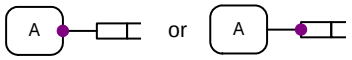
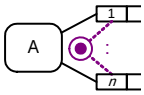
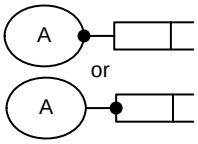
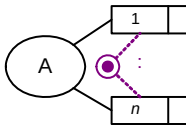
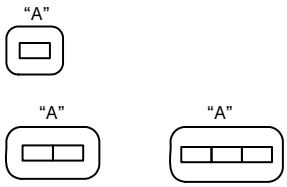
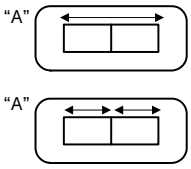
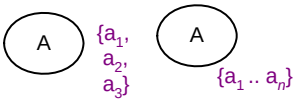
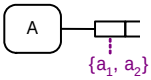
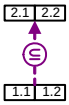
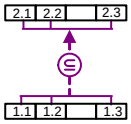
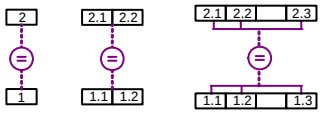
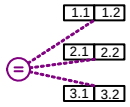
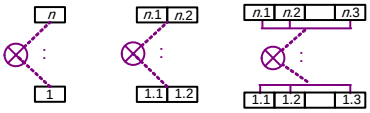
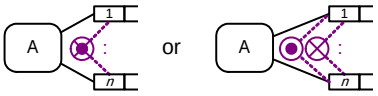
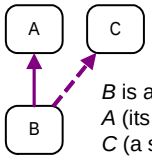
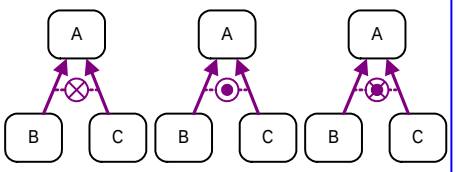
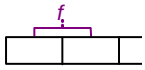
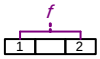
ORM 1 and ORM 2 Symbol Glossary

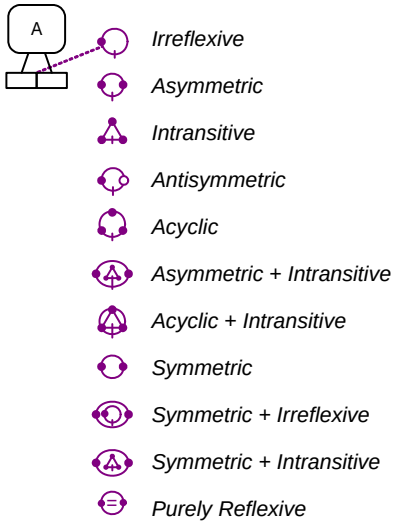
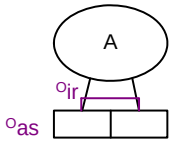
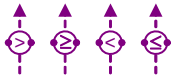
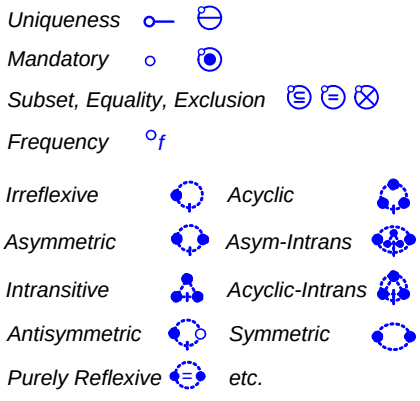
ORM 2	ORM 1
Object Types	
Entity type A (first shape is the default) A or A or A From now on, we show only the default shape.	A
A identified by *reference mode Ref* A (Ref)	A (Ref)
Value type A A	A
Independent entity type A A !	A !
Independent value type A A !	Not supported.
Duplicated object types A B	Not supported.
External object type A A^ To be finalized	A
Predicates	
Unary: R *Binary:* R, ◂S, R / S Forward reading; Inverse reading; Both readings *Ternary:* R *Quaternary:* R etc. (*n* role-boxes for *n*-ary predicate)	Predicates are basically the same, except that role boxes are larger
Role names [role1] [role2]	Role names may be entered but are not displayed

