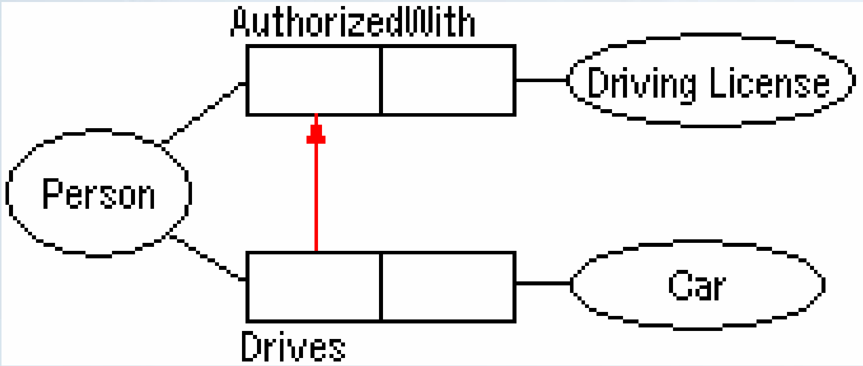
DBDBD (Dutch-Belgian Database Day) 2006
Brussels
15 November 2006

Reasoning on ORM Schemes

Mustafa Jarrar
mjarrar@vub.ac.be
STARLab, Vrije Universiteit Brussel,
Belgium

What is ORM

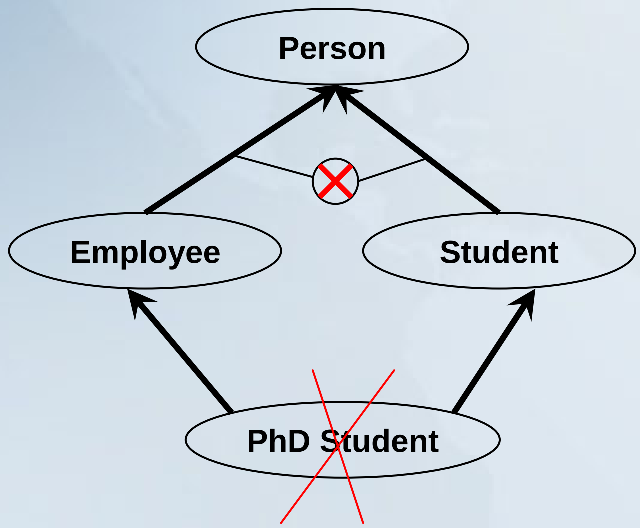
- ORM (Object-Role Modeling) is a rich conceptual modeling method
- Successor of NIAM (early 70s).
- Originally developed as a database modeling approach, but its being reused now for *ontology modeling*, business rule modeling, XML-Schema conceptual design, web form design, data warehouse, etc.



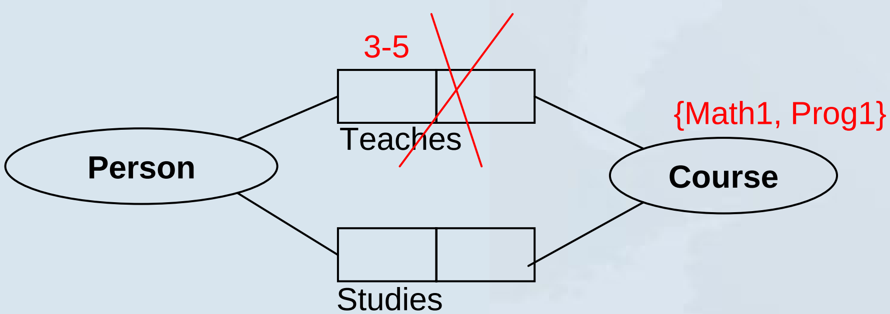
ORM supports over than 20 constraint types (identity, mandatory, uniqueness, subsumption, subset, equality, exclusion, value, frequency, symmetric, intransitive, acyclic, etc.).

Goals

How to detect (contradiction, implications, etc.) in an ORM schema?



$$Employee \sqsubseteq Person$$
$$Student \sqsubseteq Person$$
$$Student \sqcap Employee \equiv \bot$$
$$PhDStudent \sqsubseteq Student$$
$$PhDStudent \sqsubseteq Employee$$



$$R1 \sqsubseteq (Teaches : Person) \sqcap (r2 : Course)$$
$$R2 \sqsubseteq (Studies : Person) \sqcap (r2 : Course)$$
$$Course \equiv \{Math1, Prog1\}$$
$$Person \sqsubseteq \exists^{\geq 3, \leq 5}[Teaches].R1$$

Goals

How to detect (contradiction, implications, etc.) in an ORM schema?

