Representing ORM in Description Logic

Mustafa Jarrar

Birzeit University
mjarrar@birzeit.edu
www.jarrar.info

This lecture is based on

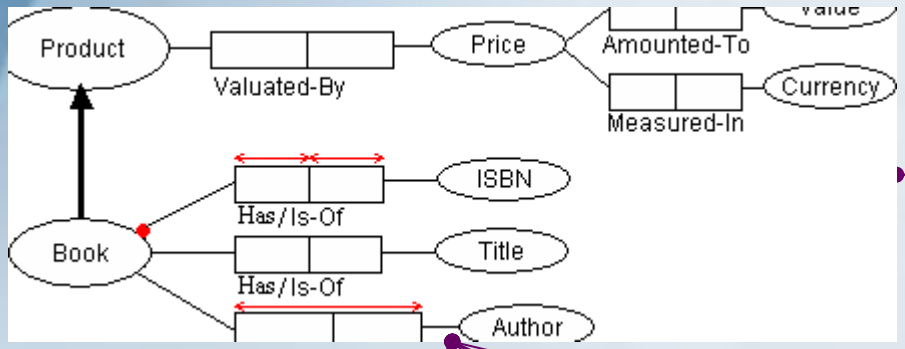
1. Mustafa Jarrar: Mapping ORM Into The SHOIN/OWL Description Logic- Towards A Methodological And Expressive Graphical Notation For Ontology Engineering . In OTM 2007 workshops: Proceedings of the International Workshop on Object-Role Modeling (ORM'07). Pages (729-741), LNCS 4805, Springer. ISBN: 9783540768890. Portogal. November, 2007

1. Mustafa Jarrar: Towards Automated Reasoning On ORM Schemes -Mapping ORM Into The DLR_idf Description Logic. In proceedings of the 26th International Conference on Conceptual Modeling (ER 2007). Pages (181-197). LNCS 4801, Springer. Auckland, New Zealand. ISBN 9783540755623. November 2007

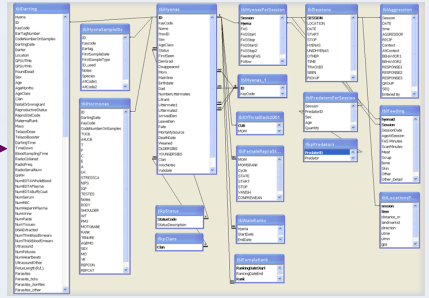
Outline

- Part I: Introduction (Why Description logics)
- Part II: Introduction to Description logics
- Part III: Mapping ORM into Description logic
- Something to think about ☺

Remember: ORM has many Applications



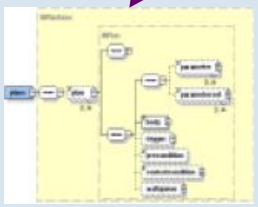
Originally



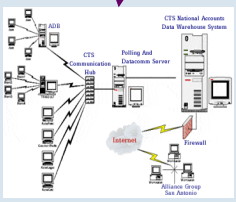
Database

Later

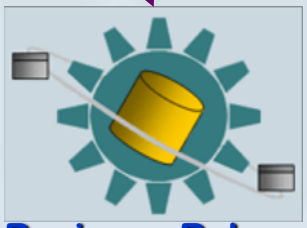
Ontology



XML Schema



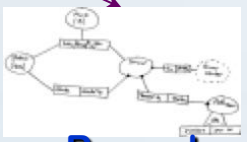
Warehouse



Business Rules

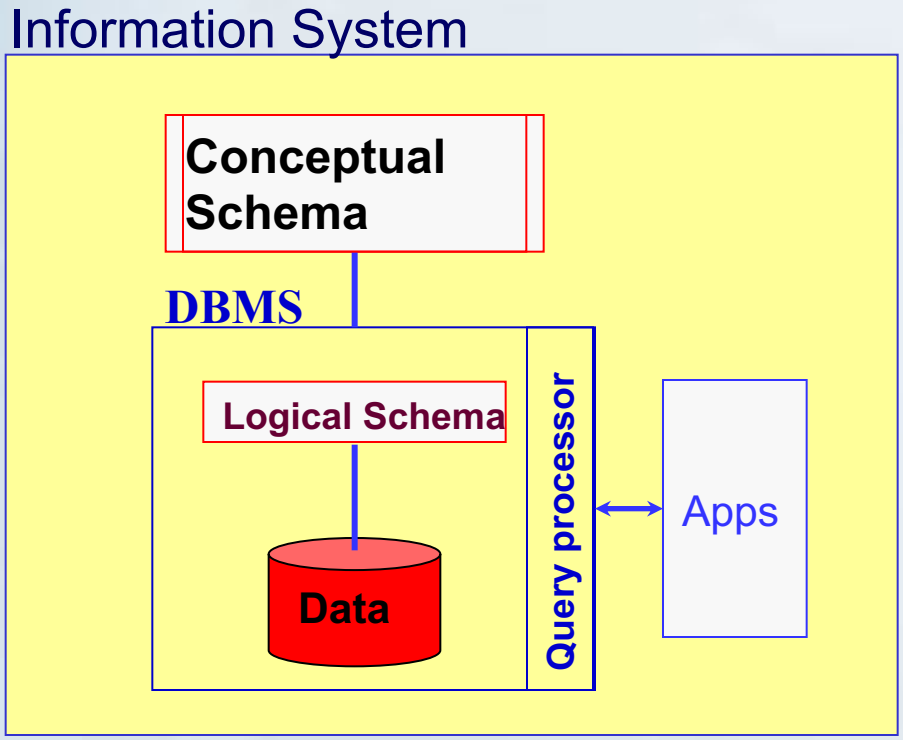
Web (x)Forms

Requirements Engineering



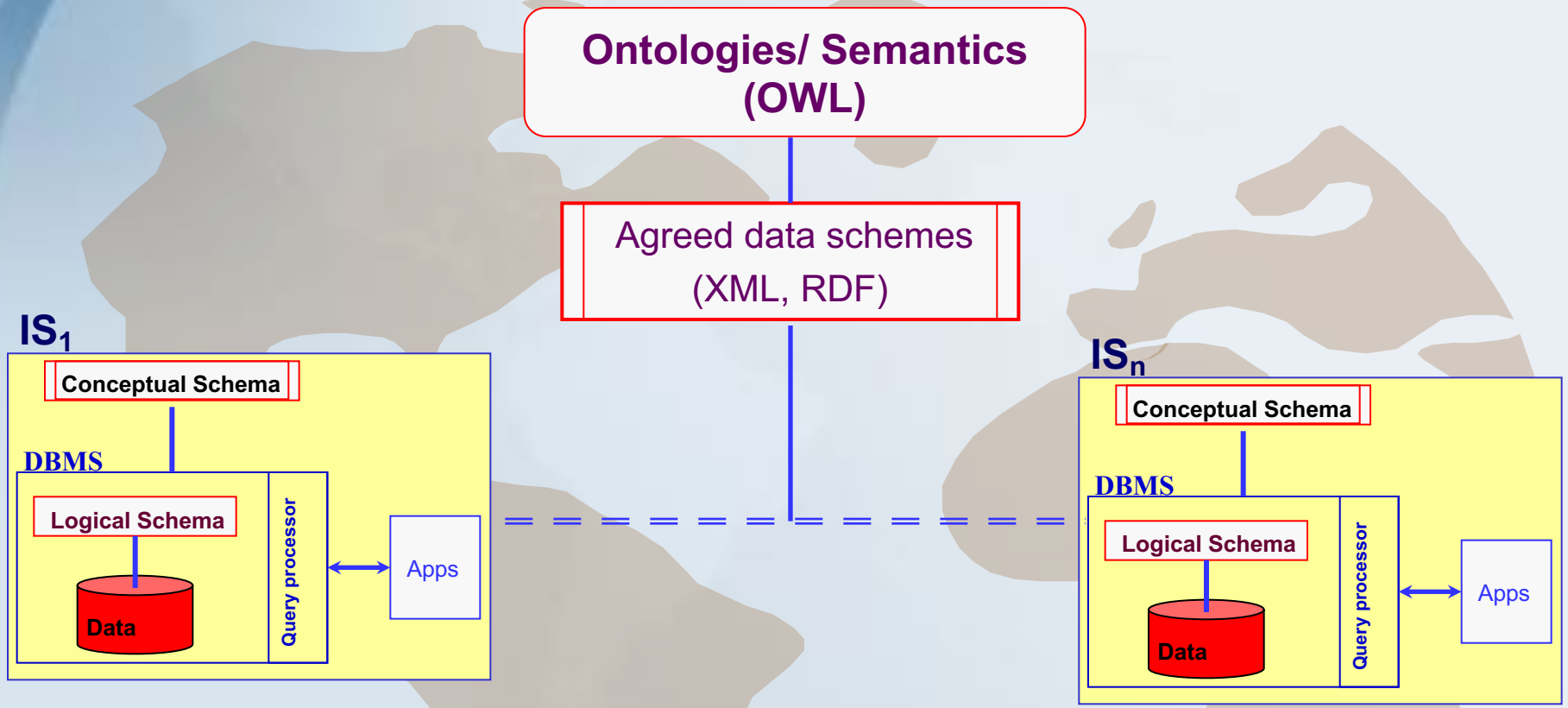
Record my recipes !

An Information System



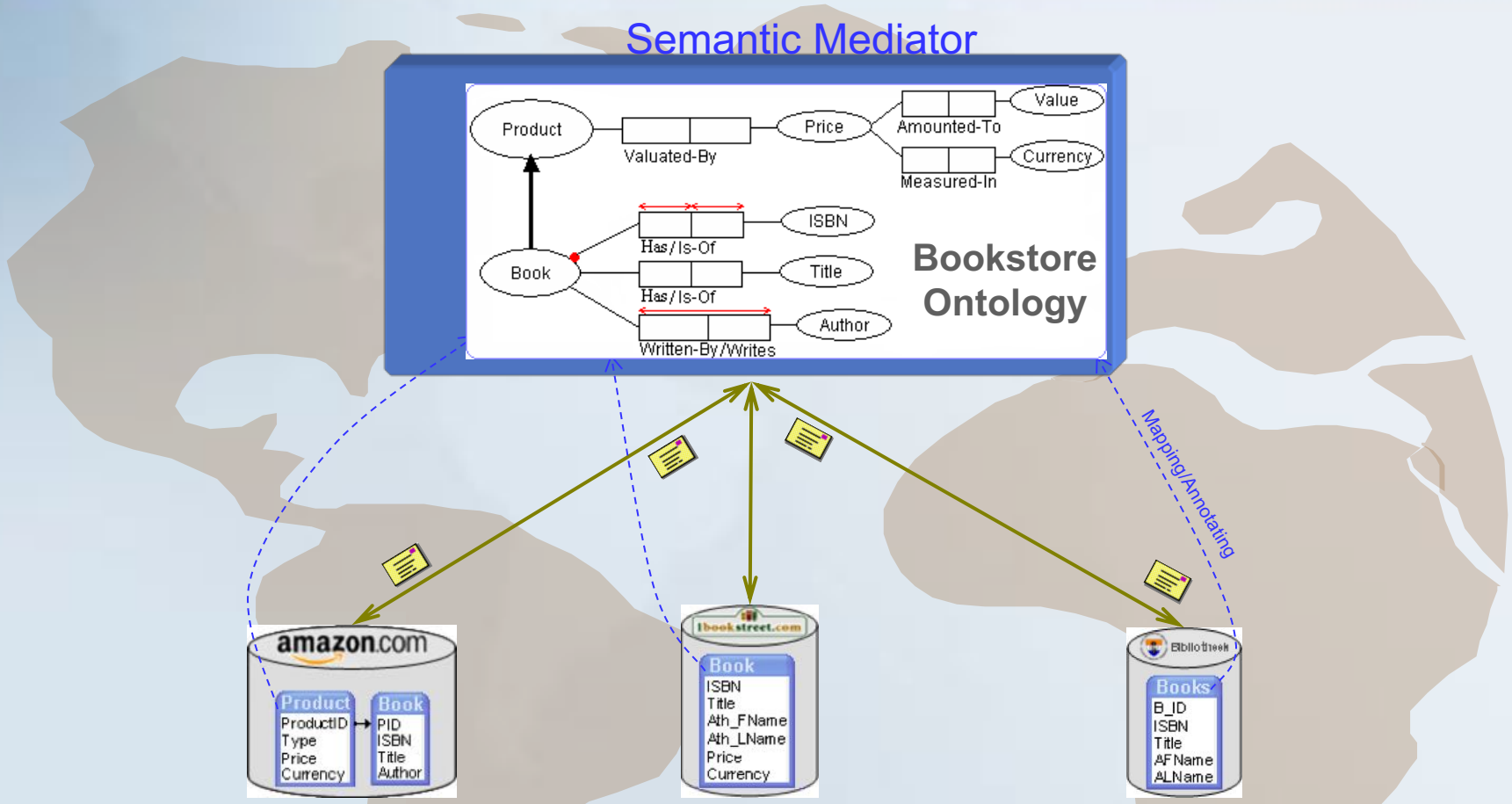
- Each Information System is made for one organization.
- Why do we need conceptual schemes? for designing Information systems at the conceptual level.

