FREIE UNIVERSITÄT BOZEN
LIBERA UNIVERSITÀ DI BOLZANO
FREE UNIVERSITY OF BOZEN - BOLZANO

KRDB RESEARCH CENTRE
KNOWLEDGE REPRESENTATION MEETS DATABASES

Faculty of Computer Science, Free University of Bozen-Bolzano, Piazza Domenicani 3, 39100 Bolzano, Italy
Tel: +39 04710 16000, fax: +39 04710 16009, http://www.inf.unibz.it/krdb/

KRDB Research Centre Technical Report:

Mapping the Object-Role Modeling language ORM2 into Description Logic language $\mathcal{DLR}_{ifd}$

C. Maria Keet

Affiliation	KRDB Research Centre, Faculty of Computer Science Free University of Bozen-Bolzano Piazza Domenicani 3, 39100 Bolzano, Italy
Corresponding author	Maria Keet keet@inf.unibz.it
Keywords	$\mathcal{DLR}$, Description Logics for conceptual modeling Object-Role Modeling, ORM2, formal conceptual modeling
Number	KRDB07-2
Date	February 15, 2007
URL	http://www.inf.unibz.it/krdb/pub/ (http://arxiv.org/abs/cs.LO/0702089v1)

Mapping the Object-Role Modeling language ORM2 into Description Logic language $\mathcal{DLR}_{ifd}$

C. Maria Keet
Faculty of Computer Science Free University of Bozen-Bolzano, Italy
keet@inf.unibz.it
Report Number: KRDB07-2

February 15, 2007

Abstract

ORM conceptual modellers are deprived of the advantages of automated reasoning over their representations of the Universe of Discourse, which could be addressed by DL reasoners. DLs are not considered user-friendly and could benefit from the easy to use ORM diagrammatic and verbalization interfaces. In addition, it would greatly expand the scope for automated reasoning with additional scenarios to improve quality of software systems. A mapping is proposed from the very expressive formal conceptual modelling language ORM2 to the Description Logic language $\mathcal{DLR}_{ifd}$. Given the many extant DL languages and none is as expressive as ORM or ORM2, the 'best-fit' $\mathcal{DLR}_{ifd}$ was chosen. For the non-mappable constraints, pointers to other DL languages are provided, which could serve as impetus for research into DL language extensions or interoperability between the extant languages.

1 Introduction

Description Logic (DL) languages have been shown useful for reasoning both over conceptual models like ER and UML [Artale *et al.* (2003), Baader *et al.* (2003)] [Calvanese *et al.* (1998), Berardi *et al.* (2005)]) and ontology languages such as OWL-DL, OWL-Lite [5], its proposed successor OWL 1.1 [4] that is based on the DL language $\mathcal{SROIQ}$ [Horrocks *et al.* (2006)], and *DL-Lite* [Calvanese *et al.* (2005)]. In particular, we are interested in the notion of using DLs as unifying paradigm for conceptual modelling to enable automated reasoning over conceptual models which, be it due to legacy, preference, or applicability, are made in different conceptual modelling languages. A tool such as iCOM [Franconi and Ng (2000), 1] already supports automated reasoning over UML or EER diagrams, which may have cross-conceptual model assertions. What is lacking, however, is a mapping from Object-Role Modeling (ORM) into a DL. One may wonder: why yet another mapping? There are three main reasons for this.

First, ORM is a so-called "true" conceptual modelling language in the sense that it is independent of the implementation and application scenario and has been mapped into both UML class diagrams and ER. That is, ORM and its successor ORM2[1] can be used in the conceptual analysis stage for database development, application software development, requirements engineering only, website development, business rules, and other areas, *e.g.*, [Balsters *et al.* (2006), Bollen (2006), Evans (2005), Halpin (2001), Hoppenbrouwers *et al.* (2005), Pepels and Plasmeijer (2005), de Troyer *et al.* (2005)]. Thus, if there is an ORM-DL mapping, the possible uses of automated reasoning scenarios —hence, improvement of software quality— is greatly expanded.

Second, an important aspect of ORMing is to have great consideration for the user and therefore ORM tools are very user-friendly, so that even domain experts unfamiliar with formalisms can start modelling after half an hour training. Furthermore, ORM tools have both diagrammatic and textual interfaces (the latter through so-called verbalizations, which are pseudo-natural language renderings of the axioms), thereby accommodating different user preferences.

Third, ORM is more expressive than either UML or ORM and, as will become clear from the mapping, is more expressive than the extant DLs as well. Most ORM constraints are supported in one DL language or another, but none supports all ORM constraints. The ORM-to-$\mathcal{DLR}_{ifd}$ mapping proposed in this report may provide some élan to examine DL language extensions not only based on interest and particular user requests from domain-modelling scenarios, but toward those (combinations of) extensions which are already known to be useful and are being used in the conceptual modelling community, or to find an implementable solution where for different (sections of) conceptual models, different languages can be used within one application interface.

The remainder of this report is organised as follows. Subsections 1.1 and 1.2 contains brief introduction to ORM and Description Logics, respectively. The main part is devoted to the mapping table in Section 2, which contains the ORM2 formalisms with its equivalent representation in $\mathcal{DLR}_{ifd}$ and pointers for the non-mappable constraints to possible options in non-$\mathcal{DLR}_{ifd}$ DL languages. Finally, some reflections and conclusions are included in Section 3.

1.1 Brief introduction to Object-Role Modeling

The basic building blocks of the Object-Role Modeling language are object types, value types, roles —where at the conceptual level no subjective distinction has to be made between classes and attributes—and a wide range of constraints. A role is that what the object type 'plays' in the relation. ORM supports n-ary relations, where n is a finite integer ≥ 1 (hence, unary relations are supported as well). ORM models can be mapped into, among others, ER and UML diagrams, IDEFX logical models, SQL table definitions, C, Visual Basic, and XML serialised. More information on these mappings can be found in *e.g.* [Halpin (2001), 3].

As preliminary for the mapping of ORM into $\mathcal{DLR}$, the basics can be summarised as follows: an n-ary predicate (relation) R, with $n \geq 1$, is composed of $r_1, ..., r_n$ roles, and each role has a relation to its object type, denoted with $C_1, ..., C_n$. There are

[1] The recently introduced ORM2 with beta-CASE tool NORMA [Halpin (2005b), 2] extends ORM with, among others, role value constraints and deontic rules.

lexical object types (LOT), also called value types such as string and number, and non-lexical object types (NOLOT).

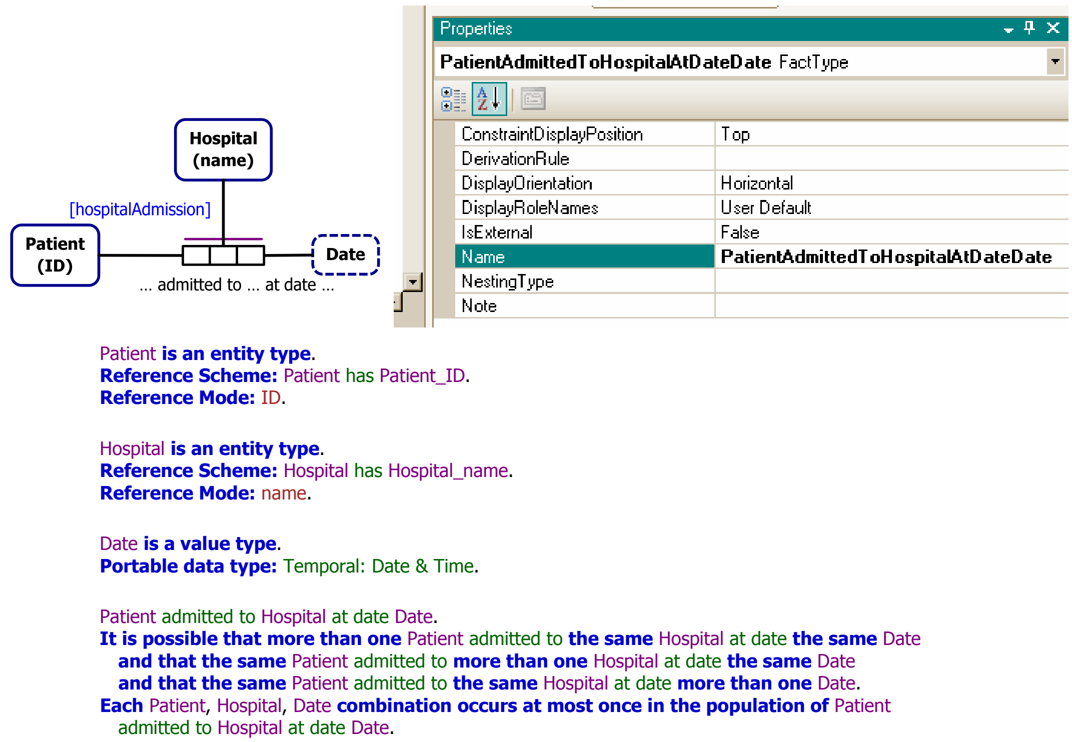

Figure 1: Top left: small ORM2 conceptual model, depicting two object types, a value type, a ternary relation, label for the reading, and name of the first role in "[ ]"; top-right: properties box of the fact type, displaying the name of the relation; bottom-half: verbalization of the fact type, its object and value types, and spanning uniqueness constraint (line above the box).