We propose two approaches:

➊ Pattern-based approach: (9 patterns of constraint contradictions)

- Very fast detection of unsatisfiability,
- Detection message are easy to understand by a nonintellectual,
- Cheap to implement.
- Incomplete reasoning.

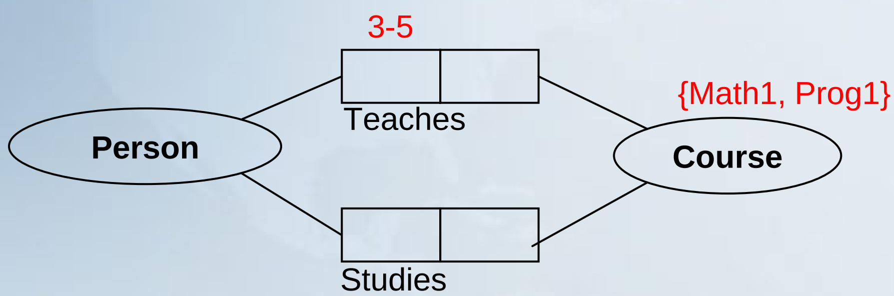
➋ Description logic based approach:

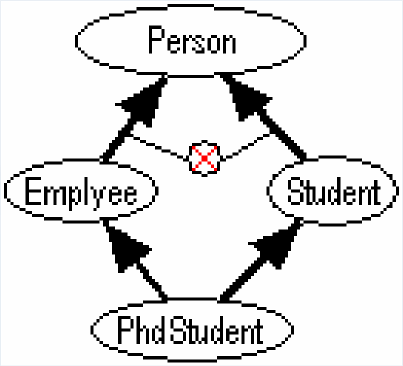
(Formalize ORM in description logic and reason automatically using Racer)

- Not all constraints can be implemented (yet) by Racer.
- Detection message are not easy to understand by a nonintellectual.
- Complete Reasoning.

•→ Both approaches are implemented in DogmaModeler (Demo)

Types of Satisfiability





- Schema satisfiability: A schema is satisfiable if and only if there is at least one concept in the schema that can be populated. ➔ Weak satisfiability
- Concept satisfiability: A schema is satisfiable if and only if all concepts in the schema can be populated.
- Role satisfiability: A schema is satisfiable if and only if all roles in the schema can be populated. ➔ Strong satisfiability

➢ Concept satisfiability implies schema satiability.

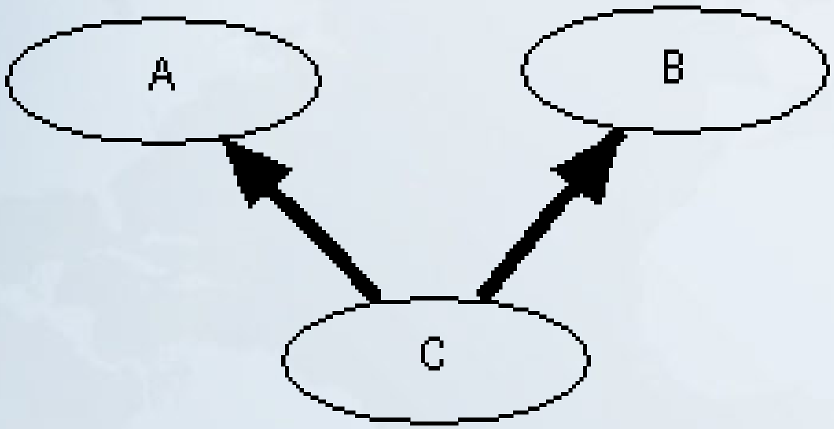
➢ Role satisfiability implies concept satiability.

➊ The patterns-based approach

- **Pattern 1** (Top common supertype)
- **Pattern 2** (Exclusive constraint between types)
- **Pattern 3** (Exclusion-Mandatory)
- **Pattern 4** (Frequency-Value)
- **Pattern 5** (Value-Exclusion-Frequency)
- **Pattern 6** (Set-comparison constraints)
- **Pattern 7** (Uniqueness-Frequency)
- **Pattern 8** (Ring constraints)
- **Pattern 9** (Loops in Subtypes)

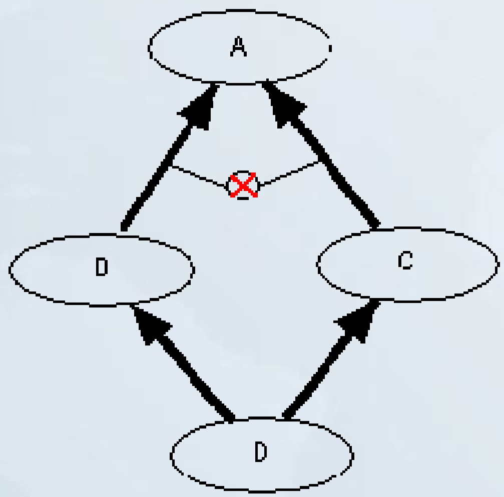
Jarrar, M., Heymans, S.: Unsatisfiability Reasoning in ORM Conceptual Schemes. In Illarramendi, A., Srivastava, D.: Proceeeding of International Conference on Semantics of a Networked World. Springer, LNCS, pp.:-. Munich, Grmany, March 2005.