(Web) Information Systems

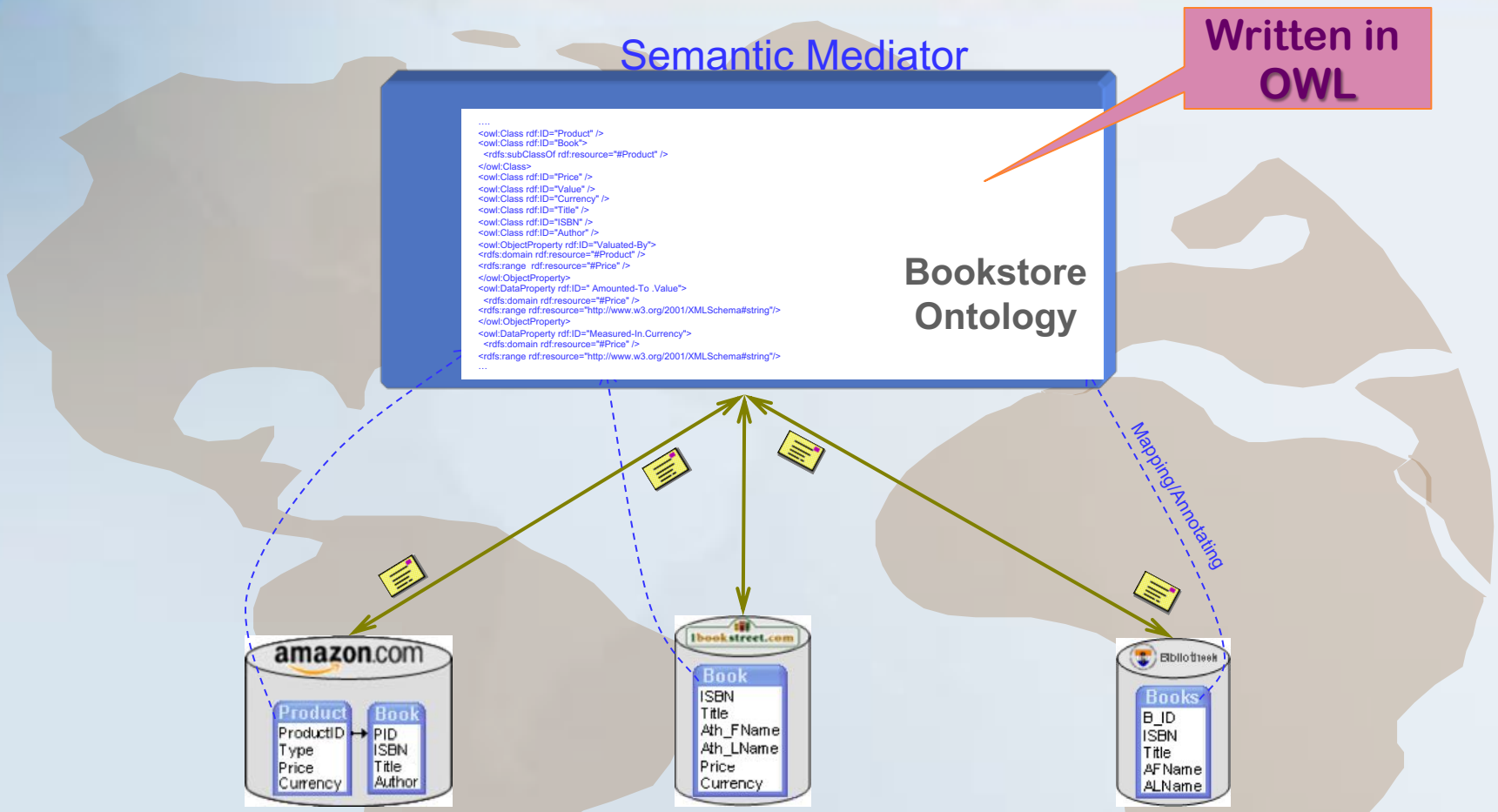


New needs:
Open data exchange, inter-organizational transactions, global queries…

(Web) Information Systems



(Web) Information Systems



OWL

- The Web Ontology Language (W3C Standard)
- Based on Description logic

```
….
<owl:Class rdf:ID="Product" />
<owl:Class rdf:ID="Book">
    <rdfs:subClassOf rdf:resource="#Product" />
</owl:Class>
<owl:Class rdf:ID="Price" />
<owl:Class rdf:ID="Value" />
<owl:Class rdf:ID="Currency" />
<owl:Class rdf:ID="Title" />
<owl:Class rdf:ID="ISBN" />
<owl:Class rdf:ID="Author" />
<owl:ObjectProperty rdf:ID="Valuated-By">
      <rdfs:domain rdf:resource="#Product" />
      <rdfs:range  rdf:resource="#Price" />
</owl:ObjectProperty>
<owl:DataProperty rdf:ID=" Amounted-To .Value">
      <rdfs:domain rdf:resource="#Price" />
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:ObjectProperty>
…
```

➔ Can we use ORM to Model OWL ontologies?

First Order Logic (FOL)

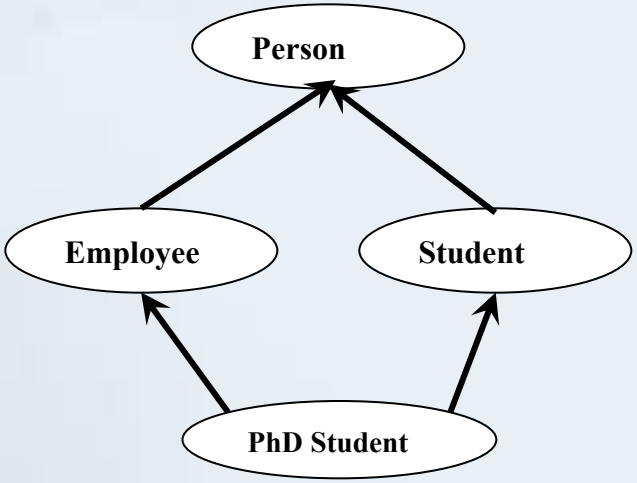
- FOL allows us to represent knowledge precisely (Syntax and Semantics).

  ∀x Employee(x) → Person (x)

  ∀x Student(x) → Person (x)

  ∀x PhDStudent(x) → Student (x)

  ∀x PhDStudent(x) → Employee (x)



- However, representation alone is not enough.

- We also need to process this knowledge and make use of it, i.e. Logical inference = (Reasoning).

First Order Logic (FOL)

Reasoning:

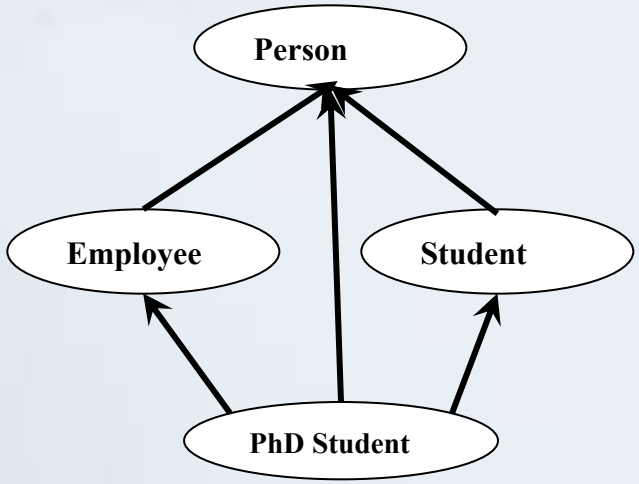
$\forall x$ Employee(x) → Person (x)

$\forall x$ Student(x) → Person (x)

$\forall x$ PhDStudent(x) → Student (x)

$\forall x$ PhDStudent(x) → Employee (x)

$\forall x$ PhDStudent(x) → Person (x)



➔How to process the above *axioms to know that an axiom can be derived from (=implied by) another axiom.*

First Order Logic (FOL)

Reasoning:

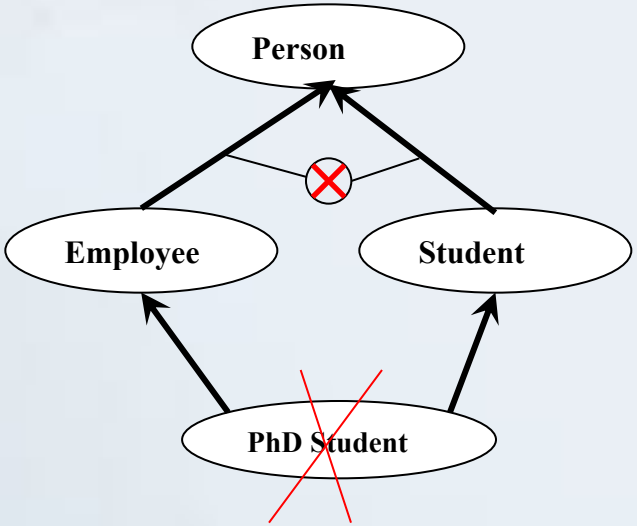
$\forall x$ Employee(x) → Person (x)

$\forall x$ Student(x) → Person (x)

$\forall x$ PhDStudent(x) → Student (x)

$\forall x$ PhDStudent(x) → Employee (x)

$\forall x$ Student(x) $\cap$ Employee (x) = $\varnothing$



➔How to process the above *axioms to know that an axiom can be derived from (=implied by) another axiom.*

➔Find contradictions (satisfiability)

➔…etc.

First Order Logic (FOL)

Reasoning on FOL is far too complex (i.e. not decidable)

→ Here comes description logics.

Introduction to

Description Logics

Description Logics

- Description logics are a **family of logics** concerned with knowledge representation.
- A description logic is a ***decidable*** fragment of first-order logic, associated with a set of automatic ***reasoning*** procedures.
- The basic constructs for a description logic are the notion of a **concept** and the notion of a **relationship**.
- **Complex concept** and relationship expressions can be constructed from atomic concepts and relationships with suitable constructs between them.
- Example:

$$Mother \sqsubseteq Female \sqcap \exists HasChild$$

$$HumanMother \sqsubseteq Female \sqcap \exists HasChild.Person$$

Description Logics

Most known description logics are :

$\mathcal{FL}^-$ The simplest and less expressive description logic.

$C, D \rightarrow A \mid C \sqcap D \mid \forall R.C \mid \exists R$

$\mathcal{AL}$ A more practical and expressive description logic.

$C, D \rightarrow A \mid \top \mid \bot \mid \neg A \mid C \sqcap D \mid \forall R.C \mid \exists R.\top$

$\mathcal{SROIQ}$ Very popular description logic.
The logic underlying OWL.