Halpin's first order logic formalization [Halpin (1989)] is included in the second column of the mapping in the table below; some of the 'long' formalisms can be simplified, which is omitted for now. Other formalizations of ORM exists, such as those from [Hofstede *et al.* (1993), Hofstede and Proper, (1998), Campbell *et al.* (1996)], which do not differ significantly from Halpin's version except that they make clearer distinctions between the role labels, their semantics, and predicate name, which makes it easier to demonstrate the objectification (reification, nesting) that is necessary in $\mathcal{DLR}_{ifd}$ for several constraints (*e.g.*, to properly specify multi-role uniqueness constraints that translate to primary keys in logical models based on ER or UML class diagrams). The naming & labelling is demonstrated in Figure 1, which was made with the NORMA CASE tool [2]: the diagrammatic representation of the relation in the conceptual model has

⋆ a label attached to the relation (rectangle divided into three roles, one for each participating object or value type), "... admitted to ... at date ...", which is used for the verbalization of the fact type (fixed-syntax pseudo-natural language sentences),

⋆ role names, such as "[hospitalAdmission]" for the the role that object type Patient plays, and

⋆ the name of the relation, which is displayed in the properties box of the relation and is generated automatically by the software (called "PatientAdmittedToHospitalAtDateDate" in the example).

1.2 Brief introduction to Description Logics

Description Logics (DL) languages are decidable fragments of first order logic and used for logic-based knowledge representation. The appropriate DL language to represent the information of the Universe of Discourse depends on requirements what the user wants to represent and what she wants to do with the knowledge base system (KBS). Basic ingredients of all DL languages are concepts (classes / entity types / object types / universals) and roles (/relations / predicates / associations)[2], where a DL role is an n-ary predicate where $n \geq 2$ (although in most DL languages $n = 2$). In addition, there is a set of supported constructors, which varies among the DL languages, to give greater or lesser expressivity and efficiency of automated reasoning over the logical theory. Usage in the KBS is split into a Terminological Box (TBox) that contains statements at the class-level and an ABox that contains assertions about instances. A TBox corresponds to a formal conceptual data model or, depending on the aim of the logical theory, can be used to represent an ontology.

Table 1: Non-exhaustive list of several constructors, DL syntax, and their semantics, where C is a concept (class) and R is a role (relation) (see also [Baader *et al.* (2003)]).

Name	DL syntax	Semantics
Top concept	$\top$	$\Delta^{\mathcal{I}}$
Bottom concept	$\bot$	$\emptyset$
Concept	C	$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
Concept disjunction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
Concept conjunction	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
Concept negation	$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
Universal restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}((x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\}$
Existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}((x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}})\}$
Subclass of	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
Subproperty of	$R_1 \sqsubseteq R_2$	$R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$
Equivalent class	$C_1 \equiv C_2$	$C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$
Equivalent property	$R_1 \equiv R_2$	$R_1^{\mathcal{I}} = R_2^{\mathcal{I}}$

The formal semantics of each DL language follows the usual notion of interpretation, $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the interpretation function $\cdot^{\mathcal{I}}$ assigns to each concept C

[2]Ontologically, the synonyms for concepts and roles do not necessarily hold exactly, and therefore are for indicative purpose only.

a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and to each relation R of arity n a subset $R^{\mathcal{I}}$ of $(\Delta^{\mathcal{I}})^n$. Table 1 shows the semantics for several often-used constructors; more introductory information about DL can be found in [Baader and Nutt (2003)], and usages and extension in [Baader *et al.* (2003)].

1.2.1 DL for conceptual modelling languages: $\mathcal{DLR}_{\textit{ifd}}$

I introduce first $\mathcal{DLR}$ [Calvanese and De Giacomo (2003)], and subsequently the "$_{ifd}$" extension for identity and functional dependence [Berardi *et al.* (2005)]. Take atomic relations ($\mathbf{P}$) and atomic concepts A as the basic elements of $\mathcal{DLR}$. We then can construct arbitrary relations with arity ≥ 2 and arbitrary concepts according to the following syntax:

$$\mathbf{R} \longrightarrow \top_n | \ \mathbf{P} \mid (\$i/n : C) \mid \neg\mathbf{R} \mid \mathbf{R}_1 \sqcap \mathbf{R}_2$$
$$C \longrightarrow \top_1 | \ A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists[\$i]\mathbf{R} \mid \leq k[\$i]\mathbf{R}$$

i denotes a component of a relation; if components are not named, then integer numbers between 1 and n_{max} are used, where n is the arity of the relation. k is a non-negative integer for multiplicity (cardinality). Only relations of the same arity can be combined to form expressions of type $\mathbf{R}_1 \sqcap \mathbf{R}_2$, and $i \leq n$, *i.e.* the concepts and relations must be well-typed.

The semantics of $\mathcal{DLR}$ is specified through the usual notion of interpretation, where $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and the interpretation function $\cdot^{\mathcal{I}}$ assigns to each concept C a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and to each n-ary $\mathbf{R}$ a subset $\mathbf{R}^{\mathcal{I}}$ of $(\Delta^{\mathcal{I}})^n$, s.t. the following conditions are satisfied:

$$\begin{aligned}
\top_n^{\mathcal{I}} &\subseteq (\Delta^{\mathcal{I}})^n \\
\mathbf{P}^{\mathcal{I}} &\subseteq \top_n^{\mathcal{I}} \\
(\neg\mathbf{R})^{\mathcal{I}} &= \top_n^{\mathcal{I}} \setminus \mathbf{R}^{\mathcal{I}} \\
(\mathbf{R}_1 \sqcap \mathbf{R}_2)^{\mathcal{I}} &= \mathbf{R}_1^{\mathcal{I}} \cap \mathbf{R}_2^{\mathcal{I}} \\
(\$i/n : C)^{\mathcal{I}} &= \{(d_1, ..., d_n) \in \top_n^{\mathcal{I}} | d_i \in C^{\mathcal{I}}\} \\
\top_1^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
A^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\exists[\$i]\mathbf{R})^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} | \exists(d_1, ..., d_n) \in \mathbf{R}^{\mathcal{I}}.d_i = d\} \\
(\leq k[\$i]\mathbf{R})^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} || \{(d_1, ..., d_n) \in \mathbf{R}_1^{\mathcal{I}} | d_i = d| \} \leq k\}
\end{aligned}$$

$\top_1$ denotes the interpretation domain, $\top_n$ for $n \geq 1$ denotes a subset of the n-cartesian product of the domain, which covers all introduced n-ary relations. Consequently, the "$\neg$" on relations mean the difference of relations rather than the complement. The $(\$i/n : C)$ denotes all tuples in $\top_n$ that have an instance of C as their i-th component. $\mathcal{DLR}$ is a proper generalization of $\mathcal{ALCQI}$, where the usual DL constructs can be reexpressed in $\mathcal{DLR}$ as:

$$\begin{array}{rcl} \exists P.C & \text{as} & \exists[\$1](P \sqcap (\$2/2 : C)) \\ \exists P^{-}.C & \text{as} & \exists[\$2](P \sqcap (\$1/2 : C)) \\ \forall P.C & \text{as} & \neg\exists[\$1](P \sqcap (\$2/2 : \neg C)) \\ \forall P^{-}.C & \text{as} & \neg\exists[\$2](P \sqcap (\$1/2 : \neg C)) \\ \leq kP.C & \text{as} & \leq k[\$1](P \sqcap (\$2/2 : C)) \\ \leq kP^{-}.C & \text{as} & \leq k\exists[\$2](P \sqcap (\$1/2 : C)) \end{array}$$

The following abbreviations can be used:
- $C_1 \sqcup C_2$ for $\neg(\neg C_1 \sqcap \neg C_2)$
- $C_1 \Rightarrow C_2$ for $\neg C_1 \sqcup C_2$
- $(\geq k[i]R)$ for $\neg(\leq k-1[i]R)$
- $\exists[i]R$ for $(\geq 1[i]R)$
- $\forall[i]R$ for $\neg\exists[i]\neg R$
- $R_1 \sqcup R_2$ for $\neg(\neg R_1 \sqcap \neg R_2)$
- $(i/n : C)$ is abbreviated to $(i : C)$ where n is clear form the context

$\mathcal{DLR}_{ifd}$ also supports identification assertions on a concept C, which has the form

$$(\mathbf{id}\ C[i_1]R_1, ..., [i_h]R_h)$$

where each R_j is a relation and each i_j denotes one component of R_j. Then, if a is an instance of C that is the i_j-th component of a tuple t_j of R_j, for $j \in \{1, ..., h\}$, and b is an instance of C that is the i_j-th component of a tuple s_j of R_j, for $j \in \{1, ..., h\}$, and for each j, t_j agrees with s_j in all components different from i_j, then a and b are the same object.