Pattern 1 (Top common supertype)



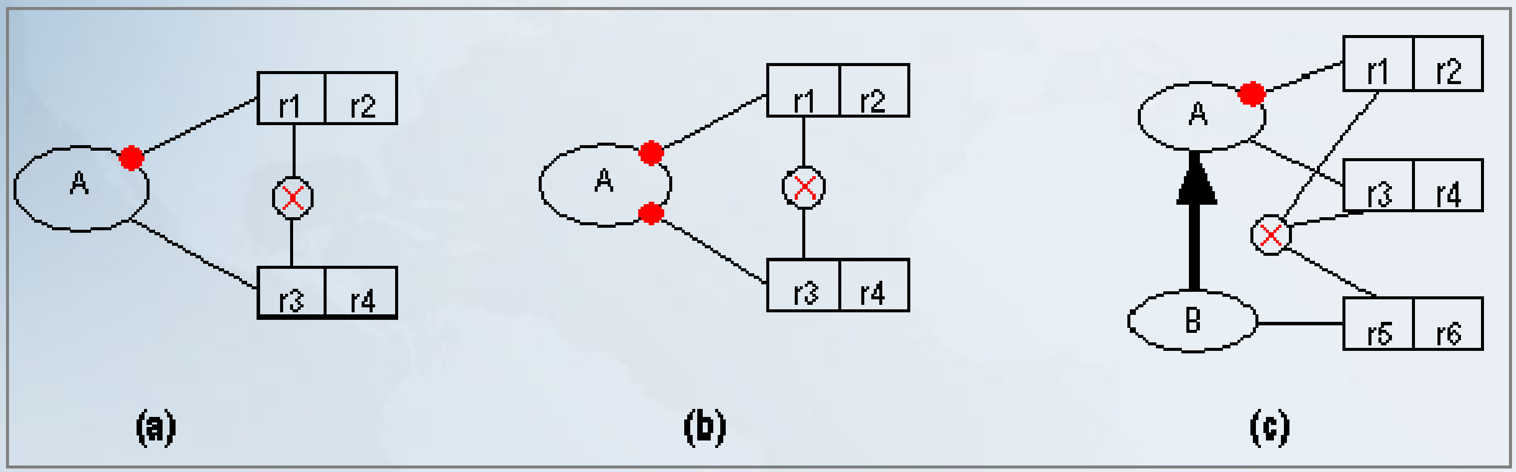
- All object types In ORM are mutually exclusive, except those that are subtypes.
- If a subtype has more than one supertype, these supertypes must share a top supertype; otherwise, the subtype cannot be satisfied.

Pattern 2 (Exclusive constraint between types)



For each exclusive constraint between a set of object types $T=\{T_1,..T_n\}$, let T_i.Subs be the set of all possible subtypes of the object type T_i, and Tj.subs be the set of all possible subtypes of the object type T_j, where $i \neq j$, the set T_i.Subs $\cap$ T_j.Subs) must be empty. Otherwise members in this set are not satisfiable; and hence, the composition is considered as **incompatible operation.**

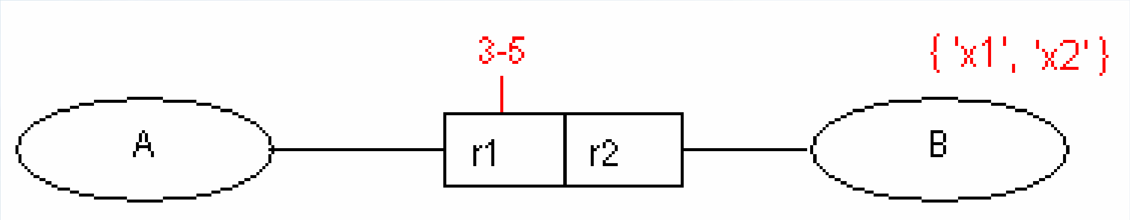
Pattern 3 (Exclusion-Mandatory)



A contradiction occurs if an object type plays a mandatory role that is exclusive with other roles played by this object type or one of its subtypes.