$\mathcal{DLR}_{idf}$ Very expressive description logic,
Capable of representing most database constructs.

Description logic $\mathcal{AL}$

Syntax	Semantics
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, A is a concept name
$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
$\bot$	$\bot^{\mathcal{I}} = \emptyset$
$\neg A$	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y: (x,y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
$\exists R.\top$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}: (x,y) \in R^{\mathcal{I}}\}$
R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, R is a role name

- Example constructs:
  - person ⊓ female
  - person ⊓ ¬female
  - person ⊓ ∃has_child.⊤
  - person ⊓ ∀has_child.⊥

More $\mathcal{AL}$ Family Members

- Disjunction ($\mathcal{U}$)

$$C \sqcup D \qquad C^{\mathcal{I}} \cup D^{\mathcal{I}}$$

- Full existential quantification ($\mathcal{E}$)

$$\exists R.C \qquad \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$

- Number restrictions ($\mathcal{N}$)

$$\exists_{\geq n} R \qquad \{x \in \Delta^{\mathcal{I}} \mid \|\{y \mid (x,y) \in R^{\mathcal{I}}\}\| \geq n\}$$
$$\exists_{\leq n} R \qquad \{x \in \Delta^{\mathcal{I}} \mid \|\{y \mid (x,y) \in R^{\mathcal{I}}\}\| \leq n\}$$

- Full negation ($\mathcal{C}$)

$$\neg C \qquad \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$$

- Example:

$$\text{person} \sqcap (\exists_{\leq 1} \text{has\_child} \sqcup (\exists_{\geq 3} \text{has\_child} \sqcap \exists \text{has\_child.female}))$$

DL as fragments of Predicate Logic

- Any concept D as unary predicate with 1 free variable
- Any role R as primitive binary predicate
- $\exists R.C$ corresponds to
  $\exists y.\ R(x,y) \wedge C(y)$
- $\forall R.C$ corresponds to
  $\forall y.\ R(x,y) \Rightarrow C(y)$
- $\exists_{\geq n} R$ corresponds to
  $\exists y_1,\ldots,y_n.\ R(x,y_1) \wedge \ldots \wedge R(x,y_n) \wedge \forall i<j.\ y_i \neq y_j$
- $\exists_{\leq n} R$ corresponds to
  $\forall y_1,\ldots,y_{n+1}.\ R(x,y_1) \wedge \ldots \wedge R(x,y_{n+1}) \Rightarrow \exists i<j.\ y_i = y_j$
- Last two examples demonstrate advantage of variable-free syntax

DL Knowledge Base

- DL Knowledge Base (KB) normally separated into 2 parts:
  - TBox is a set of axioms describing structure of domain (i.e., a conceptual schema), e.g.:
    - HappyFather ≡ Man ⊓ ∃hasChild.Female ⊓ …
    - Elephant ⊑ Animal ⊓ Large ⊓ Grey
    - transitive(ancestor)
  - ABox is a set of axioms describing a concrete situation (data), e.g.:
    - John:HappyFather
    - <John,Mary>:hasChild

Terminologies or TBoxes

- Statements about how concepts and roles are related to each other
- Terminology or TBox $\mathcal{T}$ is a set of axioms
  - interpretation $\mathcal{I}$ satisfies $\mathcal{T}$ if $\mathcal{I}$ satisfies all axioms in $\mathcal{T}$
- Terminological axioms
  - $C \sqsubseteq D$ $(R \sqsubseteq S)$ $\quad C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ $(R^{\mathcal{I}} \subseteq S^{\mathcal{I}})$
    - C is subsumed by D or D subsumes C
  - $C \equiv D$ $(R \equiv S)$ $\quad C^{\mathcal{I}} = D^{\mathcal{I}}$ $(R^{\mathcal{I}} = S^{\mathcal{I}})$
    - C is equivalent to D
- Definitions
  - if the left hand side of an axiom is a name
  - parent $\equiv$ person $\sqcap \exists$has_child person
  - fairy $\sqsubseteq$ person

Inference Services

- What type of inferences are interesting?
  - satisfiability of (named) concepts
  - subsumption of (named) concepts
  - equivalence of (named) concepts
  - disjointness of (named) concepts
- Description of concept parent
  - parent = person ⊓ ∃has_child.person
- We add two concepts
  - woman = female ⊓ person
  - mother = female ⊓ parent

Inference services based on Satisfiability

- All concept inference services can be reduced to concept satisfiability
- We assume service sat(C, $\mathcal{T}$), C a concept, $\mathcal{T}$ a TBox
- subsumes(C, D, $\mathcal{T}$) ≡ ¬ sat(¬C ⊓ D, $\mathcal{T}$)
  - C ⊒ D holds ↔ ¬(C ⊔ ¬D) unsatisfiable ↔ ¬C ⊓ D unsatisfiable
- equivalence(C, D, $\mathcal{T}$) ≡ subsumes(C, D, $\mathcal{T}$) ∧ subsumes(D, C, $\mathcal{T}$)
- disjoint(C, D, $\mathcal{T}$) ≡ ¬ sat(C ⊓ D, $\mathcal{T}$)

Inference service: concept subsumption

Is a Mother always a Woman?

Yes, Mother Subsumes Woman

Description logic reasoners offer the computation of a Subsumption (taxonomy) of all named concepts

Description Logic Reasoners

- For example:

FaCT++

- They offer reasoning services for multiple TBoxes and ABoxes.
- They run as background reasoning engines.

- They understand DIG, which is a simple protocol (based on HTTP) along with an XML Schema.

- Example: $Student \sqsubseteq Person$

```
<impliesc>
  <catom name="Student"/>
  <catom name="Person"/>
</impliesc>
```

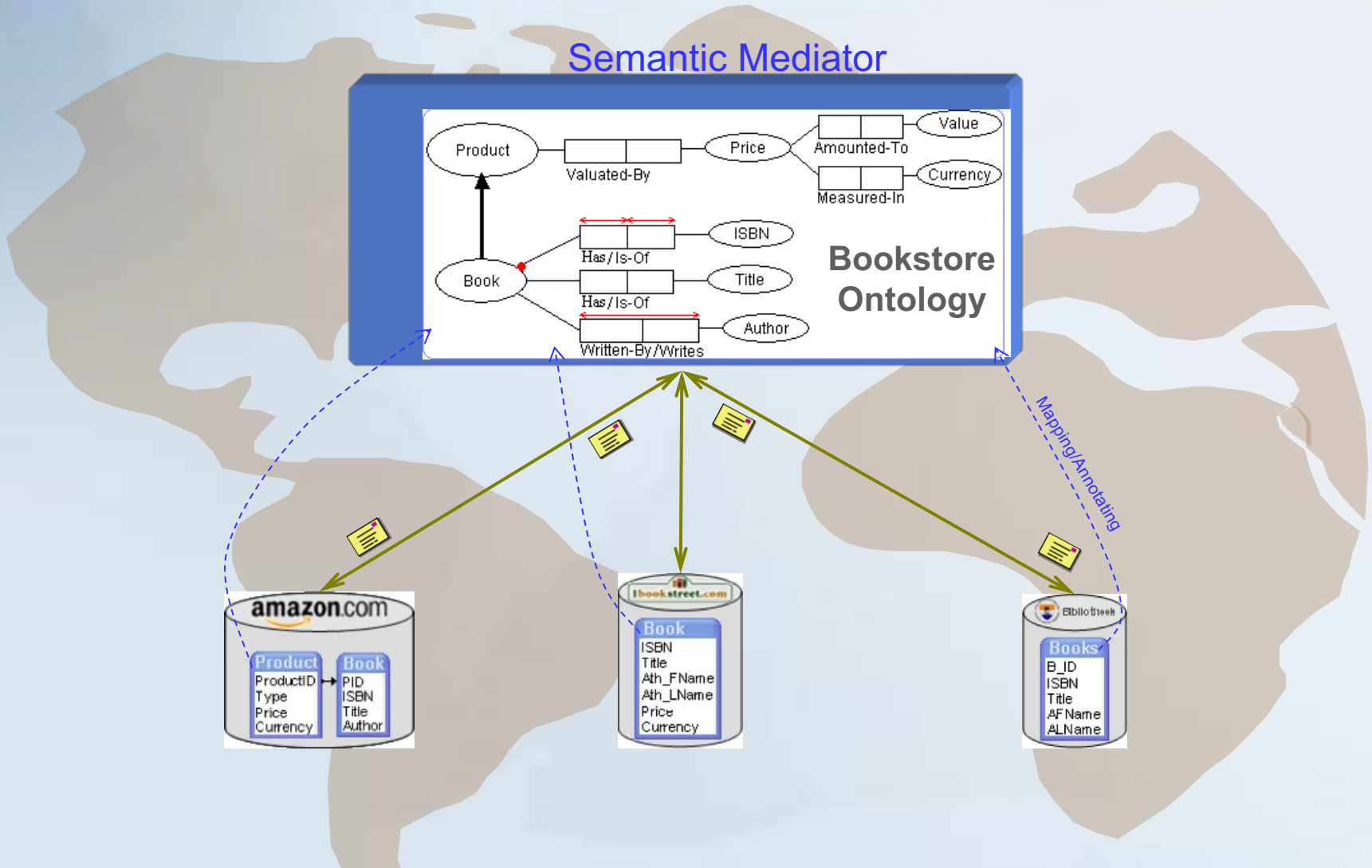
The $\mathcal{SHOIN}$ Description Logic

DL Syntax	Example	FOL Syntax
$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male	$C_1(x) \wedge \ldots \wedge C_n(x)$
$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer	$C_1(x) \vee \ldots \vee C_n(x)$
$\neg C$	¬Male	$\neg C(x)$
$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}	$x = x_1 \vee \ldots \vee x = x_n$
$\forall P.C$	∀hasChild.Doctor	$\forall y.P(x,y) \rightarrow C(y)$
$\exists P.C$	∃hasChild.Lawyer	$\exists y.P(x,y) \wedge C(y)$
$\leqslant nP$	⩽1hasChild	$\exists^{\leqslant n} y.P(x,y)$

- The logic underpinning OWL, the standard (W3C recommendation) Ontology Web Language.
- A compromise between expressive power and computational complexity.
- Does not support: n-ary relations, identification, functional dependencies.

Mapping ORM to $\mathcal{SHOIN}$

(Web) Information Systems



Mapping ORM to Description Logic

Why do we need this mapping for?

- Use ORM as a graphical notation for Ontology/description logic languages.

(The DL benefit from ORM)

- Reasoning on ORM schemes automatically.