$\mathcal{DLR}_{ifd}$ supports functional dependency assertions on a relation R to deal with operations, which has the form

$$(\mathbf{fd}\ R\ i_1, ..., i_h \rightarrow j)$$

where $h \geq 2$, and $i_1, ..., i_h, j$ denote components of R. Last, there are notational variants
- Set difference for R, where the "$\dot{\neg}$" can be used to distinguish it from normal negation (complement).
- dropping the "\$" before the i
- "$t[i]$" for the i-th component of tuple t, s.t. one can rewrite
  $(\$i/n : C)^{\mathcal{I}} = \{(d_1, ..., d_n) \in \top_n^{\mathcal{I}} | d_i \in C^{\mathcal{I}}\}$
  with the previous point into
  $(i/n : C)^{\mathcal{I}} = \{t \in \top_n^{\mathcal{I}} | t[i] \in C^{\mathcal{I}}\}$
- Use $\sharp S$ to denote the cardinality of the set S, s.t. one can rewrite
  $(\leq k[\$i]\mathbf{R})^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} || \{(d_1, ..., d_n) \in \mathbf{R}_1^{\mathcal{I}} | d_i = d|\} \leq k\}$
  with the second and third point into
  $(\leq k[i]\mathbf{R})^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} | \sharp\{t \in \mathbf{R}_1^{\mathcal{I}} | t[i] = d\} \leq k\}$

1.2.2 Other relevant DL languages

Given the above-mentioned details on $\mathcal{DLR}_{ifd}$ which adds the identification (primary key) and functional dependency (UML method or ORM derived-and-stored relation), the other three variations [Calvanese and De Giacomo (2003)] are straightforward. $\mathcal{DLR}_{\mu}$ supports fixpoint constructs for recursive structures over single-inheritance trees of a role (*i.e.*, acyclicity) [Calvanese *et al.* (1999)] and thereby also supports transitivity asymmetry and (ir)reflexivity. $\mathcal{DLR}_{reg}$ adds support for regular expressions over roles (including the role composition operator and reflexive transitive closure) [Calvanese *et al.* (1998)], and $\mathcal{DLR}_{\mathcal{US}}$ adds the $\mathcal{U}$ntil and $\mathcal{S}$ince operators for temporal EER [Artale *et al.* (2002)]. It has not been investigated if combining $\mathcal{DLR}_{ifd}$, $\mathcal{DLR}_{reg}$, and $\mathcal{DLR}_{\mu}$ remains within EXPTIME or leads to undecidability.

In the other direction toward DL-based ontology languages, there are OWL and draft OWL 1.1 [4], which are based on the DLs $\mathcal{SHOIN}$ (for OWL-DL), $\mathcal{SHIF}$ (OWL-Lite), and $\mathcal{SROIQ}$, respectively. $\mathcal{SROIQ}$ also supports local (ir)reflexivity and antisymmetry (currently not supported by any $\mathcal{DLR}$), and transitive roles. On the other hand, $\mathcal{SROIQ}$ does not support acyclic roles, not datatypes, neither "**id**" nor the "**fd**", and no 'access' to elements of a DL-role.

Rarely, if ever, are all ORM constraints used in one conceptual model. Given this, it will be more effective to take the same approach as that of the Protégé ontology development tool: let the user model what s/he wants, and determine the (sub-)language based on the constructors used, instead of covering all theoretical combinations. In addition, at the time of writing, there are still differences between theoretically computationally feasible and implemented features in reasoners like Racer, Pellet, and FaCT. (E.g. although EER and UML are mapped to $\mathcal{DLR}$, the reasoner in the iCOM tool [Franconi and Ng (2000), 1] uses $\mathcal{SHIQ}$ through an additional transformation step from $\mathcal{DLR}$.)

With the basic introduction of ORM and the semantics and notation of $\mathcal{DLR}_{ifd}$, which supports most ORM constructors, we can proceed to the mapping from ORM into $\mathcal{DLR}_{ifd}$. Corresponding graphical notation of ORM components and constrains are included in the four figures after the table.

2 Mapping

The mapping contains all components and constraints of ORM2, hence also of ORM, except deontic constraints that were recently added to ORM2. The formalisation in the second column has been taken from [Halpin (1989)], where available, which was the first formalisation of ORM. All underlined text in the third column with the mapping to $\mathcal{DLR}_{ifd}$ indicates that $\mathcal{DLR}_{ifd}$ does <u>not</u> support that particular constraint; *i.e.*, that constraint has a problem that need to be resolved, it permits only a partial mapping, or requires additional constraints for it to be mapped into $\mathcal{DLR}_{ifd}$. Finally, diagrammatic representations of the elements and constraints are shown in Figures 2-6 at the end of this document (made with VisioModeler 3.1).

Nr.	ORM component or constraint	$\mathcal{DLR}_{ifd}$ equivalent
1	**Object type** $\forall x C(x)$	$C \sqsubseteq \forall[r_i]R$
2	**Unary relation** $\forall x(R(x) \rightarrow C_i(x))$ note that here the graphical notation collapses where the name of a role r_i is equivalent to the name of the unary predicate R	$R \sqsubseteq (r_1 : C_i)$
3	**Binary relation** $\forall x, y(R(x, y) \rightarrow C_i(x) \wedge C_j(y))$	$R \sqsubseteq (r_i : C_i) \sqcap (r_j : C_j)$
4	***n*-ary relation** $\forall x_1, ..., x_n(R(x_1, ..., x_n) \rightarrow C_1(x_1) \wedge ... \wedge C_n(x_n))$	$R \sqsubseteq (r_1 : C_1) \sqcap ... \sqcap (r_n : C_n)$ or, in short: $R \sqsubseteq \sqcap_{i=1}^{n}(r_i : C_i)$
5	**Named value type** (data type, or lexical type), which permits values of some set $\{v_1, ..., v_n\}$ where the values are *not* constrained, and the value type C_j $\forall x(C_j(x) \equiv x \in \{v_1, ..., v_n\})$	$C_i \sqsubseteq \forall[r_i]R(R \Rightarrow (r_j : C_j)$ s.t. for each instance c of C_i, all objects related to c by R are instances of C_j. Note that the domain of the value type can be a user defined one, such as `String`, `Number`, etc.
6	**Named value type** (data type, or lexical type), where the values of C_j are *constrained* to specific values $\{v_1, ..., v_i\}$, and value type C_j $\forall x(C_j(x) \equiv x \in \{v_1, ..., v_i\})$	$C_i \sqsubseteq \forall[r_i]R(R \Rightarrow (r_j : C_j) \sqcap (C_j \equiv \{v_1, ..., v_i\})$ s.t. for each instance c of C_i, all objects related to c by R are instances of C_j and have a value v_1 or...or v_i. The domain of the value type can be a user defined one, such as `String`, `Number`, etc.; they are values, not objects (hence, not an enumerated class)
7	**Unnamed lexical type** in binary relation and constrained values to $\{v_1, ..., v_n\}$, then $\forall x, y(R(x, y) \rightarrow C_i(x) \wedge y = v_1 \vee ... \vee y = v_n)$	$C_i \sqsubseteq \forall[r_i]R(R \Rightarrow (r_j : C_j) \sqcap (C_j \equiv \{v_1, ..., v_n\})$ s.t. for each instance c of C_i, all objects related to c by R are instances of C_j and is one of elements in the specified set. Thus, the unnamed value type is assigned a default label (C_j in this case) in the mapping. The domain of the values can be a user defined one, such as `String`, `Number`, etc.
8	**Mandatory**, binary predicate: $\forall x(C(x) \rightarrow \exists y R(x, y))$ n-ary predicate with mandatory on role i and $i \leq n$: $\forall x_i(C_i(x_i) \rightarrow \exists x_1, ..., x_{i-1}, x_{i+1}, ..., x_n R(x_1, ..., x_n))$	$C_i \sqsubseteq \exists[r_i]R$
9	**Disjunctive mandatory** between the ith roles of n different relations, where $n \geq 2$, for m-ary relations and $i \leq m$ $\forall x(C(x) \rightarrow \exists x_1, ... x_{m-1}(R1(x_1, ..., x_{i_1-1}, x, x_{i_1+1}, ..., x_{m_1}) \vee ... \vee Rn(x_1, ..., x_{i_n-1}, x, x_{i_n+1}, ..., x_{m_n})))$	$C_i \sqsubseteq \exists[r_1]R_1 \sqcup \exists[r_1]R_2$ for disjunction of roles among n relations, each for the jth role with $j \leq n$ then $C_i \sqsubseteq \sqcup_{i=1}^{n} \exists[r_j]R_i$