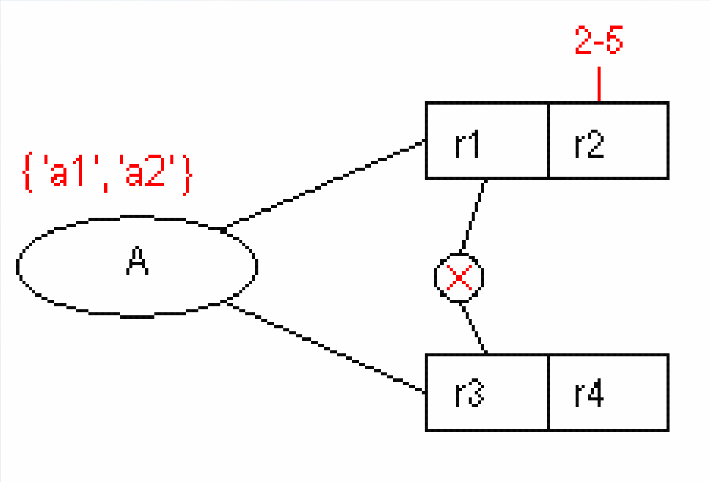
For each exclusion constraint between a set of single roles R, let $R_i.T$ be the object type that plays the role R_i, $R_i \in R$. For each (R_i, R_j), where i $\neq j$ and R_i *is mandatory*, If $R_i.T = R_j.T$ or $R_j.T \in R_i.T.Subs$ -where $R_i.O.Subs$ is the set of all subtypes of the object type $R_i.T$ - then some roles in R cannot be populated.

Pattern 4 (Frequency-Value)



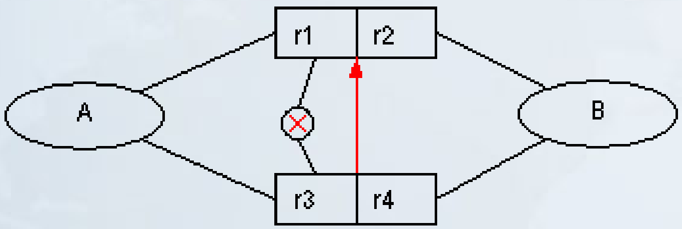
For each fact type (*A r B*), let c be the number of the possible values of *B* that can be calculated from its value constrain, and let (*n-m*) be a frequency constraint on the role *r*, *c* must be equal or more than *n*. Otherwise, the role *r* cannot be satisfied, as the value and the frequency constraints contradict each other.

Pattern 5 (Value-Exclusive-Frequency)



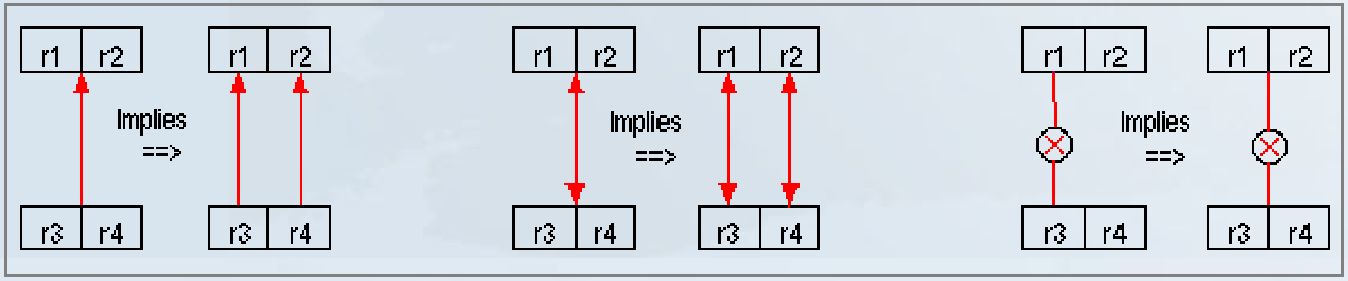
Formally, for each exclusion constraint, let be the set of roles participating in this constraint. With each of those roles R_i, we associate the inverse role S_i, and we let f_i be the minimum of the frequency constraint on S_i (if there is no frequency constraint on S_i, we take f_i equal to 1). Let be the object-type that plays all roles in *R*. Let be the number of the possible values of *T*, according to the value constraint. *C* must always be more than or equal to $f_1 + \ldots + f_n$. Otherwise, some roles in *R* cannot be satisfied.

Pattern 6 (Set-comparison constraints)

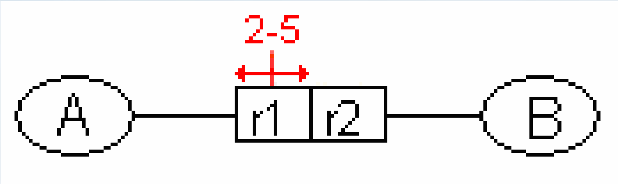


For each exclusion constraint between A and B: If A and B are two predicates, there should not be any (direct or implied) SetBath between these predicates; If A and B are single roles, there should not be any (direct or implied) SetBath between both roles or between the predicates that include these roles.