(The ORM benefit from DL)

Reasoning Services

Examples of reasoning services:

- Satisfiability

To know whether a concept can be populated or not (e.g. because of some axioms contradicting each other)

$\mathbf{SAT}(C, \mathcal{T})$ iff there is a model $\mathcal{I}$ of $\mathcal{T}$ with $C^{\mathcal{I}} \neq \varnothing$

- Subsumption

To know whether a concept is subsuming another concept (e.g. to find unwanted or missing subsumptions)

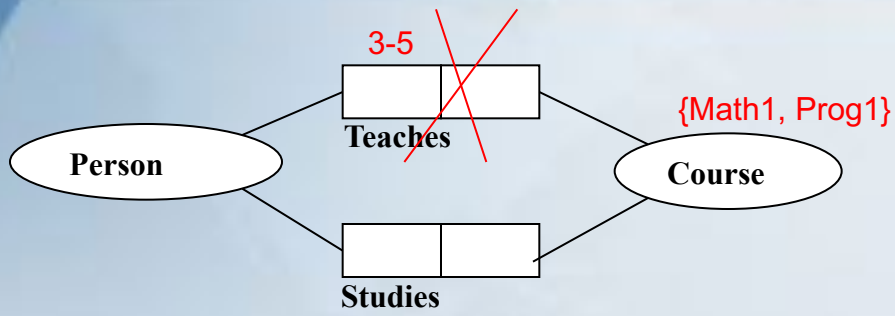
$\mathbf{SUBS}(C, D, \mathcal{T})$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all model $\mathcal{I}$ of $\mathcal{T}$

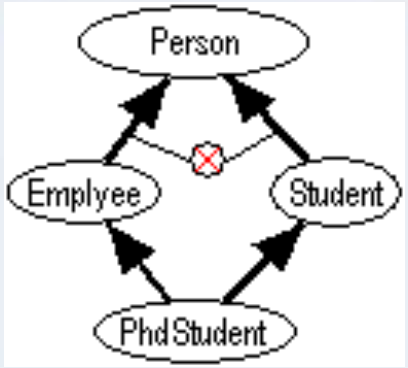
- Redundancies

To know whether two concepts are equal (e.g. to find out redundancies)

$\mathbf{EQUIV}(C, D, \mathcal{T})$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all model $\mathcal{I}$ of $\mathcal{T}$

Reasoning on ORM Schemes





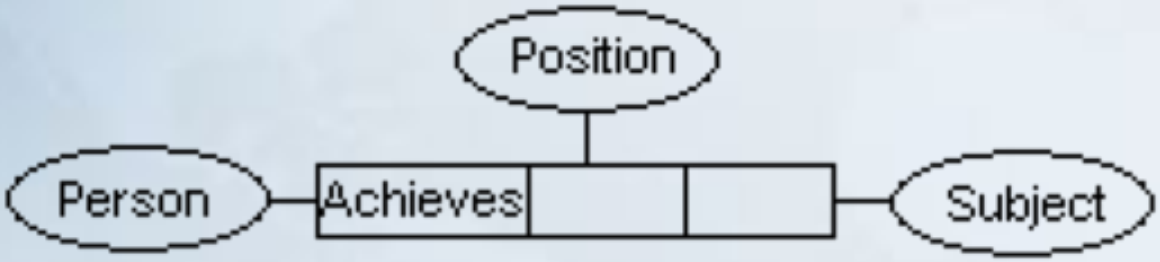
- Schema satisfiability: A schema is satisfiable if and only if there is at least one concept in the schema that can be populated. ➔ **Weak satisfiability**
- Concept satisfiability: A schema is satisfiable if and only if all concepts in the schema can be populated.
- Role satisfiability: A schema is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

Mapping ORM to *DLR* and *SHOIN*

n-ary relations:

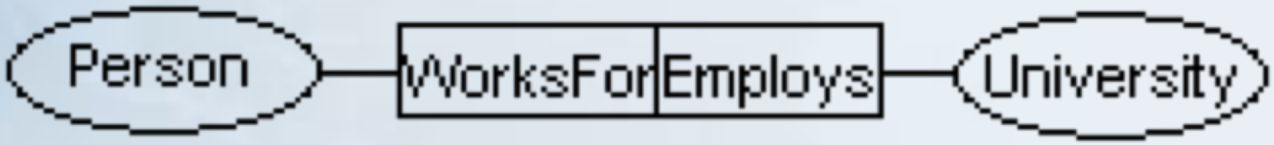


SHOIN

-- No Support --

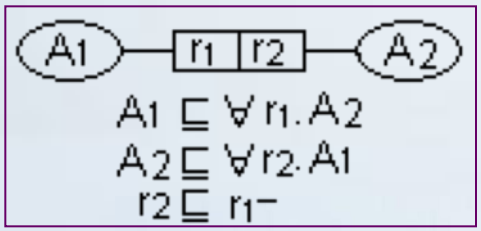
Mapping ORM to *DLR* and *SHOIN*

Binary relations:



SHOIN

$$\begin{aligned} \mathit{Person} &\sqsubseteq \forall \mathrm{WorksFor}.\mathrm{University} \\ \mathrm{University} &\sqsubseteq \forall\, \mathrm{Employs}.\mathit{Person} \\ \mathrm{Employs} &\sqsubseteq \mathrm{WorksFor}^{-} \end{aligned}$$



Mapping ORM to $\mathcal{DLR}$ and $\mathcal{SHOIN}$

Unary roles:

$\mathcal{SHOIN}$

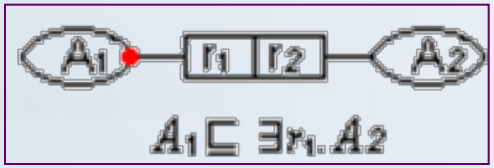
$$Person \sqsubseteq \forall Smokes.BOOLEAN$$

Mapping ORM to $\mathcal{DLR}$ and $\mathcal{SHOIN}$

Role Mandatory:

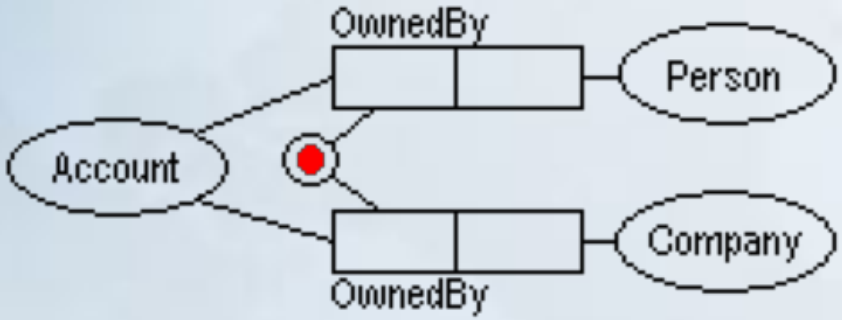
$\mathcal{SHOIN}$

$$Professor \sqsubseteq \exists\, WorksFor.University$$



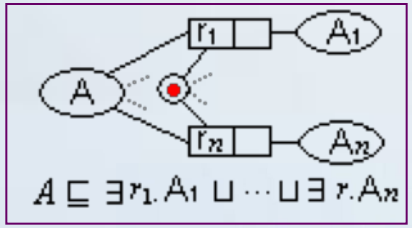
Mapping ORM to *DLR* and *SHOIN*

Disjunctive Mandatory:



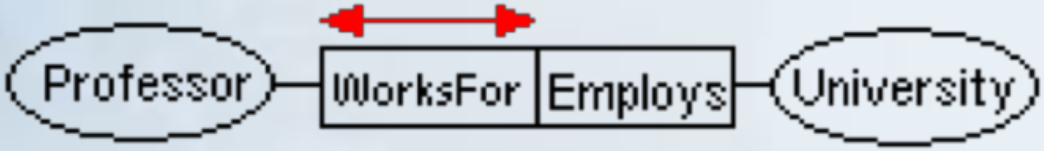
SHOIN

$$Account \sqsubseteq \exists OwnedBy1.\text{Person} \sqcup \exists OwnedBy2.\text{Company}$$



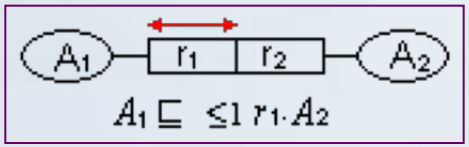
Mapping ORM to *DLR* and *SHOIN*

Role Uniqueness:



SHOIN

$$Professor \sqsubseteq \leq 1\ WorksFor.University$$



Mapping ORM to *DLR* and *SHOIN*

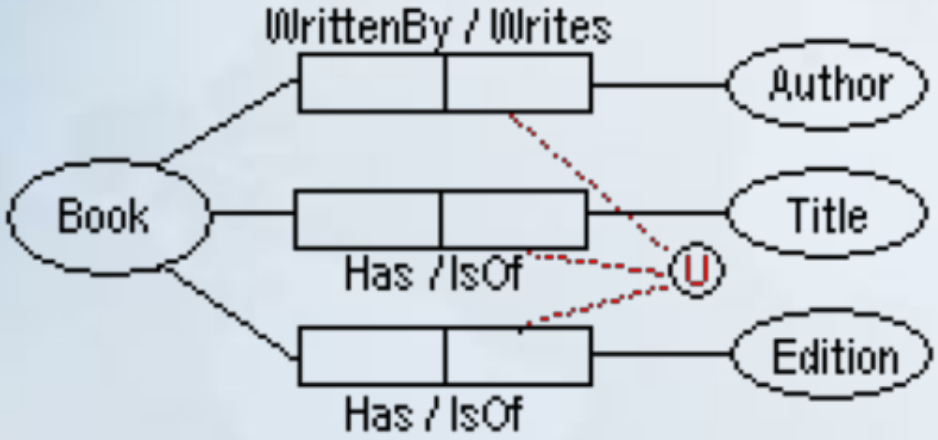
Predicate Mandatory:

SHOIN

-- No Support --

Mapping ORM to $\mathcal{DLR}$ and $\mathcal{SHOIN}$

External Mandatory:

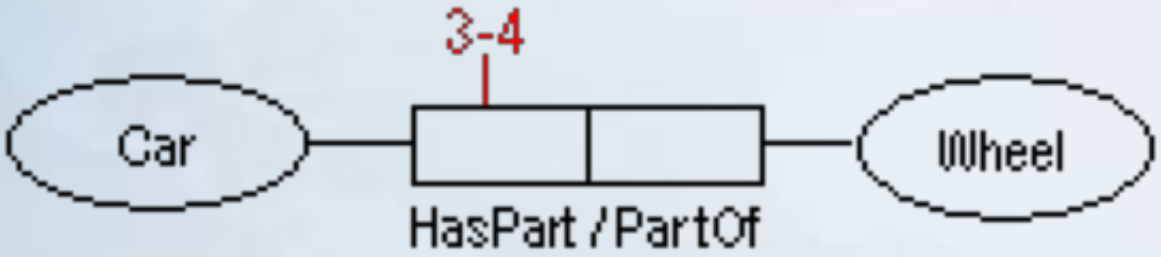


$\mathcal{SHOIN}$

-- No Support --

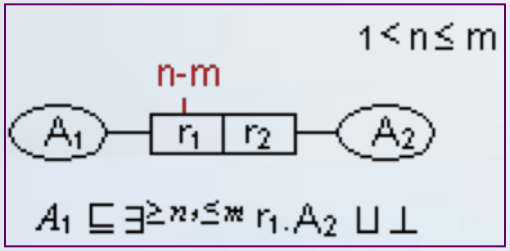
Mapping ORM to *DLR* and *SHOIN*

Role Frequency:



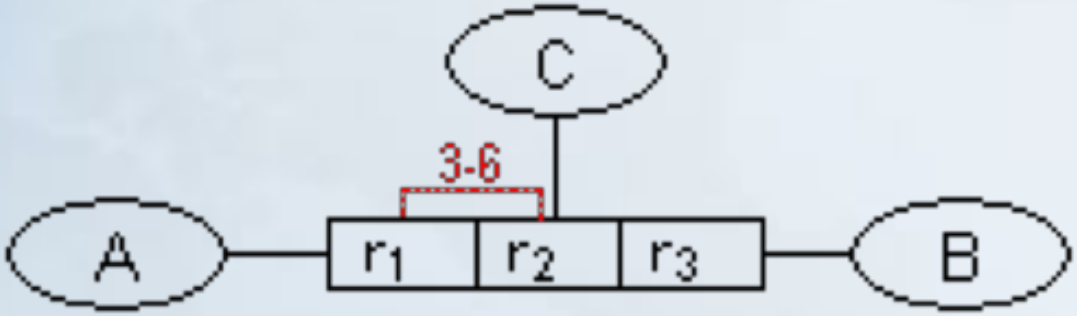
SHOIN

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



Mapping ORM to $\mathcal{DLR}$ and $\mathcal{SHOIN}$

Multiple-Role Frequency:

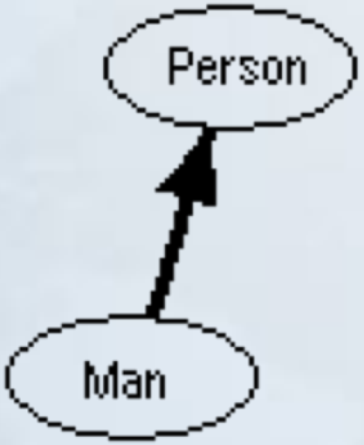


$\mathcal{SHOIN}$

-- No Support --

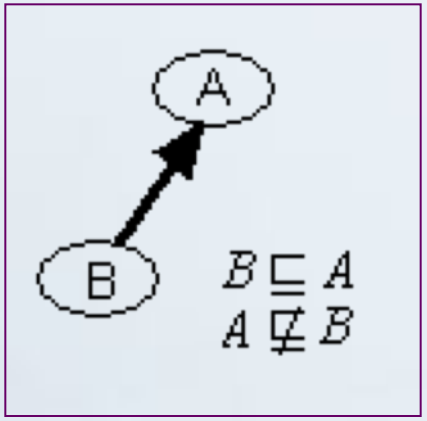
Mapping ORM to $\mathcal{DLR}$ and $\mathcal{SHOIN}$

Subtypes:



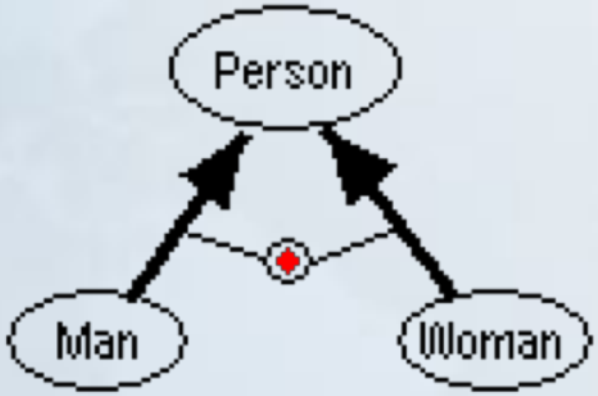
$\mathcal{SHOIN}$

$$Man \sqsubseteq Person$$
$$Person \not\sqsubseteq Man$$



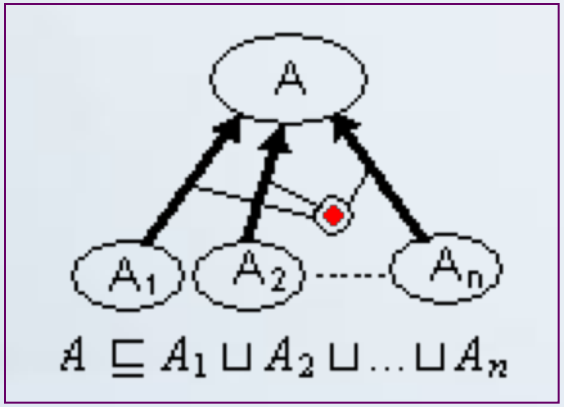
Mapping ORM to $\mathcal{DLR}$ and $\mathcal{SHOIN}$

Total Subtypes:



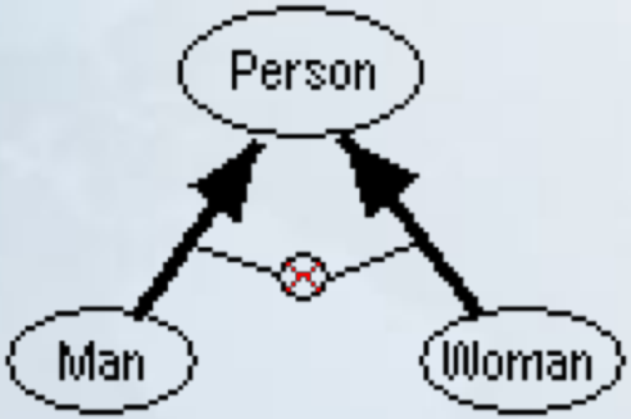
$\mathcal{SHOIN}$

$$Person \sqsubseteq Man \sqcup Woman$$



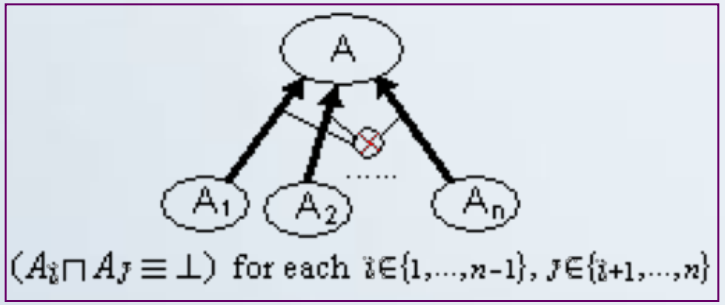
Mapping ORM to $\mathcal{DLR}$ and $\mathcal{SHOIN}$

Exclusive Subtypes:



$\mathcal{SHOIN}$

$$Man \sqcap Woman \equiv \bot$$



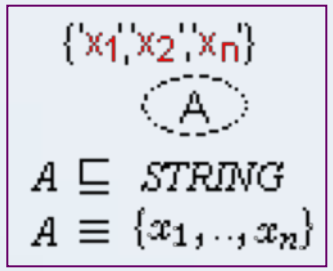
Mapping ORM to $\mathcal{DLR}$ and $\mathcal{SHOIN}$

Value Constraint:

STRING:

$\mathcal{SHOIN}$

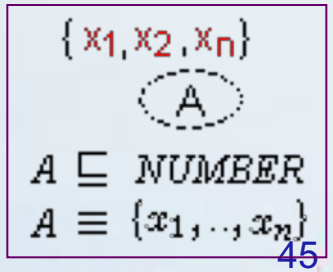
$$Gender \sqsubseteq STRING$$
$$Gender \equiv \{Male, Female\}$$



Number:

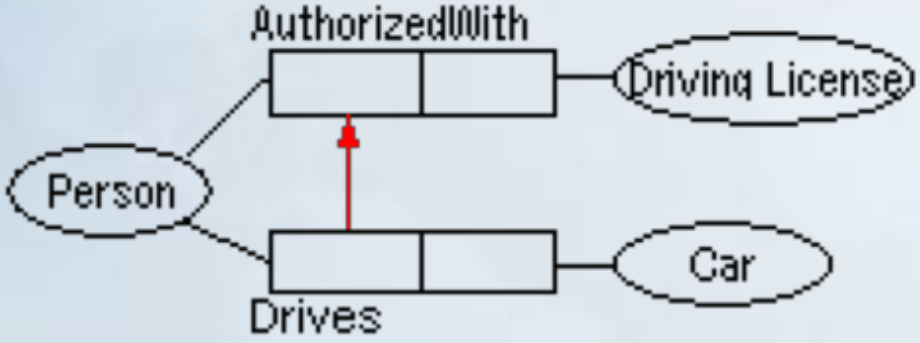
$\mathcal{SHOIN}$

$$ZipCode \sqsubseteq NUMBER$$
$$ZipCode \equiv \{1040, 1160\}$$



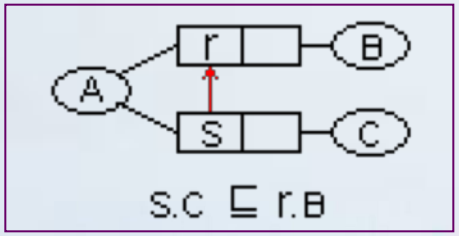
Mapping ORM to DLR and SHOIN

Role Subset Constraint:



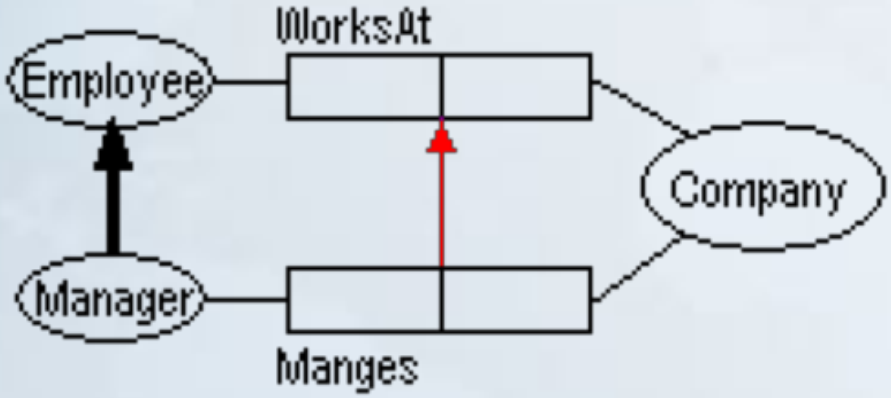
SHOIN

$$Drives.Car \sqsubseteq AuthorizedWith.DrivingLicense$$



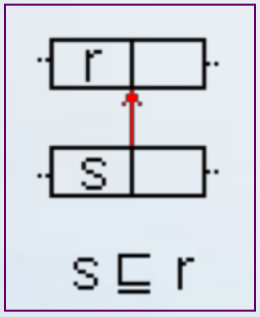
Mapping ORM to DLR and SHOIN

Predicate Subset Constraint:



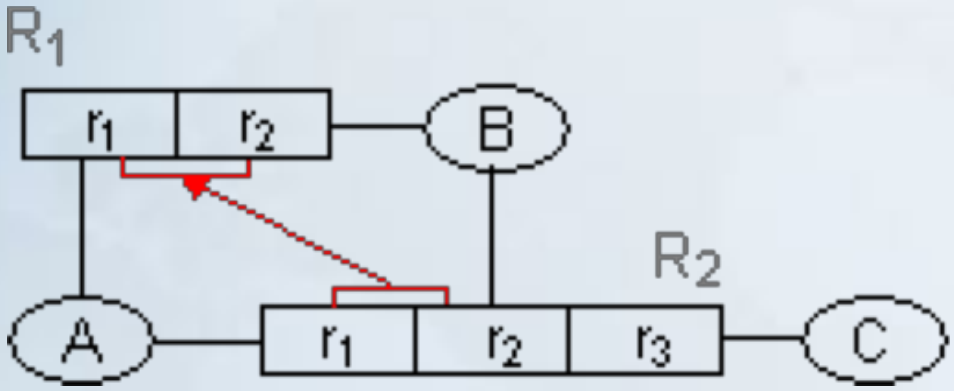
SHOIN

$$Manges \sqsubseteq WorksAt$$



Mapping ORM to DLR and SHOIN

Role-sequence Subset Constraint:

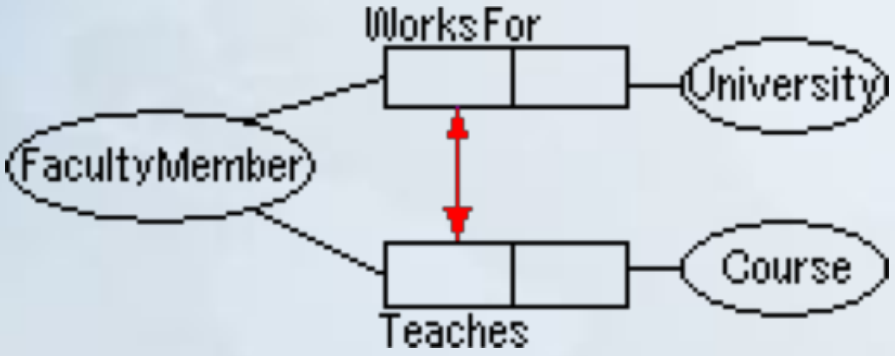


SHOIN

-- No Support --

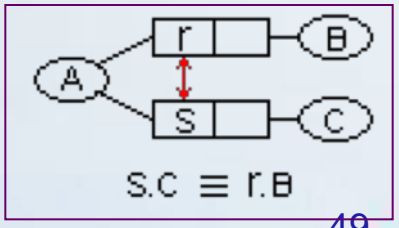
Mapping ORM to DLR and SHOIN

Role Equality Constraint:



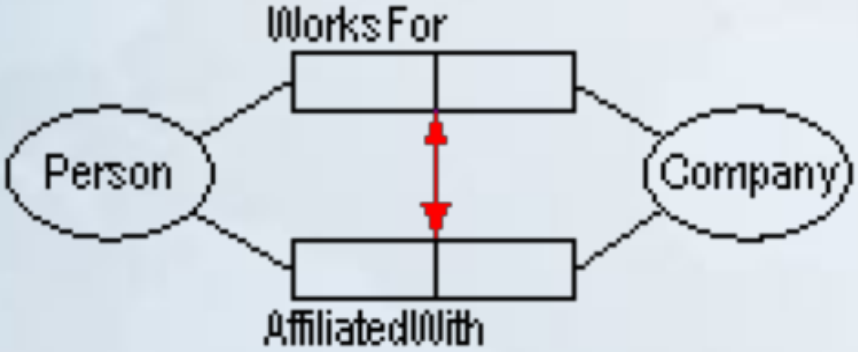
SHOIN

$$WorksFor.University \equiv Teaches.Course$$



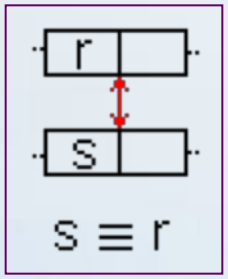
Mapping ORM to DLR and SHOIN

Predicate Equality Constraint:



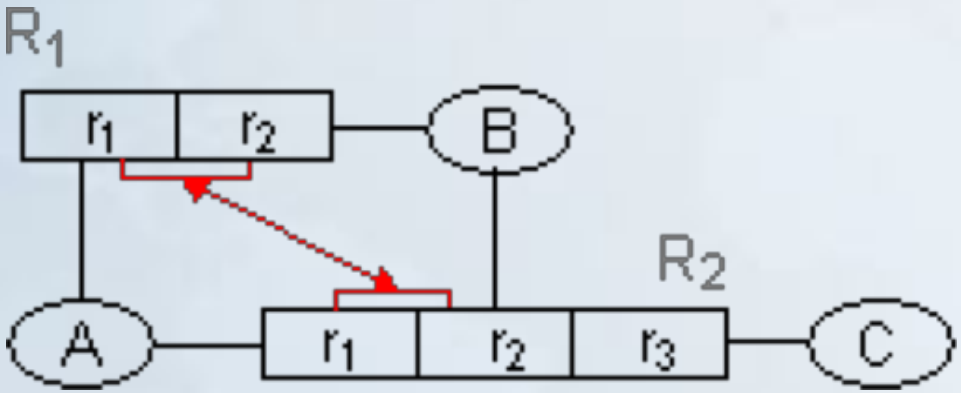
SHOIN

$$WorksFor \equiv AffiliatedWith$$



Mapping ORM to DLR and SHOIN

Role-sequence Equality Constraint:

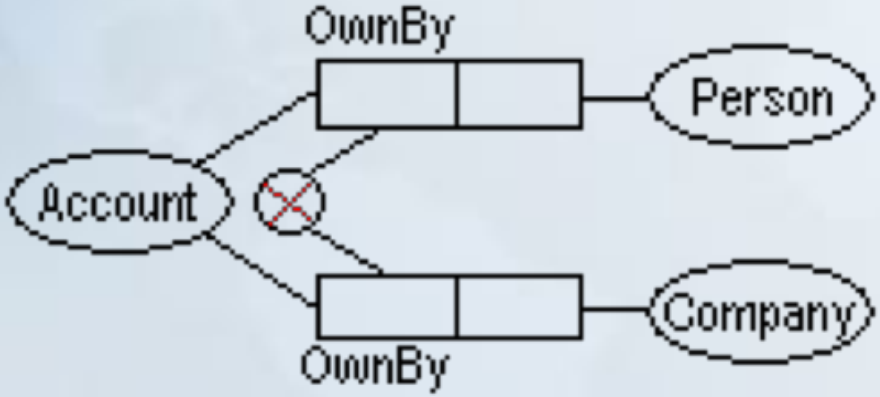


SHOIN

-- No Support --

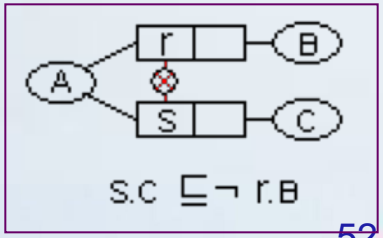
Mapping ORM to DLR and SHOIN

Role exclusion Constraint:



SHOIN

$$OwnedBy.Person \sqsubseteq \neg\, OwnedBy.Account$$

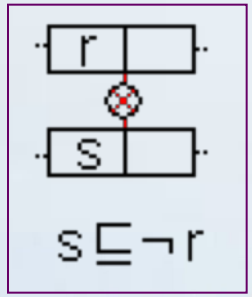


Mapping ORM to DLR and SHOIN

Predicate Exclusion Constraint:

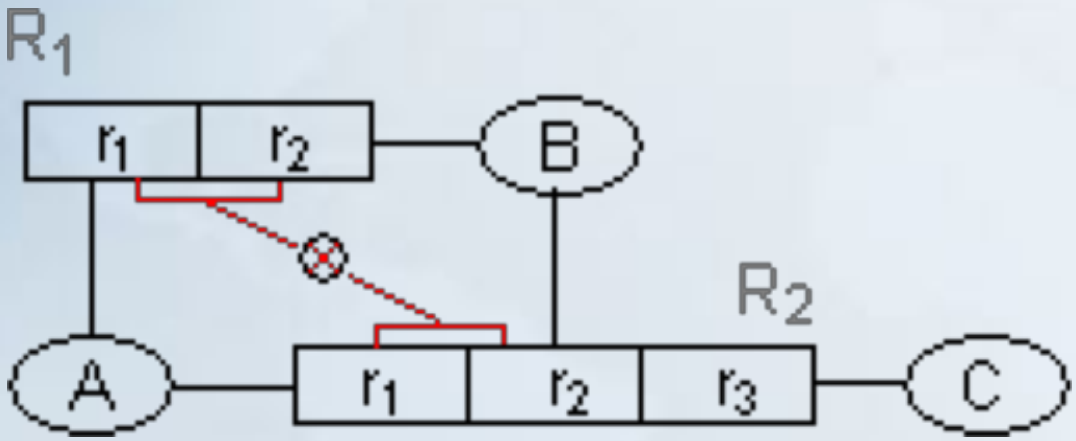
SHOIN

$Writes \sqsubseteq \neg Reviews$



Mapping ORM to DLR and SHOIN

Role- sequence Exclusion Constraint:

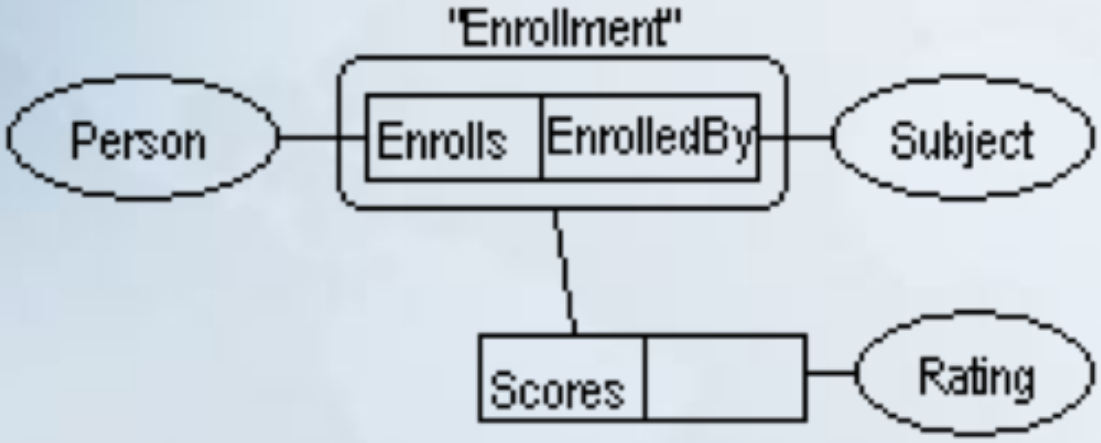


SHOIN

-- No Support --

Mapping ORM to *DLR* and *SHOIN*

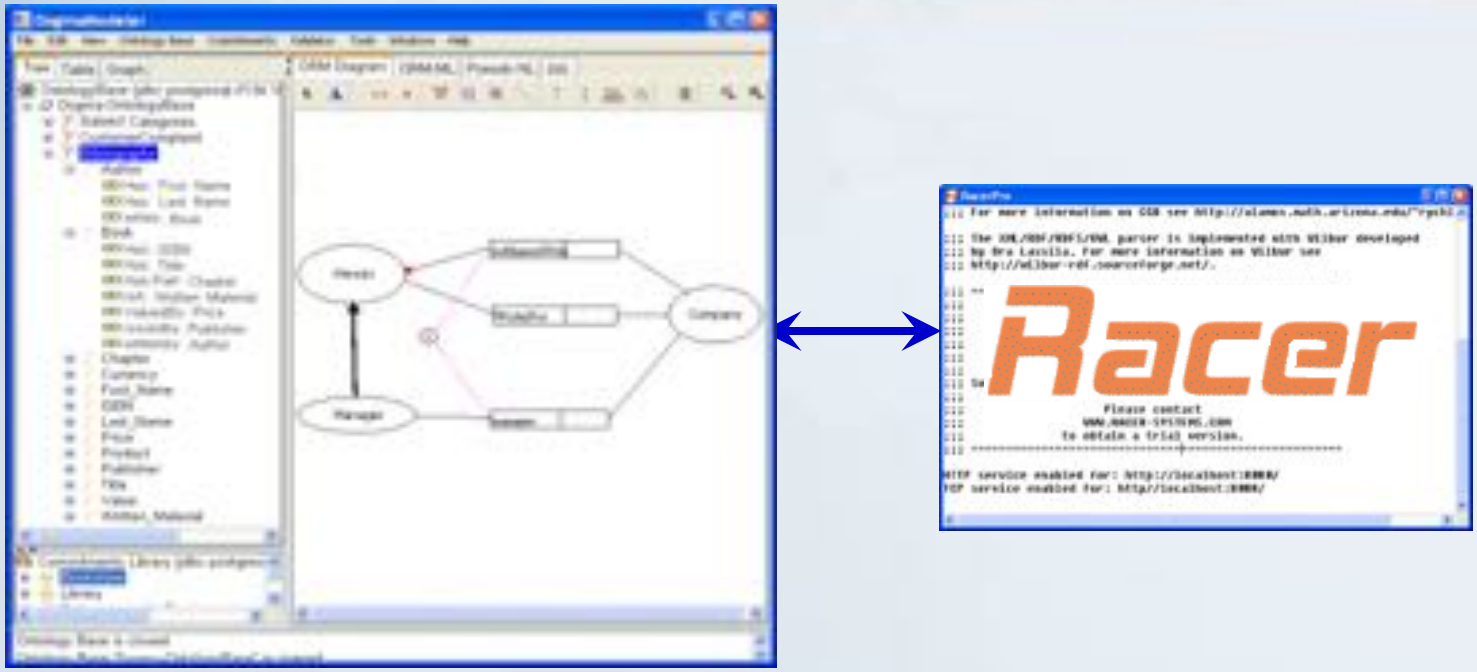
Objectification:



SHOIN

-- No Support --

DogmaModeler



- DogmaModeler is an ontology engineering tool
- It uses ORM as a graphical notation
- It uses Racer as a background reasoning engine

Other functionalities of DogmaModeler:

- Verbalization of ORM into 11 human languages.
- Modularization of auto composition of ORM schemes
- ….