10	**Uniqueness, 1:n**, binary relation $\forall x, y, z(R(x, y) \wedge R(x, z) \rightarrow y = z)$	$C_j \sqsubseteq (\leq 1[r_i]R)$
11	**Uniqueness, 1:1**, binary relation, which is built up from two single-role uniqueness constraints	$C_i \sqsubseteq (\leq 1[r_i]R)$ and $C_j \sqsubseteq (\leq 1[r_j]R)$
12	**Uniqueness, m:n** on a n-ary relation, $n \geq 2$, covering all n roles: repetition of a proposition does not have a logical significance, and is ignored [Halpin (1989)] p4-5, yet the case is included in the next constraint nr.13 when $i = n$	($\mathbf{id}$ $R[1]r_1, ..., [1]r_i$) over i roles in n-ary relation, $i = n$, and R is a reified (objectified) relation (see also nr.34 below)
13	**Uniqueness, n-ary** relation where $1 \leq j \leq n$, $n \geq 2$, uniqueness constraint spans at least $n-1$ roles (for it to be elementary), and j is not included in the uniqueness constraint $\forall x_1, ..., x_j, ..., x_n, y(R(x_1, ..., x_j, ..., x_n) \wedge (R(x_1, ..., y, x_{j+1}, ..., x_n) \rightarrow x_j = y)$	($\mathbf{id}$ $R[1]r_1, ..., [1]r_i$) over i roles in n-ary relation, $1 \leq i \leq n$, and R is a reified (objectified) relation (see also nr.34 below)
14	**External uniqueness** 1) among two roles: $\forall x_1, x_2, y, z(R1(x_1, y) \wedge R2(x_1, z) \wedge R1(x_2, y) \wedge R2(x_2, z) \rightarrow x_1 = x_2)$ 2) among m roles: $\forall x_1, x_2, y_1, y_m(R1(x_1, y_1) \wedge ... \wedge Rm(x_1, y_m) \wedge R1(x_2, y_1) \wedge ... \wedge Rm(x_2, y_m) \rightarrow x_1 = x_2)$	Remodel as n-ary relation, where $n = m + 1$ s.t. ($\mathbf{id}$ $R[1]r_1, ..., [1]r_m$) or one after the other with a natural join of the predicates $((C_j \sqsubseteq (\leq 1[r_j]R_1)) \sqcap ... \sqcap (C_m \sqsubseteq (\leq 1[r_j]R_m)))$, where $m \geq 2$
15	**Role frequency** with 1) exactly a times, $a \geq 1$ $\forall x(\exists y_1 R(x, y_1) \rightarrow \exists y_2, ..., y_a(y_1 \neq y_2 \wedge ... \wedge y_{a-1} \neq y_a \wedge R(x, y_2) \wedge ... \wedge R(x, y_a))) \wedge \forall x, y_1, ..., y_{a+1}(R(x, y_1) \wedge ... \wedge R(x, y_{a+1}) \rightarrow y_1 = y_2 \vee y_1 = y_3 \vee ... \vee y_a = y_{a+1})$ 2) at least a or 3) at most a times	1) $C_i \sqsubseteq (\geq a[r_i]R) \sqcap (\leq a[r_i]R)$ where $a \geq 1$ 2) $C_i \sqsubseteq (\geq a[r_i]R)$ $C_i \sqsubseteq (\leq a[r_i]R)$
16	**Role frequency** with at least a and at most b, $1 \leq a$ and $a \leq b$ $\forall x(\exists y_1 R(x, y_1) \rightarrow \exists y_2, ..., y_a(y_1 \neq y_2 \wedge ... \wedge y_{a-1} \neq y_a \wedge R(x, y_2) \wedge ... \wedge R(x, y_a))) \wedge \forall x, y_1, ..., y_{b+1}(R(x, y_1) \wedge ... \wedge R(x, y_{b+1}) \rightarrow y_1 = y_2 \vee y_1 = y_3 \vee ... \vee y_b = y_{b+1})$	$C_i \sqsubseteq (\geq a[r_i]R) \sqcap (\leq b[r_i]R)$ where $1 \leq a \leq b$ and $i \leq n$

17a	**Multi-role frequency** spanning 2 roles r_i and r_j in n-ary relation, with $n \geq 2$, and $1 \leq a \leq b$ $\forall x, y(\exists z_1 R(x, y, z_1) \rightarrow \exists z_2, ..., z_a(z_1 \neq z_2 \wedge ... \wedge z_{a-1} \neq z_a \wedge R(x, y, z_2) \wedge ... \wedge R(x, y, z_a))) \wedge \forall x, y, z_1, ..., z_{b+1}(R(x, y, z_1) \wedge ... \wedge R(x, y, z_{b+1}) \rightarrow z_1 = z_2 \vee z_1 = z_3 \vee ... \vee z_b = z_{b+1})$ This constraint can be used iff there is no uniqueness constraint over both r_i and r_j only. Given that an elementary fact type must have uniqueness over $n-1$ roles, then *either* 1) r_i or r_j is part of a single role uniqueness constraint but not both *2)* r_i or r_j is part of a multi-role uniqueness constraint but not both *or* 3) multi-role uniqueness includes r_i, r_j, and ≥ 1 other role in that relation (hence, $n \geq 3$) *or* 4) the relation is not an elementary fact type (because then the multi-role uniqueness spans $\leq n-2$ roles) and ought to be remodelled to be elementary	1) This implies that either i) $a = 1$ or ii) $b = 1$. For i) with r_i having the uniqueness constraint, then "a-b" reduces to $\leq b$ frequency on r_j only, for which the mapping 11a is valid. For option ii) then it has to be included in the uniqueness constraint, s.t. mapping nr.9 holds (*i.e.*, the frequency constraint is redundant) 2) E.g. for ternary relation with roles r_h, r_i, and r_j, uniqueness over (r_h, r_i) and frequency over (r_i, r_j), then uniqueness constraint can be reduced to r_h only. Then, see point 4 below. 3) E.g. for ternary relation with roles r_h, r_i, and r_j, uniqueness over (r_h, r_i, r_j) and frequency over (r_i, r_j), then uniqueness constraint can be reduced to r_h only. Then, see point 4 below. 4) N/A, because it depends on how it is remodelled, or it is not supported in $\mathcal{DLR}_{ifd}$ but only in the application software implemented. A *partial mapping* is possible, s.t. at least $C_i \sqsubset (\geq a[r_i]R)$ and $C_j \sqsubset (\geq a[r_j]R)$ hold
17b	**Multi-role frequency** spanning i roles of an n-ary relation, $i > 2$, and $i \leq n$ (TFC5 in [Halpin (1989)] p4-13). Assuming correct usage is possible, this constraint is rare, if used at all	See nr.17a: N/A or not supported.
18	**Proper subtype**, which holds for subsumption of either object types or value types, but which cannot be mixed (and note that at times their extensions may contain the same elements) $\forall x(D(x) \rightarrow C(x))$	$D \sqsubseteq C$ and $\neg(C \sqsubseteq D)$ (latter to ensure that the concepts D and C are never equivalent)
19	**Subtypes, total (exhaustive) covering** (not formalised in [Halpin (1989)])	$C \sqsubseteq D_1 \sqcup ... \sqcup D_n$, where the indexed concepts D are subtypes of C. In short: $C \sqsubseteq \sqcup_{i=1}^{n} D_i$
20	**Exclusive (disjoint) subtypes** (not formalised in [Halpin (1989)])	defined among the $1, ..., n$ subtypes: $D_i \sqsubseteq \sqcap_{j=i+1}^{n} \neg D_j$ for each $i \in \{1, ..., n\}$
20a	**Exclusive (disjoint) subtypes, total** (not formalised in [Halpin (1989)])	use both nr.19 and nr.20
21	**Subset over two roles** r_i in two n-ary relations R_j and R_i $\forall x(\exists y R_j(x, y) \rightarrow \exists z R_i(x, z))$	$[r_i]R_j \sqsubseteq [r_i]R_i$

22	**Subset over two n-ary relations**, for binary $\forall x, y(R_j(x, y) \rightarrow R_i(x, y))$ and more cumbersome for n-ary relation, an underlined variable like $\underline{x}$ is an abbreviation for a sequence $x_1, ..., x_n$ in an n-ary relation $\forall x, y(\exists\ \underline{z}\ (R_j(\underline{z})\ \wedge x = z_j \wedge y = z_{j+1}) \rightarrow \exists\ \underline{w}\ (R_i(\underline{w}) \wedge x = w_i \wedge y = w_{i+1}))$	$R_j \sqsubseteq R_i$
23	**Subset over k roles** in two n-ary relations, where $k < n$, abbreviation as in nr.22, and the corresponding roles must match in domain $\forall x_1, ...x_n(\exists\ \underline{y}\ (R_j(\underline{y})\ \wedge x_1 = y_{j_1} \wedge ... \wedge x_n = y_{j_n}) \rightarrow \exists\ \underline{z}\ (R_i(\underline{z}) \wedge x_1 = z_{i_1} \wedge ... \wedge x_n = z_{i_n}))$	$([r_1]R_j \sqsubseteq [r_1]R_i) \sqcap ([r_2]R_j \sqsubseteq [r_2]R_i) \sqcap ... \sqcap ([r_k]R_j \sqsubseteq [r_k]R_i)$ because it is a role-by-role subset constraint. This mapping does not say that the *combination* of the k roles in R_j is a subset of the combination of k roles in R_i, but given that the roles must be typed the same, it is acceptable. To get the latter in $\mathcal{DLR}_{\mathit{ifd}}$, I have to create two new relations R_b and R_a s.t. R_b consists of the k roles of R_j and R_a consists of the k roles of R_i, and then $R_b \sqsubseteq R_a$, but this can lead to undecidability cf. projections of the relations *unless* there is a uniqueness constraint over exactly those k roles.
24	**Set-equality over two roles** r_i in two n-ary relations R_j and R_i $\forall x(\exists y R_j(x, y) \equiv \exists z R_i(x, z))$	$[r_i]R_j \equiv [r_i]R_i$
25	**Set-equality over two n-ary relations** for binary $\forall x, y(R_j(x, y) \equiv R_i(x, y))$ for n-ary relation, abbreviation as in n.22 $\forall x, y(\exists\ \underline{z}\ (R_j(\underline{z})\ \wedge x = z_j \wedge y = z_{j+1}) \equiv \exists\ \underline{w}\ (R_i(\underline{w}) \wedge x = w_i \wedge y = w_{i+1}))$	$R_j \equiv R_i$
26	**Set-equality over k roles** in two n-ary relations, where $k < n$, abbreviation as in nr.22, and the corresponding roles must match in domain $\forall x_1, ...x_n(\exists\ \underline{y}\ (R_j(\underline{y})\ \wedge x_1 = y_{j_1} \wedge ... \wedge x_n = y_{j_n}) \equiv \exists\ \underline{z}\ (R_i(\underline{z}) \wedge x_1 = z_{i_1} \wedge ... \wedge x_n = z_{i_n}))$	$([r_1]R_j \equiv [r_1]R_i) \sqcap ([r_2]R_j \equiv [r_2]R_i) \sqcap ... \sqcap ([r_k]R_j \equiv [r_k]R_i)$ because it is a role-by-role equivalence, although this mapping does not say that the *combination* of the k roles in R_j is equivalent to the combination of k roles in R_i. To get the latter in $\mathcal{DLR}_{\mathit{ifd}}$, I create two new relations R_b and R_a s.t. R_b consists of the k roles of R_j and R_a consists of the k roles of R_i, and then $R_b \equiv R_a$, provided there is a uniqueness constraint over the k roles (see also nr.23)
27	**Role exclusion between two roles** r_i and r_j each in n-ary relations R_i and R_j (which do not necessarily have the same arity), in abbreviated form where the "$x \in A =_{def} A(x)$ and the "$R_i.r_i$" and "$R_j.r_j$" the r_i and r_j role in relation R_i and R_j, respectively, and $1 \leq i \leq n$ $\forall x \neg(x \in R_i.r_i \wedge x \in R_j.r_i)$	$[r_i]R_i \sqsubseteq \neg[r_j]R_j$ note this is role difference, not role negation