Main set-comparison implications:



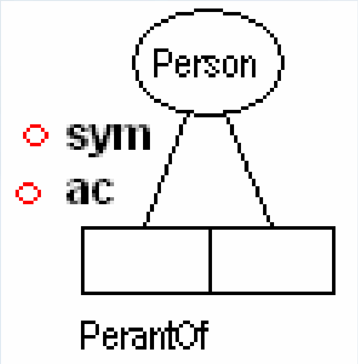
Pattern 7 (Uniqueness-Frequency)



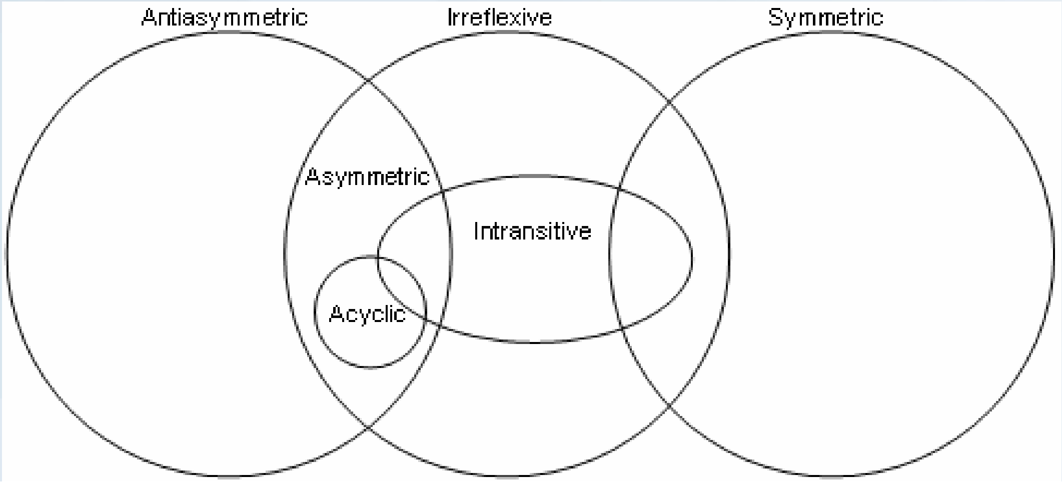
unsatisfiability of a role occurs if there is a frequency constraint FC(min-max) and a uniqueness constraint on some role (or predicate) r where min is strictly greater than 1.

Pattern 8 (Ring constraints)

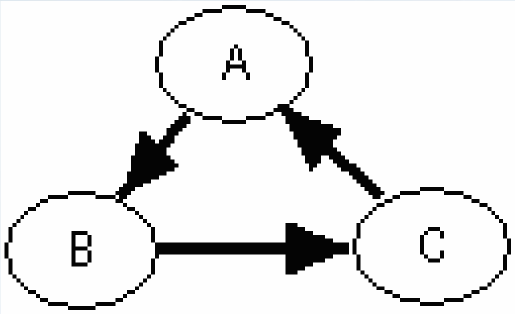
RM allows ring constraints: antisymmetric (ans), asymmetric (as), acyclic (ac), irreflexive (ir), intransitive (it), and symmetric (sym)



Any combination of ring constraints should have intersection in the following diagram:



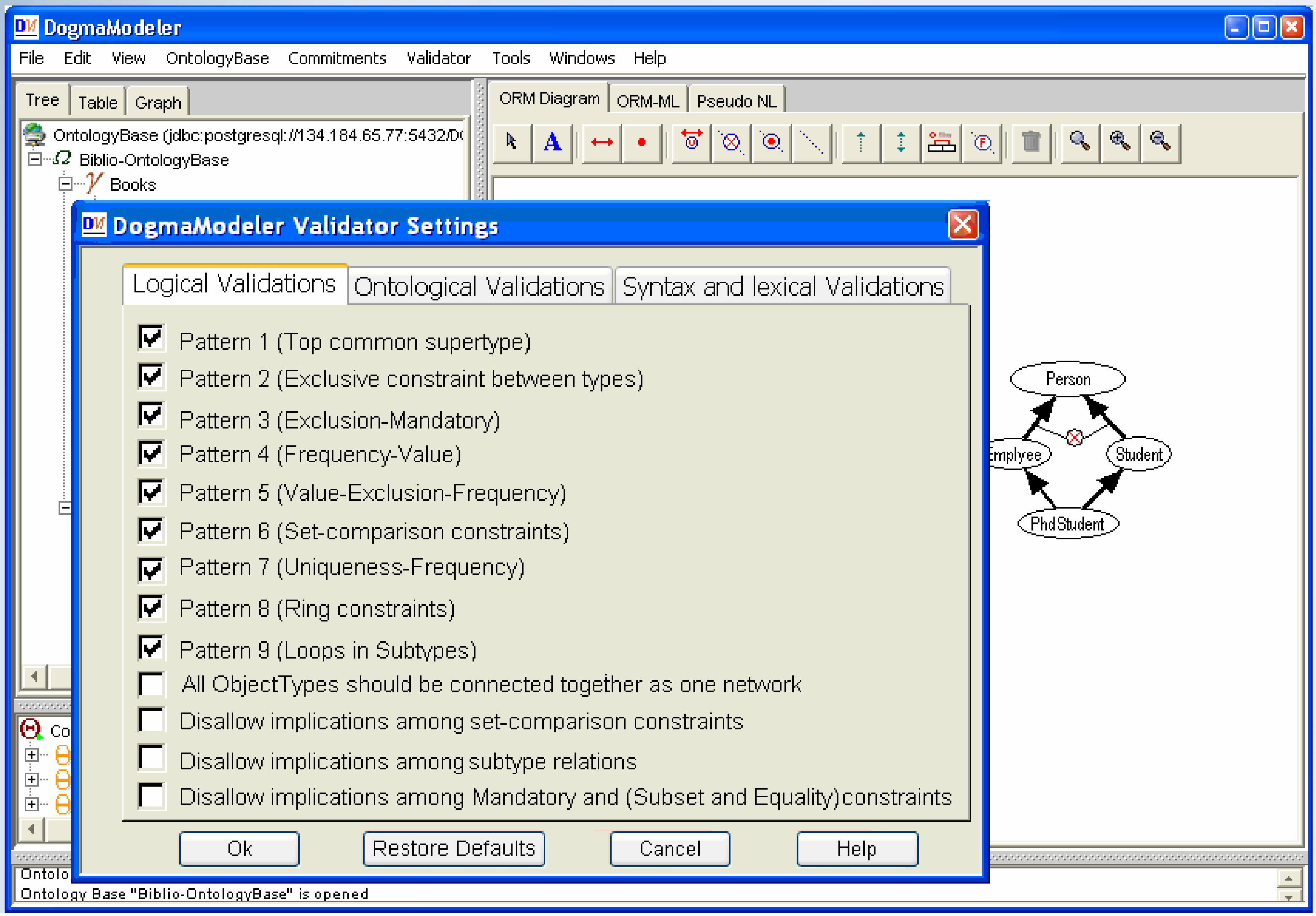
Pattern 9 (Loops in Subtypes)



Formally, for each subtype *T* in the schema, let *T*.Supers be the set of all supertypes of *T*. If *T* in *T*.Supers, then the object-type *T* cannot be satisfied.

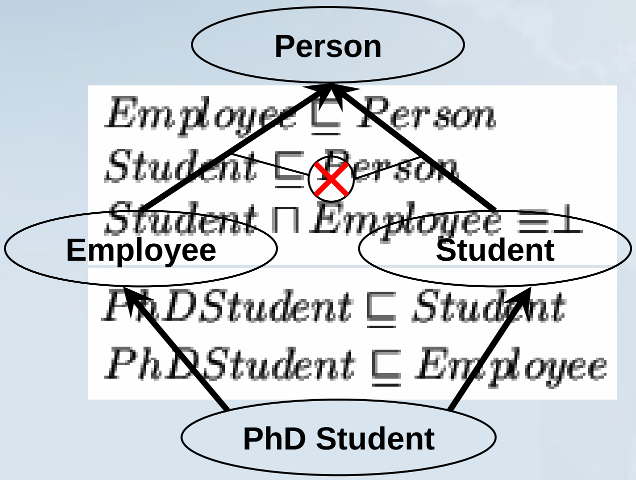
DogmaModeler

An ontology engineering tool and business rules (uses ORM). See [J05].

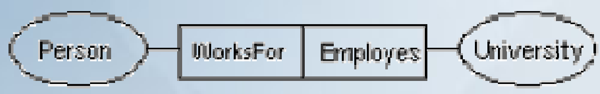


❷ Description logic based approach

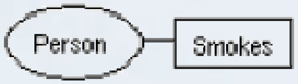
Map ORM into description logic [JF06], and based on this, use Racer to reason about ORM schema [JD06].



[JF06]: Jarrar, M., Franconi, E.: Mapping ORM into the DLR description logic. (submitted), September 2006.
[JD06]: Jarrar, M., Damag, M.: reasoning on ORM using Racer. (Submitted), August 2006.

