. Halpin, T. A. 2004, 'Verbalizing Business Rules: Part 8', *Business Rules Journal*, Vol. 5, No. 9 (September, 2004), URL: <http://www.BRCommunity.com/a2004/b205.html>.
11. Halpin, T. A. 2004, 'Verbalizing Business Rules: Part 9', *Business Rules Journal*, Vol. 5, No. 12 (December, 2004), URL: <http://www.BRCommunity.com/a2004/b215.html>.
12. 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.
13. Halpin, T. A. 2005, 'Verbalizing Business Rules: Part 10', *Business Rules Journal*, Vol. 6, No. 4 (April, 2005), URL: <http://www.BRCommunity.com/a2005/b229.html>.
14. Halpin, T. A. 2005, 'Verbalizing Business Rules: Part 11', *Business Rules Journal*, Vol. 6, No. 6 (June 2005), URL: <http://www.BRCommunity.com/a2005/b238.html>.
15. Halpin, T. 2005, 'Verbalizing Business Rules: Part 12', *Business Rules Journal*, Vol. 6, No. 10 (Oct. 2005), URL: <http://www.BRCommunity.com/a2005/b252.html>.
16. Halpin, T. 2005, 'Verbalizing Business Rules: Part 13', *Business Rules Journal*, Vol. 6, No. 12 (December 2005), URL: <http://www.BRCommunity.com/a2005/b261.html>.
17. Halpin, T. 2006, 'Verbalizing Business Rules: Part 14', *Business Rules Journal*, (in press).
18. Halpin, T. 2006, 'Verbalizing Business Rules: Part 15', *Business Rules Journal*, (in press).
19. 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.