28	**Relation exclusion between two relations** R_i and R_j $\forall x, y \neg(\exists \underline{z}(R_i(\underline{z} \wedge x = z_i \wedge y = z_{i+1}) \wedge \exists\ \underline{w}\ (R_j(\underline{w}) \wedge x = w_j \wedge y = w_{j+1}))$	$R_i \sqsubseteq \neg R_j$ note this is role difference, not role negation
29	**Role exclusion over k roles** in two n-ary relations R_i and R_j $\forall x_1, ..., x_n \neg(\exists \underline{y}(R_i(\underline{y} \wedge x_1 = y_{i_1} \wedge ... \wedge x_n = y_{i_n}) \wedge \exists\ \underline{z}\ (\ R_j(\underline{z}) \wedge x_1 = z_{j_1} \wedge ... \wedge x_n = z_{j_n}))$	Analogous to nr.23 and nr.26 s.t. it has to be split-up into two constraints in $\mathcal{DLR}_{ifd}$, one for the individual exclusions among the pairs of roles, then if there is a uniqueness over the k roles, then exclusion among the two new k-ary relations $(([r_1]R_i \sqsubseteq \neg[r_1]R_j) \sqcap ... \sqcap [r_k]R_i \sqsubseteq \neg[r_k]R_j))$ $R_a \sqsubseteq \neg R_b$
30	**Role exclusion between n roles** $r_1, ..., r_n$ each one in an m-ary relation $R_1, ..., R_n$ (which do not necessarily have the same arity) $\forall x \neg((x \in R_1.r_1 \wedge x \in R_2.r_2) \vee (x \in R_1.r_1 \wedge x \in R_3.r_3) \vee ... \vee (x \in R_{n-1}.r_{n-1} \wedge x \in R_n.r_n))$	$([r_1]R_1 \sqsubseteq \neg[r_2]R_2) \sqcup ([r_1]R_1 \sqsubseteq \neg[r_3]R_3) \sqcup ... \sqcup ([r_{n-1}]R_{n-1} \sqsubseteq \neg[r_n]R_n)$
31	**Join-subset** among four, not necessarily distinct, relations R_i, R_j, R_k, R_l, where $R_i * R_j[c_i, c_j]$ is the projection on columns c_i and c_j of the natural join of R_i and R_j. Then with four distinct relations: $R_i * R_j[c_i, c_j]$ is the subset of $R_k * R_l[c_k, c_l]$ $R_i * R_j[c_i, c_j] \subseteq R_k * R_l[c_k, c_l]$ where the compared pairs must belong to the same type, like *e.g.* r_i of R_i and r_k of R_k might be played by C_a and r_j of R_j and r_l of R_l might be played by C_b. See also the example for 3 relations in nr.32	Extending nr.21 for subsets of two roles, this $([r_i]R_i \sqcap [r_j]R_j) \sqsubseteq ([r_k]R_k \sqcap [r_l]R_l)$ Reduces to query containment (see ch16 DL handbook [Baader *et al.* (2003), Calvanese *et al.* (1998), Calvanese *et al.* (1999)].
32	**Join-equality**, see nr.31 for notation, then 1) with four distinct relations $R_i * R_j[c_i, c_j] \equiv R_k * R_l[c_k, c_l]$ 2) Example with three distinct relations R_i, R_j, and R_k s.t. $\forall x, y(\exists z(R_j(z, x) \wedge R_k(z, y)) \equiv \exists w R_i(x, y, w))$	1) Extending nr.21 for subsets of two roles, this $([r_i]R_i \sqcap [r_j]R_j) \equiv ([r_k]R_k \sqcap [r_l]R_l)$ query containment in both directions, see nr.31 2) simpler version of 1) as $([r_j]R_j \sqcap [r_k]R_k) \equiv ([r_i]R_i \sqcap [r_j]R_i)$
33	**Join-exclusion**, see nr.31 for notation, then $R_i * R_j[c_i, c_j] \subseteq \neg R_k * R_l[c_k, c_l]$ See also the example for 3 relations in nr.32	Extending nr.21 for subsets of two roles $([r_i]R_i \sqcap [r_j]R_j) \sqsubseteq \neg([r_k]R_k \sqcap [r_l]R_l)$ see also nr.31

34	**Objectification** (nesting, reification), full uniqueness constraint over the n roles of the n-ary relation, and R_o is the objectified relation of R $\forall x(R_o(x) \equiv \exists x_1,...,x_n(R(x_1,...,x_n) \wedge x = (x_1,....,x_n)))$ see also table footnote 1.	$R \sqsubseteq \exists[1]r_1 \sqcap (\leq 1[1]r_1) \sqcap \forall[1](r_1 \Rightarrow (2 : C_1))\sqcap$ $\exists[1]r_2 \sqcap (\leq 1[1]r_2) \sqcap \forall[1](r_2 \Rightarrow (2 : C_2))\sqcap$ $\vdots$ $\exists[1]r_n \sqcap (\leq 1[1]r_n) \sqcap \forall[1](r_n \Rightarrow (2 : C_n))$ where the $\exists[1]r_i$ (with $i \in, \{1,...,n\}$) specifies that concept R must have all components $r_1,...,r_n$ of the relation R, $(\leq 1[1]r_i)$ (with $i \in \{1,...,n\}$) specifies that each such component is single-valued, and $\forall[1](r_i \Rightarrow (2 : C_i))$ (with $i \in \{1,...,n\}$) specifies the class each component has to belong to.
35	**Derived fact type**, implied by the constraints of the roles from which the fact is derived, *i.e.* the original fact types and derived fact type relate through $\leftrightarrow$	Implied by the constraints of the roles from which the fact is derived, hence N/A
36	**Derived-and-stored** fact type, or conditional derivation, where the predicate indicates that the derivation rule provides only a partial definition of the predicate, *i.e.* the original fact types and derived fact type relate through $\rightarrow$	A derived-and-stored derivation rule maps to $\mathcal{DLR}_{ifd}$'s **fd**. With m parameters belonging to the classes $P_1,...P_m$ (the known part of the partial definition of the predicate) and the result belongs to R (the computed 'unknown' part of the partial definition of the predicate), then we have the relation $f_{P_1,...,P_m}$ with arity $1+m+1$, then $f_{P_1,...,P_m} \sqsubseteq (2 : P_1) \sqcap ... \sqcap (m+1 : P_m)$ (**fd** $f_{P_1,...,P_m} 1,...,m+1 \rightarrow m+2$) $C \sqsubseteq \forall[1](f_{P_1,...,P_m} \Rightarrow (m+2 : R))$ note that for a derivation rule $m \geq 1$
37	**Intransitive** (ring) constraint $\forall x,y,z(R(x,y) \wedge R(y,z) \rightarrow \neg R(x,z))$	DL roles (relations) are intransitive by default
38	**Antisymmetry** ring constraint (not formalised in [Halpin (1989)]) $\forall x,y(R(x,y) \wedge R(y,x) \rightarrow x = y)$ or, as in [Halpin (2001)] $\forall x,y(\neg(x = y) \wedge R(x,y) \rightarrow \neg R(y,x))$	Need to <u>add</u> antisymmetry to $\mathcal{DLR}$ (to check if the tableau algorithm has an automata-based counterpart). Note that this antisymmetry is limited to the $\mathcal{SROIQ}$ antisymmetry which implies irreflexivity; the more generic one of reflexive antisymmetry is an <u>open issue</u> [Horrocks *et al.* (2006)].
39	**Irreflexive** (ring) constraint on binary relation $\forall x \neg(R(x,x))$ Note that an irreflexive, functional relations (like a binary with 1:n uniqueness constraint) must be intransitive	<u>open issue</u> for $\mathcal{DLR}_{ifd}$, but should be possible. Is possible with $\mathcal{DLR}_{\mu}$ thanks to least/greatest fixpoint and in $\mathcal{SROIQ}$ with `Self` [Horrocks *et al.* (2006)] (irreflexive: $\top \sqsubseteq \neg\exists R.Self$, and reflexive for *simple* roles R then: $\top \sqsubseteq \exists R.Self$).
40	**Acyclic** (ring) constraint (not formalised in [Halpin (1989)]) where an x cannot be directly, or indirectly through a chain, related to itself. Acyclicity implies asymmetry, which in turn implies irreflexivity and antisymmetry. Recursive definition in [Halpin (2001)]: R is acyclic iff $\forall x \neg(x$ has path to $x)$. I consider acyclicity on two roles of an arbitrary relation and acyclicity on a ring constraint with one object type	Can add this to $\mathcal{DLR}_{ifd}$ with the repeat PDL (transitive closure of roles, R^+ of the role, *i.e.* $\bigcup_{n \geq 1}(R^{\mathcal{I}})^n$) using the least fixpoint construct $\mu X.C$, which is in DL syntax [Calvanese *et al.* (1999), Calvanese and De Giacomo (2003)]: $\exists R^*.C = \mu X(C \sqcup \exists R.X)$ which should work, but <u>verify</u> that a "$\mathcal{DLR}_{\mu ifd}$" is ok.