ORM 2	ORM 1
Internal Uniqueness Constraints *Unary:* or *Binary:* *n:1* *1:n* *1:1* *m:n* *UC on role pair 1-2* *UC on role pair 1, 3* Many UC combinations are possible *Preferred uniqueness:* ORM 2 enables display of preferred uniqueness constraints on *n*-aries to be toggled on/off.	or P P
Role Connection A Role is played only by *A*	A
External Uniqueness Constraints A B_1 B_n Each B_1,…, B_n combination ($n > 1$) relates to only one instance of *A* *Preferred uniqueness:* A B_1 B_n	A B_1 B_n U A B_1 B_n P

	ORM 2	ORM 1
Mandatory Role Constraints	*Simple:*  Role is mandatory for population of *A* *Disjunctive (inclusive-or constraint):*  Each instance in the population of *A* plays at least one of the *n* attached roles (*n* > 1). Role numbers are not displayed.	 
Objectification	 Fact type is objectified as object type *A*. ORM 2 allows any fact type to be objectified.	ORM 1 does not support objectified unaries. ORM 1 allows objectification only if there is a spanning UC or the UC pattern is 1:1 
Object Value Constraints	*Enumeration* *Range*  *Semi-bounded discrete range* { a.. } { ..a } *Bounded continuous range* {[a1 .. a2]} *includes both end values* {(a1 .. a2)} *excludes both end values* {[a1 .. a2)} *includes first value* {(a1 .. a2]} *includes last value* Combinations are allowed.	 { a.. } { ..a } {a1..a2} ORM1 does not support exclusion of any end values Combinations are allowed.
Role Value Constraints	 Same patterns as above	Not supported

	ORM 2	ORM 1
Subset Constraints *Simple:*  Each object that plays role 1 also plays role 2		Same
Contiguous Role-pair:  Each object pair that plays the role sequence 1.1, 1.2 also plays the role sequence 2.1, 2.2		Same
Other cases:  Each object tuple that plays the role sequence 1.1, 1.2, 1.3 also plays the role sequence 2.1, 2.2, 2.3		Same
ORM 2 also displays subset constraints over join paths		ORM 1 does not display subset constraints over join paths
Equality Constraints *2 role-sequences (of 1 or more roles):*  Populations of role-sequences must be equal		Same
3 or more role-sequences: e.g.  		Not supported (instead use multiple 2-sequence constraints)
Exclusion Constraints  Populations of 2 or more role-sequences must be mutually exclusive		Same

	ORM 2	ORM 1
Exclusive-Or Constraints	 Each instance in *A*'s population plays exactly one of the *n* attached roles (*n* > 1)	Same
Subtyping	 *B* is a proper subtype of *A* (its primary supertype) and *C* (a secondary supertype)	Same
	 *Exclusive* *Total* *Partition*	ORM 1 does not display constraints over subtyping connections
Frequency Constraints	 Each instance that plays role 1 does so *f* times	Same
	 Each instance pair that plays roles 1, 2 does so *f* times	
	 Each instance pair that plays roles 1, 2 does so *f* times	Same
	The frequency specification *f* may be any of the following	
	n exactly *n* (a positive integer) ≥ *n* at least *n* ≤ *n* at most *n* *n..m* at least *n* and at most *m*	*n* >=*n* <=*n* n..m

	ORM 2	ORM 1
Ring Constraints		
Value-comparison Constraints		Not supported
Derived Fact Types * = derived, ** = derived and stored + = semi-derived		Same for first two options. 3rd option not supported.
Deontic Constraints Deontic constraints are colored blue rather than violet. Most include an “o” for “obligatory”. Deontic ring constraints instead use dashed lines.		No deontic constraints are supported