Home Work

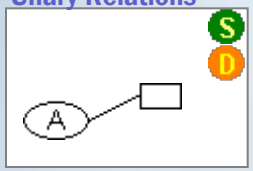
- Map your ORM model (project 3) into description logic.
- Reason about your ORM model using Racer.
- ➔MS Word file containing:

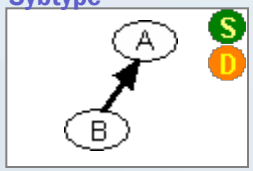
1. You ORM model
2. Map ORM into DL
3. Map ORM into DIG and load it to Racer
4. Screen shorts of Racer results, that (a)your model is satisfiable, (b) your model is unsatisfiable (make something wrong)

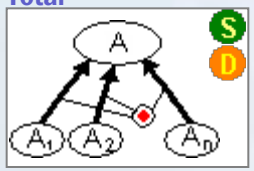
The next Lecture assumes that you know **XML**, if not please read about it.

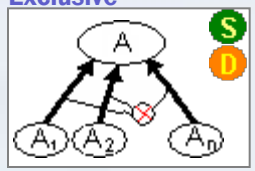
Conclusions and Discussion

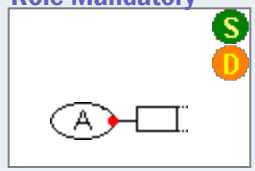
ORM Constructs and constraints

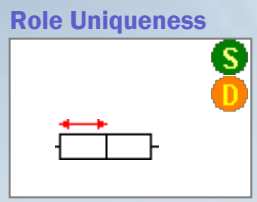


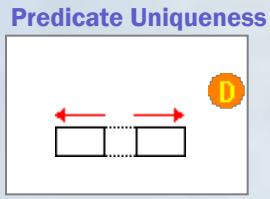


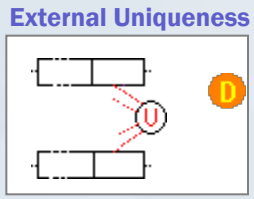


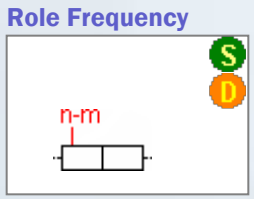


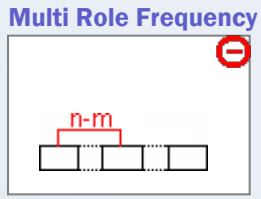


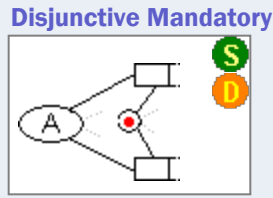


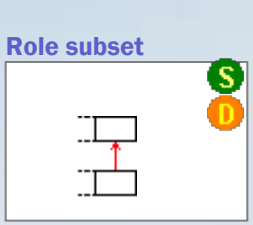


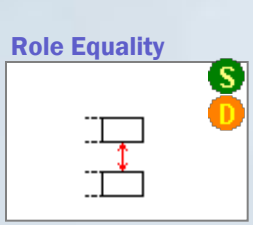


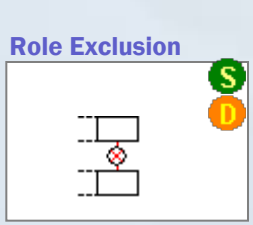


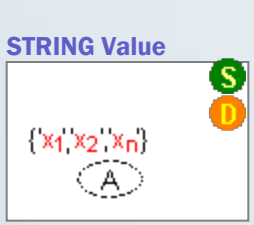


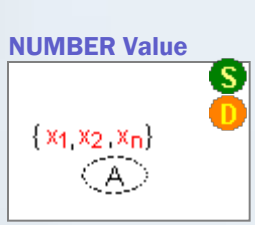


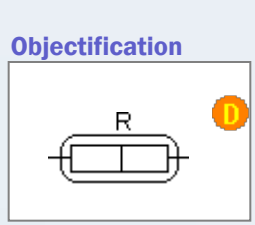


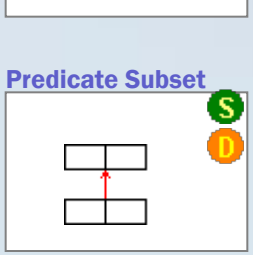


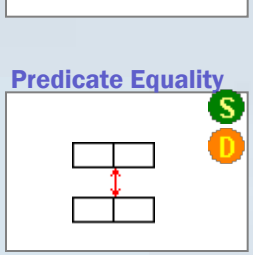


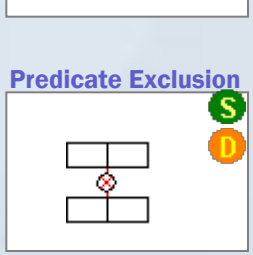


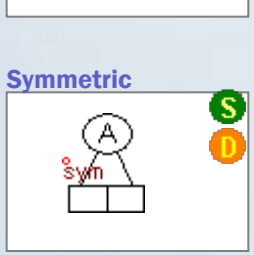


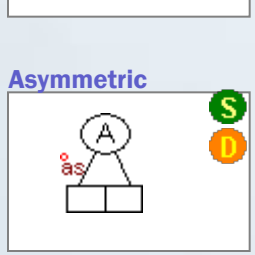


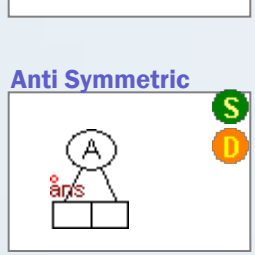


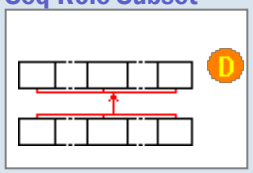


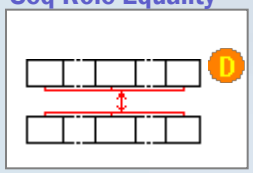


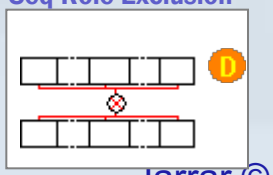


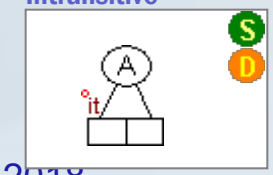


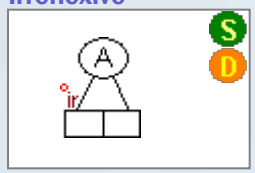






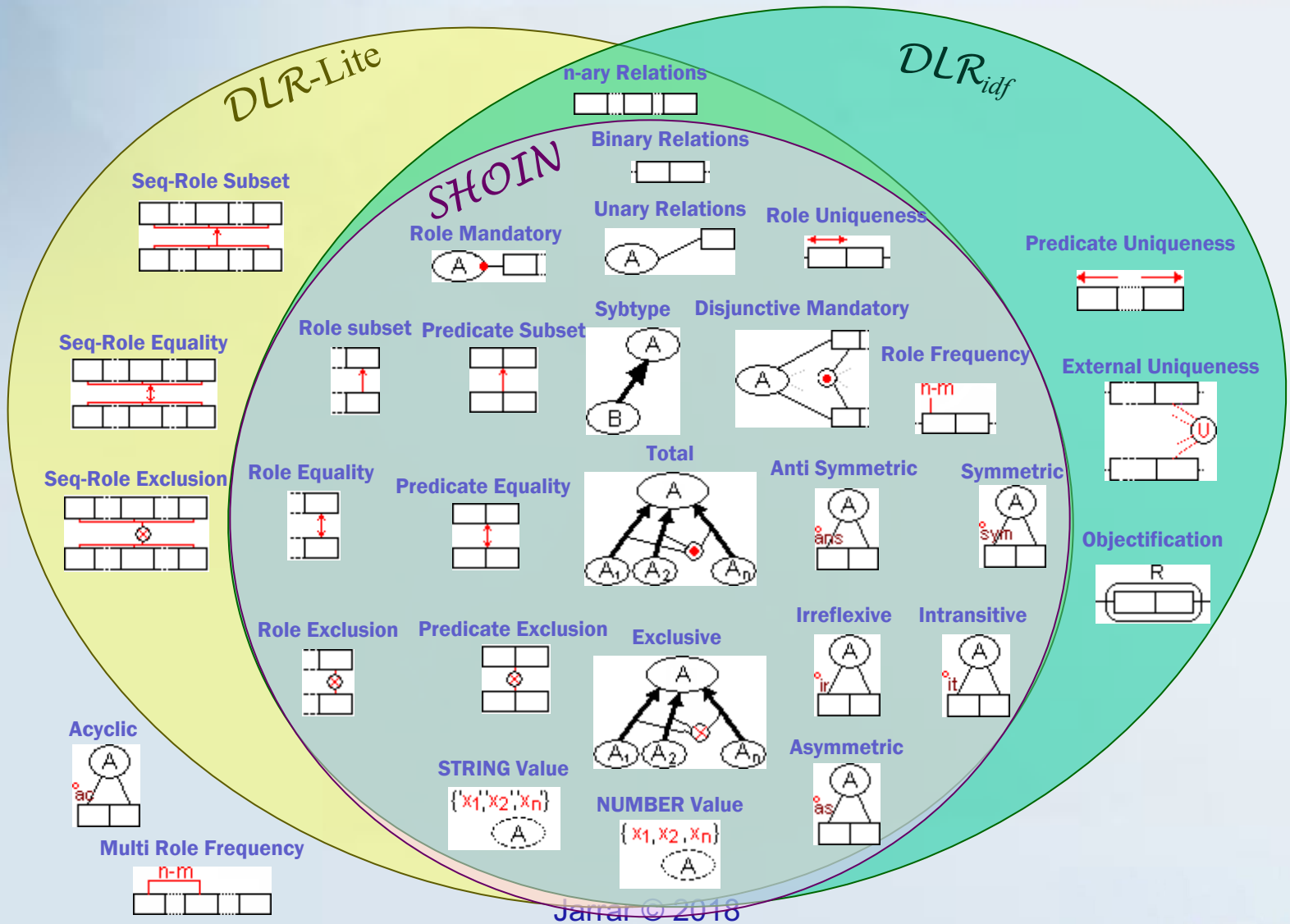




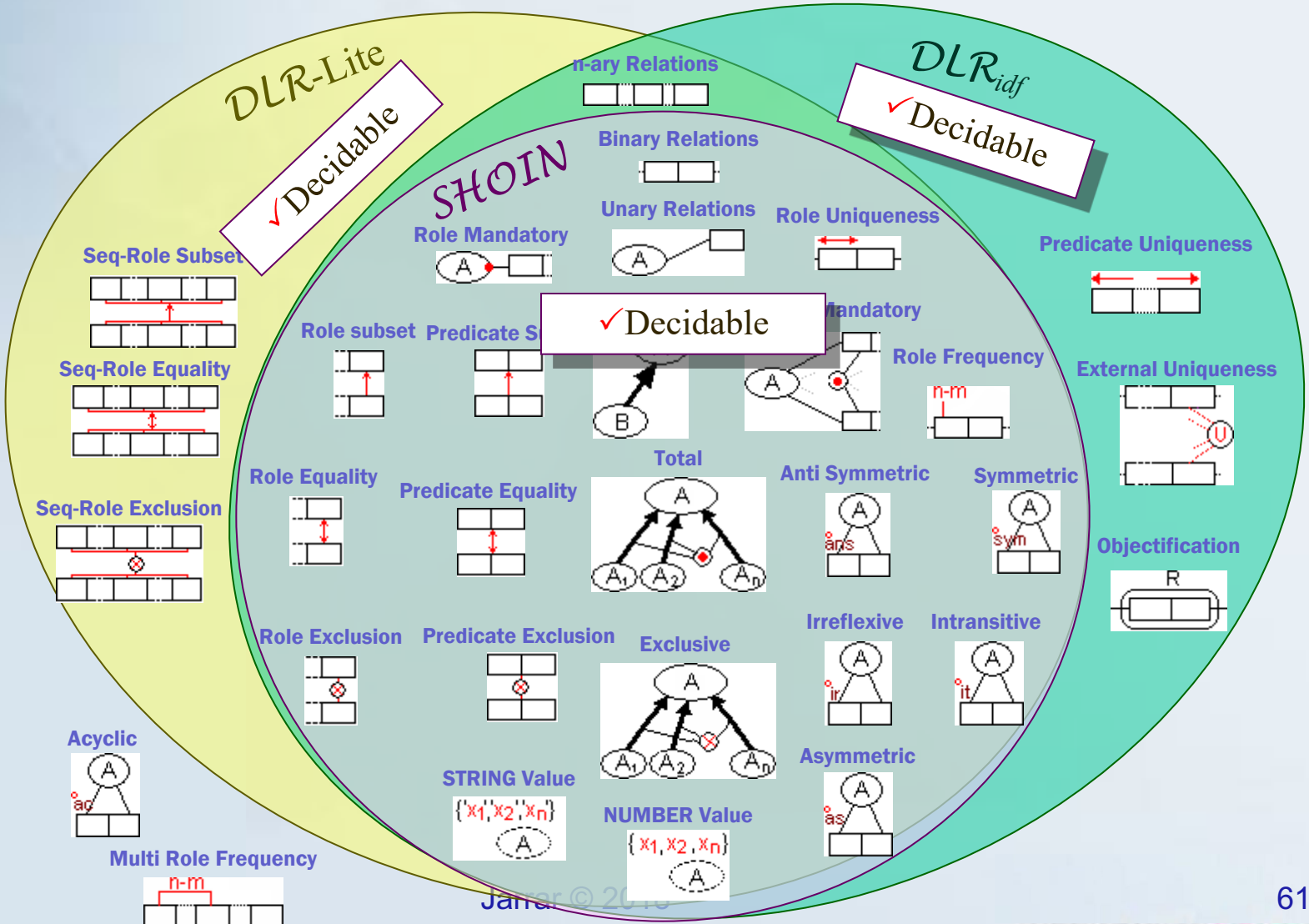


Mapped to DLR (D) Mapped to SHONIOWL (S) Cannot map (-)

Lesson 1: what is mapped to what



Lesson 2: Decidable fragments



Description Logic Reasoners

a short tutorial on DIG

Description Logic Reasoners

- For example:

FaCT++

- They offer reasoning services for multiple TBoxes and ABoxes.
- They run as background reasoning engines.

- They understand DIG, which is a simple protocol (based on HTTP) along with an XML Schema.

- Example: $Student \sqsubseteq Person$

```
<impliesc>
  <catom name="Student"/>
  <catom name="Person"/>
</impliesc>
```

DIG Interface (http://dig.sourceforge.net/)

S. Bechhofer: The DIG Description Logic Interface: DIG/1.1
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.3837&rep=rep1&type=pdf

DIG Interface (http://dig.sourceforge.net/)

S. Bechhofer: The DIG Description Logic Interface: DIG/1.1
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.3837&rep=rep1&type=pdf

DIG Protocol

- DIG is only an XML schema for a description logic along with ask/tell functionality
- You write a new Knowledge base (using the DIG XML syntax), and send it to Racer using the TELL functionality.
- You can then write you Query/Question (using the DIG, XML syntax), and send it to Racer using the ASK functionality.

- You may communicate with the Racer through HTTP or SOAP.

Create e a new Knowledge Base

The newKB Message

```
<?xml version="1.0" encoding="UTF-8"?>
<newKB
xmlns="http://dl.kr.org/dig/2003/02/lang"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
http://dl-web.man.ac.uk/dig/2003/02/dig.xsd"/>
```

The Response Message

```
<?xml version="1.0" encoding="UTF-8"?>
<response
xmlns="http://dl.kr.org/dig/2003/02/lang"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
http://dl-web.man.ac.uk/dig/2003/02/dig.xsd">
<kb uri="urn:uuid:abcdefgh-1234-12345689ab"/>
```

This URI should then be used during TELL and ASK requests made against the knowledge base

Tell Syntax

A TELL request must contain in its body a tells element, which itself consists of a number of tell statements.

Example: Driver ⊑ Person ⊓ ∃Drives.Vehicle

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tells
    xmlns="http://dl.kr.org/dig/2003/02/lang"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
    http://dl-web.man.ac.uk/dig/2003/02/dig.xsd"
    uri="urn:uuid:abcdefgh-1234-1234-12345689ab">
      <defconcept name="driver"/>
       <equalc>
          <catom name="driver"/>
          <and>
             <catom name="person"/>
             <some>
                <ratom name="drives"/>
                  <catom name="vehicle"/>
               </some>
           </and>
       </equalc>
    <defconcept name="person"/>
    <defconcept name="vehicle"/>
    <defrole name="drives"/>
</tells>
```

Tell Syntax

Tell Language	
Primitive Concept Introduction	<defconcept name="CN"/> <defrole name="CN"/> <deffeature name="CN"/> <defattribute name="CN"/> <defindividual name="CN"/>
Concept Axioms	<impliesc>C1 C2</impliesc> <equalc>C1 C2</equalc> <disjoint>C1... Cn</disjoint>
Role Axioms	<impliesr>R1 R2</impliesc> <equalr>R1 R2</equalr> <domain>R E</domain> <range>R E</range> <rangeint>R</rangeint> <rangestring>R</rangestring> <transitive>R</transitive> <functional>R</functional>
Individual Axioms	<instanceof>I C</instanceof> <related>I1 R I2</related> <value>I A V</value>

Concept Language	
Primitive Concepts	<top/> <bottom/> <catom name="CN"/>
Boolean Operators	<and>E1... En</and> <or>E1... En</or> <not>E</not>
Property Restrictions	<some>R E</some> <all>R E</all> <atmost num="n">R E</atmost> <atleast num="n">R E</atleast> <iset>I1... In</iset>