41	**Symmetric** ring constraint (not formalised in [Halpin (1989)] but in [Halpin (2001)]) $\forall x, y(R(x,y) \rightarrow R(y,x))$	Not supported in any of the $\mathcal{DLR}$s. $R \sqsubseteq R^{-}$ is supported in $\mathcal{SROIQ}$ [Horrocks *et al.* (2006)].
42	**Asymmetric** ring constraint $\forall x, y(R(x,y) \rightarrow \neg R(y,x))$	$R \sqsubseteq \neg R^{-}$ is not supported in $\mathcal{DLR}_{ifd}$, but asymmetry is supported in $\mathcal{DLR}_{\mu}$ through the stronger notion of well-foudnedness ($\top \sqsubseteq \neg\Delta R$).
43-I	***ac* and *it***, intersecting acyclicity and intransitivity	Only if nr.40 is possible
43-II	***ans* and *it***, intersecting antisymmetry with intransitivity	Only if ans (nr.38) can be done with automata-based technique (with intransitivity then we have the irreflexive antisymmetry)
43-III	***it* and *sym***, intersecting intransitivity and symmetry	nr.37 and nr.41: not supported because of nr.41
43-IV	***ir* and *sym***, intersecting irreflexivity and symmetry	No, even if nr.39 can be fixed for some $\mathcal{DLR}_{\mu ifd}$, then nr.41 is still a problem.
44	**Role value constraint**, where the object type C_i only participates in role r_i if an instance has any of the values $\{v_i, ...v_n\}$, with binary relation then $\forall x, y(x \in \{v_i, ...v_n\} \rightarrow (R(x,y) \rightarrow C_i(x) \wedge C_j(y)$ (a new constraint in ORM2)	This may be mapped using several approaches, where the easiest is to create new subtype C_i' for the set of values to which the role is constrained, where the value can be any of $\{v_i, ...v_n\}$, and let C_i' play the role, s.t. $C_i' \sqsubseteq C_i$ and $C_i' \sqsubseteq \forall[r_i]R$ but does not address it fully yet, therefore use nr.6 for the value constraints on C_i'. Or try role values, though note that role values are currently supported inly in *DL-Lite*$_{\mathcal{A}}$ [Calvanese *et al.* (2006)].

[1] ORM allows objectification of fact types if it either has a spanning uniqueness or is a binary fact type with 1:1 uniqueness [Halpin (2003)]. This restriction has been relaxed for ORM2: "A fact type may be objectified only if: (a) it has only a spanning uniqueness constraint; or (b) its uniqueness constraint pattern is likely to evolve over time (*e.g.* from n:1 to m:n, or m:n:1 to m:n:p); or (c) it has at least two uniqueness constraints spanning n-1 roles (n ¿ 1), and there is no obvious choice as to which of the n-1 role uniqueness constraints is the best basis for a smaller objectification based on a spanning uniqueness constraint; or (d) the objectification significantly improves the display of semantic affinity between fact types attached to the objectified type." [Halpin (2005a)].

3 Discussion and Conclusions

As is clear from the mapping table, the ORM ring constraints / DL-role meta-properties are most problematic for $\mathcal{DLR}_{ifd}$, but most of them can be met by $\mathcal{DLR}_{\mu}$ or $\mathcal{SROIQ}$. On the other hand, $\mathcal{DLR}_{\mu}$ and $\mathcal{SROIQ}$ do not have a constructor for primary keys that $\mathcal{DLR}_{ifd}$ does have, and $\mathcal{SROIQ}$ does not support n-ary relations where $n > 2$. It is for these reasons that $\mathcal{DLR}_{ifd}$ was chosen. Also, while data types are supported in $\mathcal{SROIQ}(D)$, the notion of role values is not, but which is recently introduced with $DL\text{-}Lite_{\mathcal{A}}$. Regardless the DL language, reflexive antisymmetry is an open problem for all DLs and no language will support multi-role frequency (nr.17a & 17b), because on its own it leads to undecidability. Scenarios nr.17a 1-3 can be mapped into $\mathcal{DLR}_{ifd}$ only when respecting 'proper' ORMing where the requirement for so-called elementary fact types is enforced (meaning that the uniqueness constraint over an n-ary relation must span at least n-1 roles).

Summarizing, most ORM2 constructs and constraints can be mapped into $\mathcal{DLR}_{ifd}$, which already could be used for a wide range of ORM models that do not use ORM's full expressive capabilities; *e.g.*, to do model checking, compute derived relations, and classification (and, hence, finding inconsistencies). Conversely, when the present mapping is implemented, DLs have a pleasant user interface enabling domain experts to take part in representing their Universe of Discourse. Several approaches are possible to narrow th gap between ORM2 and DL languages, where a "$\mathcal{DLR}_{\mu ifd}$" or $\mathcal{SROIQ}$ with n-ary relations seem close by. Alternatively, if this leads to undecidability, one could investigate possibilities for certain modularizations where a large model can be split-up into sections (ideally, hidden from the modeller) and perform the reasoning services on the separate subsections with different reasoner software.

Acknowledgments
The author gratefully acknowledges Diego Calvanese for his helpful comments on meta-properties of DL-roles in the various $\mathcal{DLR}$ flavours.

References

[Artale *et al.* (2002)] Artale, A., Franconi, E., Wolter, F., Zakharyaschev, M. A temporal description logic for reasoning about conceptual schemas and queries. In: *Proceedings of the 8th Joint European Conference on Logics in Artificial Intelligence (JELIA-02)*, S. Flesca, S. Greco, N. Leone, G. Ianni (eds.), Springer Verlag, 2002, LNAI 2424, pp98-110.

[Artale *et al.* (2003)] Artale, A., Franconi, E., Mandreoli, F. Description logics for modelling dynamic information. In Chomicki, J., van der Meyden, R., Saake, G. (Eds), *Logics for Emerging Applications of Databases*. Lecture Notes in Computer Science, Springer-Verlag, Berlin. 2003.

[Baader and Nutt (2003)] Baader, F., Nutt, W. Basic Description Logics. In: *Description Logics Handbook*, Baader, F. Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds). Cambridge University Press, 2003. pp47-100.

[Baader *et al.* (2003)] Baader, F. Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds). *Description Logics Handbook*, Cambridge University Press, 2003.

[Balsters *et al.* (2006)] Balsters, H., Carver, A., Halpin, T., Morgan, T. Modeling dynamic rules in ORM. 2nd International Workshop on Object-Role Modelling (ORM 2006), Montpellier, France, Nov 2-3, 2006. In: OTM Workshops 2006. Meersman, R., Tari, Z., Herrero., P. et al. (Eds.), Lecture Notes in Computer Science 4278. Berlin: Springer-Verlag, 2006. pp1201-1210.

[Berardi *et al.* (2005)] Berardi, D., Calvanese, D., De Giacomo, G. Reasoning on UML class diagrams. *Artificial Intelligence*, 2005, 168(1-2):70-118.

[Bollen (2006)] Bollen, P. Using fact-orientation for instructional design. 2nd International Workshop on Object-Role Modelling (ORM 2006), Montpellier, France, Nov 2-3, 2006. In: OTM Workshops 2006. Meersman, R., Tari, Z., Herrero., P. et al. (Eds.), Lecture Notes in Computer Science 4278. Berlin: Springer-Verlag, 2006. pp1231-1241.

[Evans (2005)] Evans, K. Requriemetns engineering with ORM. International Workshop on Object-Role Modeling (ORM'05). Cyprus, 3-4 November 2005. In: OTM Workshops 2005. Halpin, T., Meersman, R. (eds.), Lecture Notes in Computer Science 3762. Berlin: Springer-Verlag, 2005. pp646-655.

[Calvanese *et al.* (2006)] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati. Linking data to ontologies: The description logic DL-Lite A. In *Proc. of the 2nd Workshop on OWL: Experiences and Directions (OWLED 2006)*, 2006.

[Calvanese *et al.* (2005)] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R. DL-Lite: Tractable description logics for ontologies. In: *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pp602-607.

[Calvanese and De Giacomo (2003)] Calvanese, D., De Giacomo, G. Expressive description logics. In: *The Description Logic Handbook: Theory, Implementation and Applications*, Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (Eds). Cambridge University Press, 2003. pp178-218.

[Calvanese *et al.* (1999)] Calvanese, D., De Giacomo, G., Lenzerini, M. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In: *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pp84-89, 1999.

[Calvanese *et al.* (1998)] Calvanese, C., De Giacomo, G., Lenzerini, M. On the decidability of query containment under constraints. In: *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pp149-158, 1998.

[Calvanese *et al.* (1998)] Calvanese, D., Lenzerini, M., Nardi, D. (1998) Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*. Kluwer, Amsterdam.

[Campbell *et al.* (1996)] Campbell, L.J., Halpin, T.A. and Proper, H.A. Conceptual Schemas with Abstractions: Making flat conceptual schemas more comprehensible. *Data & Knowledge Engineering*, 1996, 20(1): 39-85.