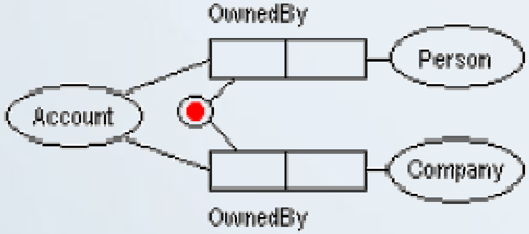


$R \sqsubseteq (WorksFor : Person) \sqcap (Employes : University)$

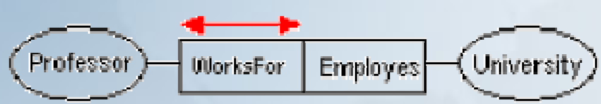


$R \sqsubseteq (Smokes : Person) \sqcap (r2 : BOOLEAN)$

$R \sqsubseteq (WorksFor : Professor) \sqcap (r2 : University)$
$Professor \sqsubseteq \exists[WorksFor].R$

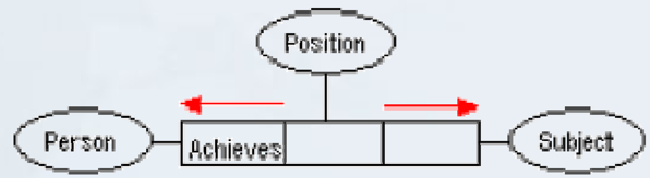


$R1 \sqsubseteq (OwnedBy : Account) \sqcap (r2 : Person)$
$R2 \sqsubseteq (OwnedBy : Account) \sqcap (r2 : Company)$
$Account \sqsubseteq \exists[OwnedBy].R1 \sqcup \exists[OwnedBy].R2$



$R \sqsubseteq (WorksFor : Professor) \sqcap (Employes : University)$

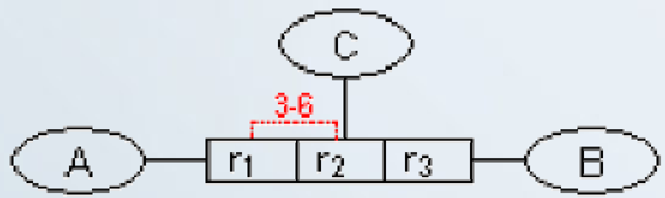
$Professor \sqsubseteq \exists^{\leq 1}[WorksFor].R$

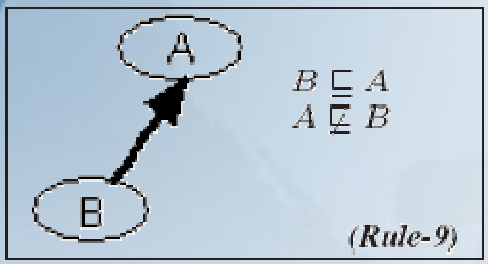


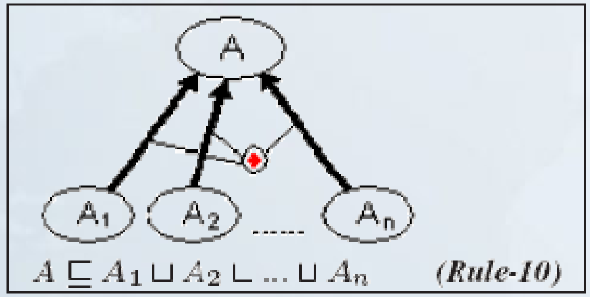
$R \sqsubseteq (Achieves : Person) \sqcap (r2 : Position) \sqcap (r3 : Subject)$

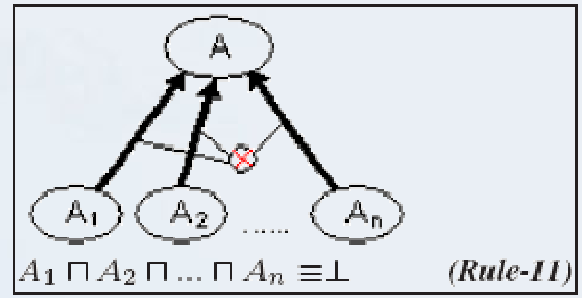
fd $R\ Achieves, r_2 \rightarrow r_3$

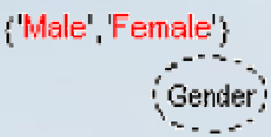
$Car \sqsubseteq \exists^{\geq 3, \leq 4}[HasPart].R \sqcup \bot$