Concrete Domain Expressions	<defined>A</defined> <stringmin val="s">A</stringmin> <stringmax val="s">A</stringmax> <stringequals val="s">A</stringequals> <stringrange min="s" max="t">A</stringrange> <intmin val="i">A</intmin> <intmax val="i">A</intmax> <intequals val="i">A</intequals> <intrange min="i" max="j">A</intrange>
Role Expressions	<ratom name="CN"/> <feature name="CN"/> <inverse>R</inverse> <attribute name="CN"/> <chain>F1... FN A</chain> Individuals <individual name="CN"/>

Ask Syntax

- An ASK request must contain in its body an asks element.
- Multiple queries in one request is possible.

$KB \models Vehicle$

asks about satisfiability of the Vehicle concept

$a \mid \Sigma \models Peron(a) \sqcap \exists Drives.Vehicle$

asks for all those concepts subsumed by the description given, i.e. all the drivers

$C \mid \Sigma \models C(JohnSmith)$

asks for the known types of the given individual

```
<?xml version="1.0"?>
<asks
    xmlns="http://dl.kr.org/dig/2003/02/lang">
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang"
    http://dl-web.man.ac.uk/dig/2003/02/dig.xsd"
    uri="urn:uuid:abcdefgh-1234-1234-12345689ab">
  <satisfiable id="q1">
    <catom name="Vehicle"/>
  </satisfiable>
  <descendants id="q2">
    <and>
      <catom name="person"/>
      <some>
        <ratom name="drives"/>
        <catom name="vehicle"/>
      </some>
    </and>
  </descendants>
  <types id="q3">
    <individual name="JohnSmith"></individual>
  </types>
</asks>
```

Ask Syntax

Ask Language	
Primitive Concept Retrieval	<allConceptNames/> <allRoleNames/> <allIndividuals/>
Satisfiability	<satisfiable>C</satisfiable> <subsumes>C1 C2</subsumes> <disjoint>C1 C2</disjoint>
Concept Hierarchy	<parents>C</parents> <children>C</children> <ancestors>C</ancestors> <descendants>C<descendants/> <equivalents>C</equivalents>
Role Hierarchy	<rparents>R</rparents> <rchildren>R</rchildren> <rancestors>R</rancestors> <rdescendants>R<rdescendants/>
Individual Queries	<instances>C</instances> <types>I</types> <instance>I C</instance> <roleFillers>I R</roleFillers> <relatedIndividuals>R</relatedIndividuals>

References

1. Mustafa Jarrar: Mapping ORM Into The SHOIN/OWL Description Logic- Towards A Methodological And Expressive Graphical Notation For Ontology Engineering . In OTM 2007 workshops: Proceedings of the International Workshop on Object-Role Modeling (ORM'07). Pages (729-741), LNCS 4805, Springer. ISBN: 9783540768890. Portogal. November, 2007
2. Mustafa Jarrar: Towards Automated Reasoning On ORM Schemes. -Mapping ORM Into The DLR_idf Description Logic. In proceedings of the 26th International Conference on Conceptual Modeling (ER 2007). Pages (181-197). LNCS 4801, Springer. Auckland, New Zealand. ISBN 9783540755623. November 2007
3. Mustafa Jarrar and Robert Meersman: Ontology Engineering -The DOGMA Approach. Book Chapter in "Advances in Web Semantics I". Chapter 3. Pages 7-34. LNCS 4891, Springer.ISBN:978-3540897835. (2008).
4. Mustafa Jarrar, Anton Deik, Bilal Faraj: Ontology-Based Data And Process Governance Framework -The Case Of E-Government Interoperability In Palestine . In pre-proceedings of the IFIP International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA'11). Pages(83-98). ISBN 978-88-903120-2-1. Campione, Italy. June 30, 2011.
5. Mustafa Jarrar and Stijn Heymans: Unsatisfiability Reasoning In ORM Conceptual Schemes. In Current Trends in Database Technology - EDBT 2006: Proceeding of the IFIP-2.6 International Conference on Semantics of a Networked. Pages (517-534). LNCS 4254, Springer. Munich, Germany. ISBN: 3540467882. March 2006.
6. Mustafa Jarrar and Stijn Heymans: Towards Pattern-Based Reasoning For Friendly Ontology Debugging . Journal of Artificial Intelligence Tools. Volume 17. No.4. World Scientific Publishing. August 2008.
7. Mustafa Jarrar, Maria Keet, and Paolo Dongilli: Multilingual Verbalization Of ORM Conceptual Models And Axiomatized Ontologies. Technical report. STARLab, Vrije Universiteit Brussel, February 2006.
8. Sergey Lukichev and Mustafa Jarrar: Graphical Notations For Rule Modeling . Book chapter in "Handbook of Research on Emerging Rule-Based Languages and Technologies". IGI Global. ISBN:1-60566-402-2. (2009)
9. Mustafa Jarrar: Modularization And Automatic Composition Of Object-Role Modeling (ORM) Schemes .OTM 2005 Workshops: Proceedings of the Object-Role Modeling (ORM'05). Pages (613-625). LNCS 3762, Springer. ISBN: 3540297391. 2005.
10. Mustafa Jarrar: Towards Methodological Principles For Ontology Engineering. PhD Thesis. Vrije Universiteit Brussel. (May 2005)
11. Mustafa Jarrar, Jan Demey, and Robert Meersman: On Using Conceptual Data Modeling For Ontology Engineering . Journal on Data Semantics, Special issue on "Best papers from the ER/ODBASE/COOPIS 2002 Conferences". LNCS 2800. No 1. Springer. 2003.
12. Jan Demey, Mustafa Jarrar, and Robert Meersman: A Markup Language For ORM Business Rules . Proceedings of the International Workshop on Rule Markup Languages for Business Rules on the Semantic Web (RuleML 2002). Pages(107-128). Volume 60. CEUR Workshop Proceedings. ISSN 1613-0073. June 2002
13. Mustafa Jarrar: Towards Effectiveness And Transparency In E-Business Transactions, An Ontology For Customer Complaint Management . A book chapter in "Semantic Web Methodologies for E-Business Applications". chapter 7. IGI Global. (2008)
14. Mustafa Jarrar: ORM Markup Language, Version 3 . Technical Report. STAR Lab, Vrije Universiteit Brussel, Belgium. January 2007