[Franconi and Ng (2000)] Franconi, F., Ng, G. The ICOM Tool for Intelligent Conceptual Modelling. *7th Intl. Workshop on Knowledge Representation meets Databases (KRDB'00)*, Berlin, Germany, August 2000.

[Halpin (1989)] Halpin, T.A. *A logical analysis of information systems: static aspects of the data-oriented perspective*. PhD Thesis, University of Queensland, Australia. 1989.

[Halpin (2001)] Halpin, T. *Information Modeling and Relational Databases*. San Francisco: Morgan Kaufmann Publishers, 2001.

[Halpin (2003)] Halpin, T. Uniqueness constraints on objectified associations. *Journal of Conceptual Modeling*, October 2003. http://www.inconcept.com/jcm.

[Halpin (2005a)] Halpin, T. Objectification of relationships. *Proceedings of the 10th International IFIP WG8.1 Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD'05).* Porto, Portugal, 13-14 June, 2005.

[Halpin (2005b)] Halpin, T. ORM2. International Workshop on Object-Role Modeling (ORM'05). Cyprus, 3-4 November 2005. In: OTM Workshops 2005. Halpin, T., Meersman, R. (eds.), Lecture Notes in Computer Science 3762. Berlin: Springer-Verlag, 2005. pp676-687

[Hofstede *et al.* (1993)] Hofstede, A.H.M. ter, Proper, H.A., Weide, Th.P. van der. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 1993, 18(7):489-523.

[Hofstede and Proper, (1998)] Hofstede, A.H.M. ter, Proper, H.A.. How to Formalize It? Formalization Principles for Information Systems Development Methods. *Information and Software Technology*, 1998, (40(10): 519-540.

[Horrocks *et al.* (2006)] Horrocks, I., Kutz, O., Sattler, U. The Even More Irresistible $\mathcal{SROIQ}$. In: *Proceedings of the 10th International Conference of Knowledge Representation and Reasoning (KR-2006)*, Lake District, UK, 2006.

[Hoppenbrouwers *et al.* (2005)] Hoppenbrouwers, S.J.B.A., Proper, H.A., van der Weide, Th.P. Fact calculus: using ORM and Lisa-D to reason about domains. International Workshop on Object-Role Modeling (ORM'05). Cyprus, 3-4 November 2005. In: OTM Workshops 2005. Halpin, T., Meersman, R. (eds.), Lecture Notes in Computer Science 3762. Berlin: Springer-Verlag, 2005. pp720-729.

[Juristo and Moreno (2000)] Juristo, N., Moreno, A.M. introductory paper: reflections on conceptual modelling. *Data & Knowledge Engineering*, 2000, 33(2):103-117.

[Pepels and Plasmeijer (2005)] Pepels, B., Plasmeijer, R. Generating applications from Object Role Models. International Workshop on Object-Role Modeling (ORM'05). Cyprus, 3-4 November 2005. In: OTM Workshops 2005. Halpin, T., Meersman, R. (eds.), Lecture Notes in Computer Science 3762. Berlin: Springer-Verlag, 2005. pp656-665.

[de Troyer *et al.* (2005)] Troyer, O. de, Casteleyn, S., Plessers, P. Using ORM to model web systems. International Workshop on Object-Role Modeling (ORM'05). Cyprus, 3-4 November 2005. In: OTM Workshops 2005. Halpin, T., Meersman, R. (eds.), Lecture Notes in Computer Science 3762. Berlin: Springer-Verlag, 2005. pp700-709.

[1] iCOM: intelligent Conceptual modelling tool. http://www.inf.unibz.it/~franconi/icom.

[2] NORMA: Neumont ORM Architect. http://sourceforge.net/projects/orm/

[3] Object-Role Modeling. http://www.orm.net.

[4] OWL Web Ontology Language 1.1 (Editor's draft of 27-11-2006). http://owl1_1.cs.manchester.ac.uk/owl_specification.html.

[5] OWL Web Ontology Language. http://www.w3.org/TR/2004/REC-owl-semantics-20040210/syntax.html.

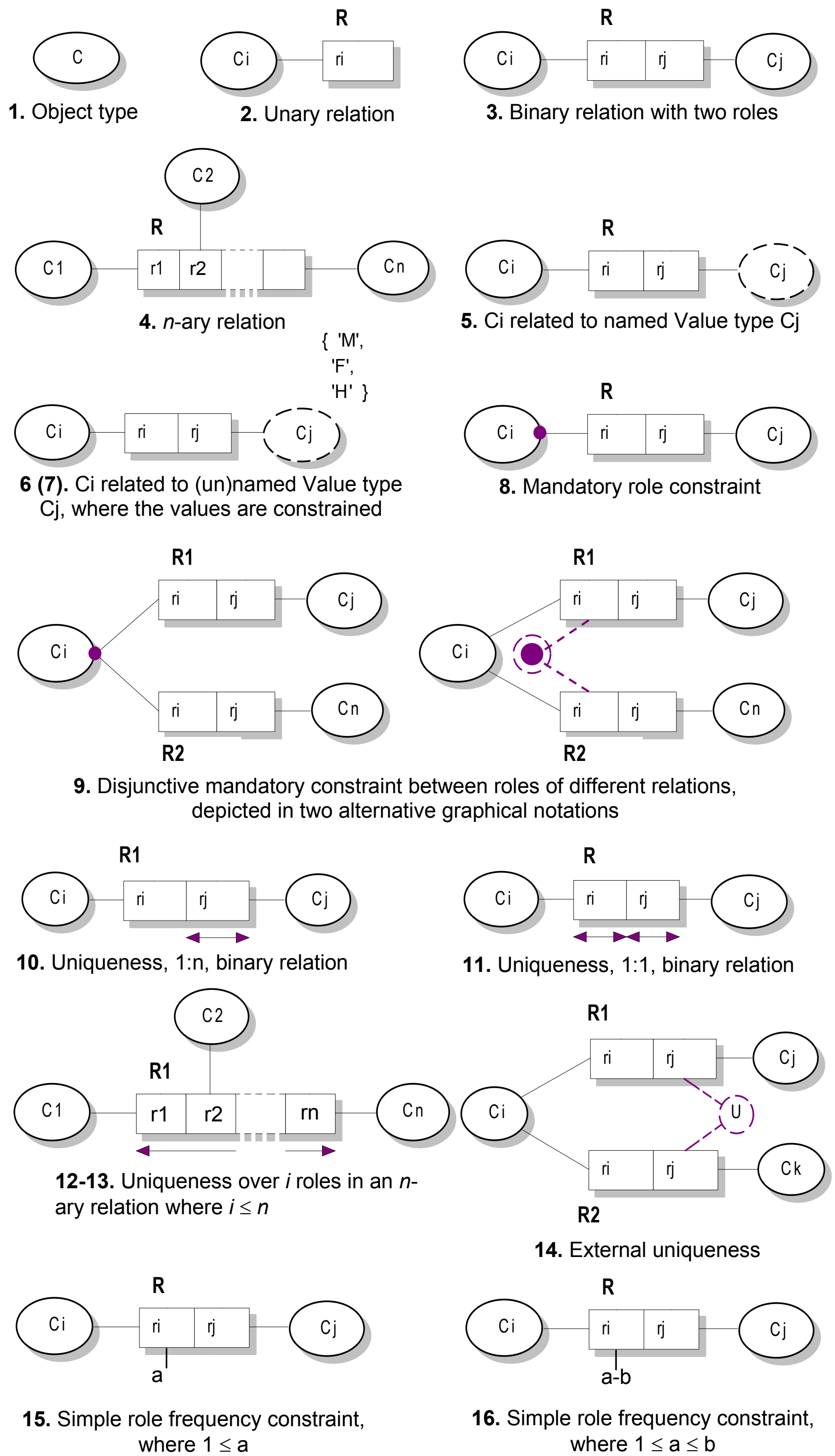

Figure 2: Diagrammatic representation of ORM object types, value types, roles and several constraints.