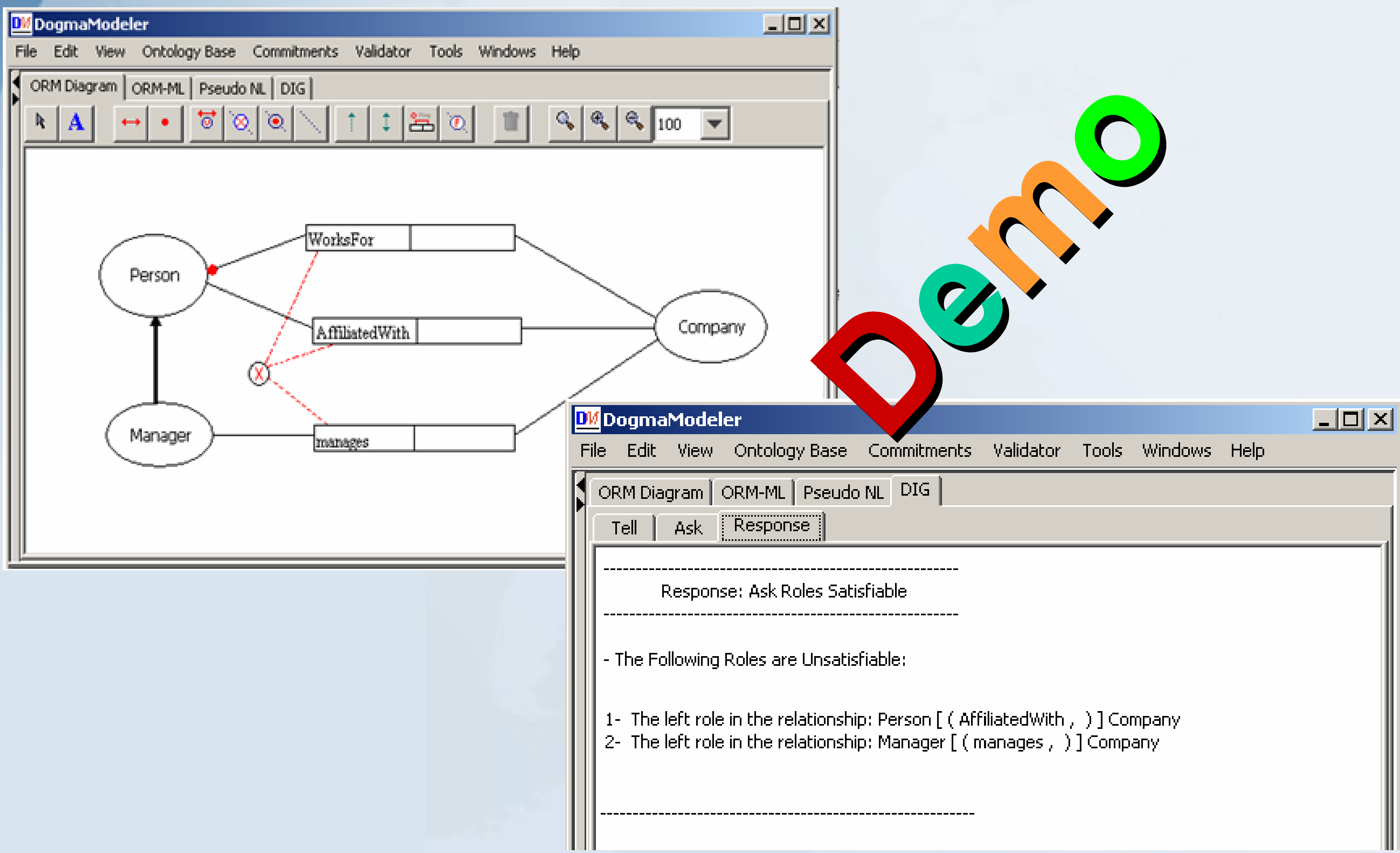
$$Gender \sqsubseteq STRING$$
$$Gender \equiv \{Male, Female\}$$

$$Gender \sqsubseteq NUMBER$$
$$Gender \equiv \{1, 2, 3\}$$

Etc…..

DogmaModeler

An ontology engineering tool and business rules (uses ORM). See [J05].



[J05]: Jarrar M.: Towards methodological principles for ontology engineering . PhD Thesis. Vrije Universiteit Brussel. (May 2005)

Discussion and Conclusions

Comparison: pattern detection approaches and a complete reasoning procedure in description logic, the pattern approach:

- Detection messages are easy to understand by a nonintellectual,
- Suitable for in interactive modeling,
- Easy to implement,

Experience from the CCFORM: (A Customer Complaint Ontology, built by (about) 50 lawyers.)

- Detecting unsatisfiability in an interactive manner helps ontology builders in quick detection of mistakes.
- Interactive detection of unsatisfiability improves the modeling skills of ontology builders, especially those who are not well trained in ontology modeling and logics.
- Although these patterns might be not complete, but they cover a lot of the ground.

Thank You