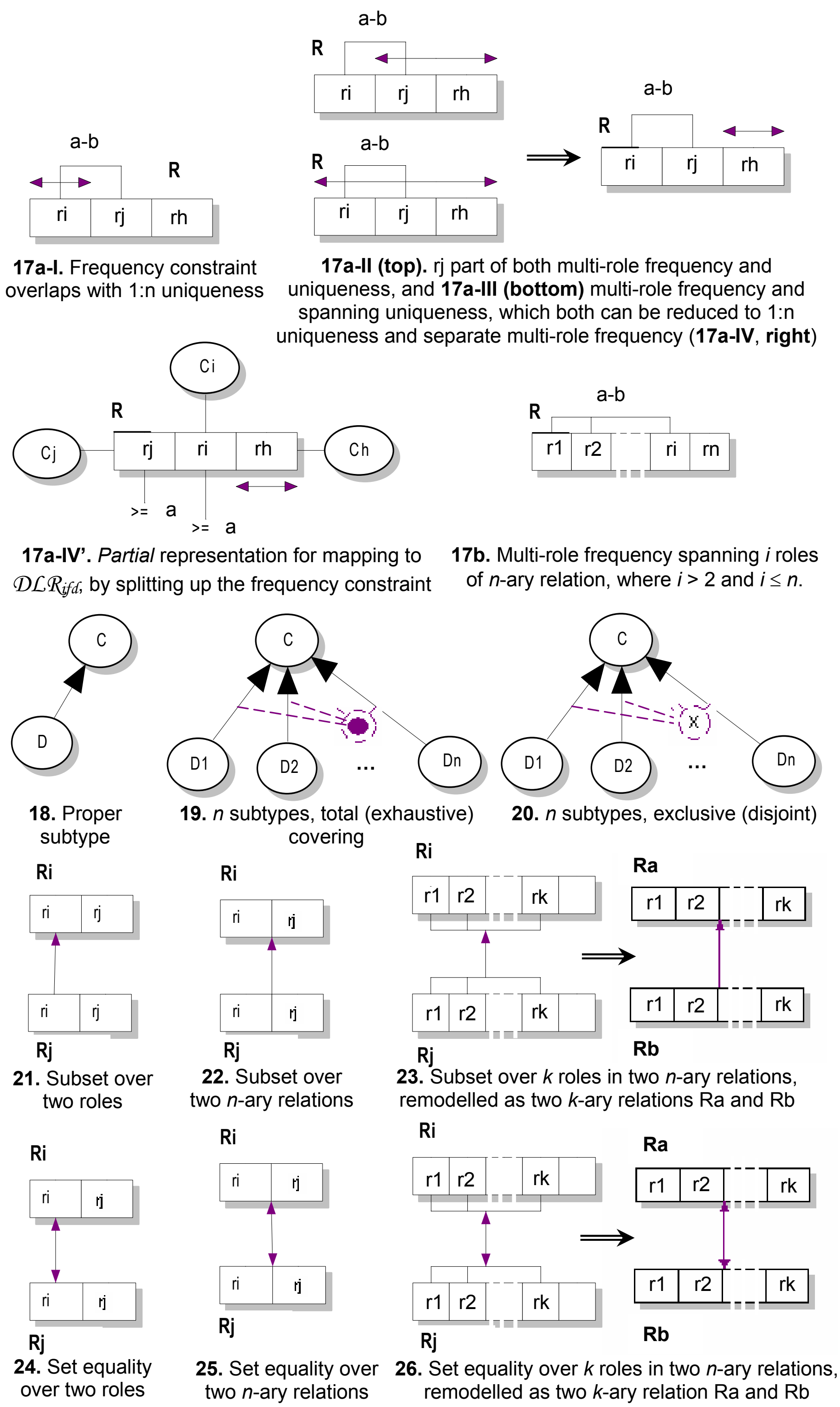

Figure 3: Diagrammatic representation of several ORM constraints.

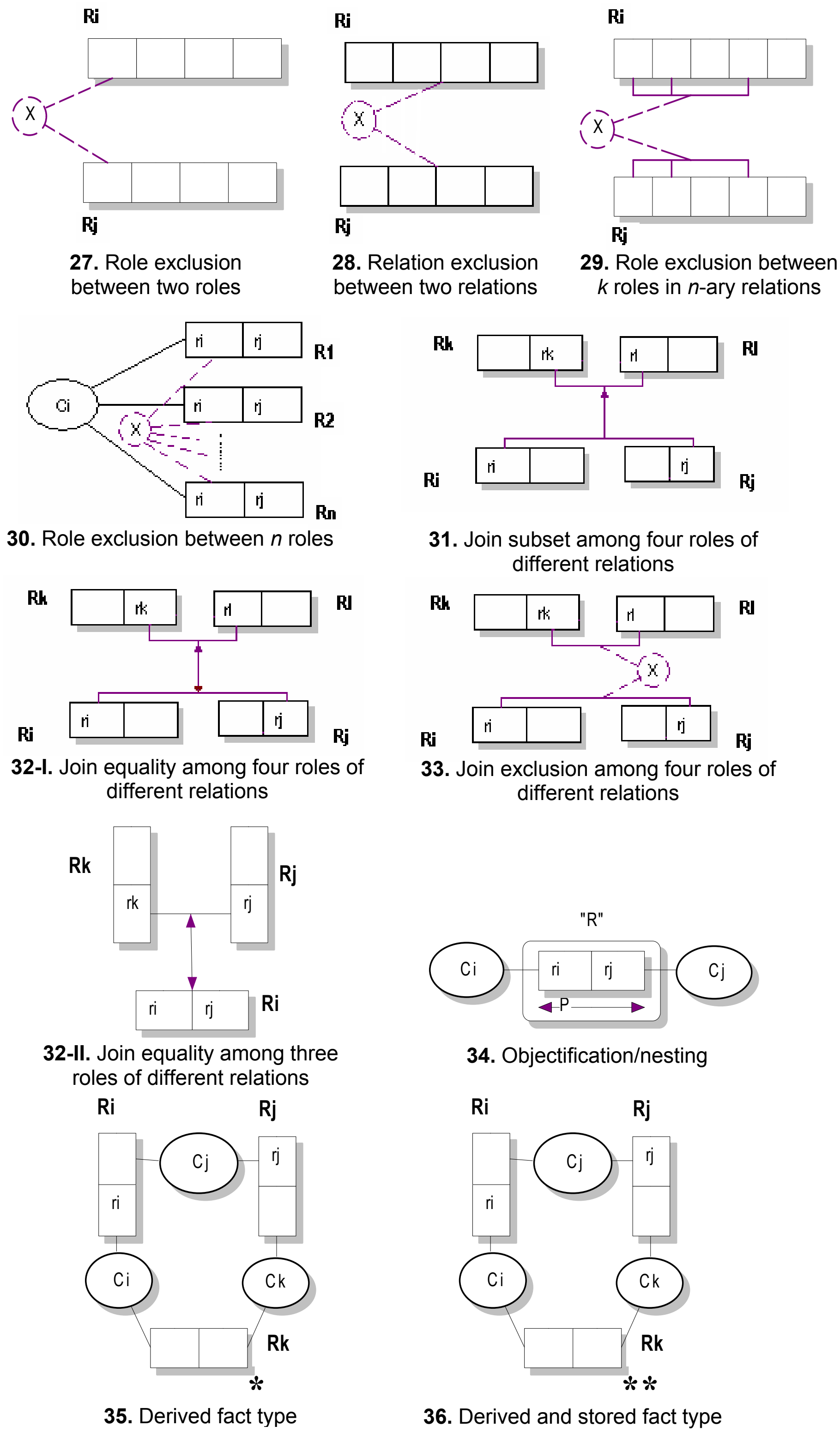

Figure 4: Diagrammatic representation of more ORM constraints.

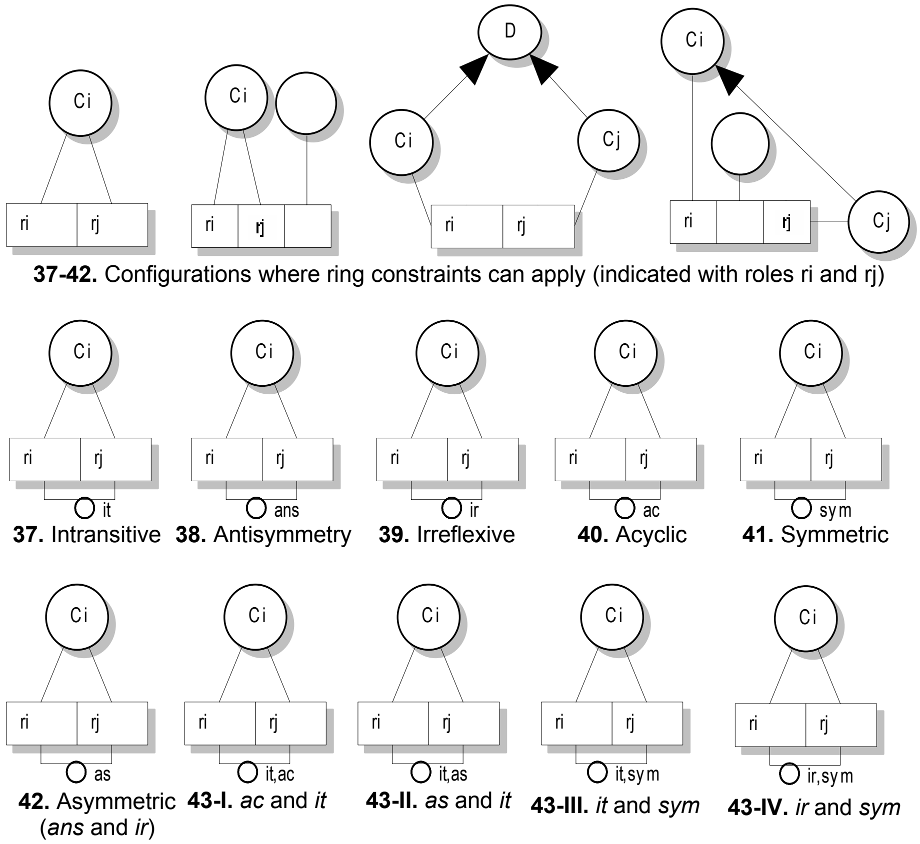

Figure 5: Diagrammatic representation of ORM ring constraints.

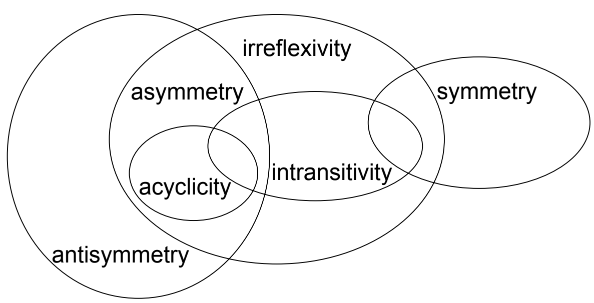

Figure 6: Relations between the possible ring constraints (after [Halpin (2001)]).