Designing an appropriate logic for conceptual modelling languages

(on devising a formalism that'll fit with need or use)

C. Maria Keet[1]

Department of Computer Science, University of Cape Town, South Africa,
mkeet@cs.uct.ac.za

Workshop on Logic, Algebra and Category Theory (LAC 2018)
12-16 February 2018, Melbourne, Australia

[1] Joint work with Pablo Rubén Fillottrani, Universidad Nacional del Sur, Bahía Blanca, Argentina

Outline

Outline

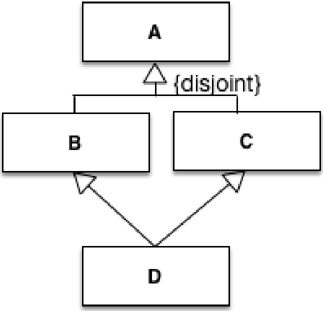
Context

- I'm mostly a logic user—LAC18 is alike 5-day window shopping (shopping spree?) to see if there's something that satisfies some of the needs in computing
- This talk: on those requirements, solutions, some 'loose ends', and things that might possibly be done more elegantly (?)

Context

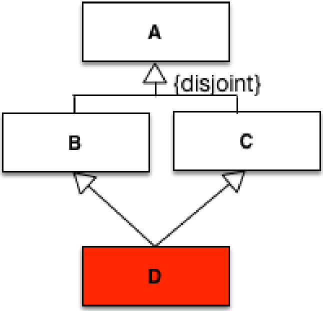
- Conceptual data modelling for complex system development and information integration
- Languages for conceptual modelling:
  - UML Class Diagrams, for generating object-oriented code
  - ER and EER, for generating relational databases
  - ORM and ORM2, for generating OO code, relational databases, link to business rules
- Need formal basis for model linking and integration, tools, and techniques

Conceptual data models–UML Class Diagram, inferences



$B \sqsubseteq A$	$\forall x(B(x) \rightarrow A(x))$
$C \sqsubseteq A$	$\forall x(C(x) \rightarrow A(x))$
$D \sqsubseteq B$	$\forall x(D(x) \rightarrow B(x))$
$D \sqsubseteq C$	$\forall x(D(x) \rightarrow C(x))$
$B \sqcap C \sqsubseteq \bot$	$\forall x(B(x) \land C(x) \rightarrow \bot)$

Conceptual data models–UML Class Diagram, inferences



$B \sqsubseteq A$ $\quad \forall x(B(x) \rightarrow A(x))$

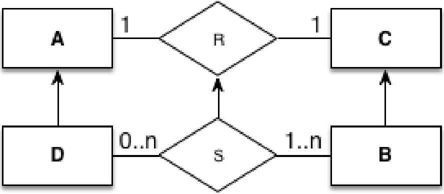
$C \sqsubseteq A$ $\quad \forall x(C(x) \rightarrow A(x))$

$D \sqsubseteq B$ $\quad \forall x(D(x) \rightarrow B(x))$

$D \sqsubseteq C$ $\quad \forall x(D(x) \rightarrow C(x))$

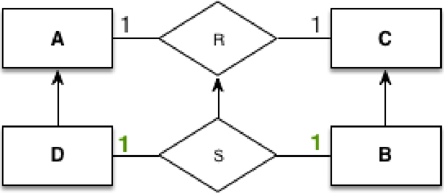
$B \sqcap C \sqsubseteq \bot$ $\quad \forall x(B(x) \land C(x) \rightarrow \bot)$

Conceptual data models–EER diagram, inferences



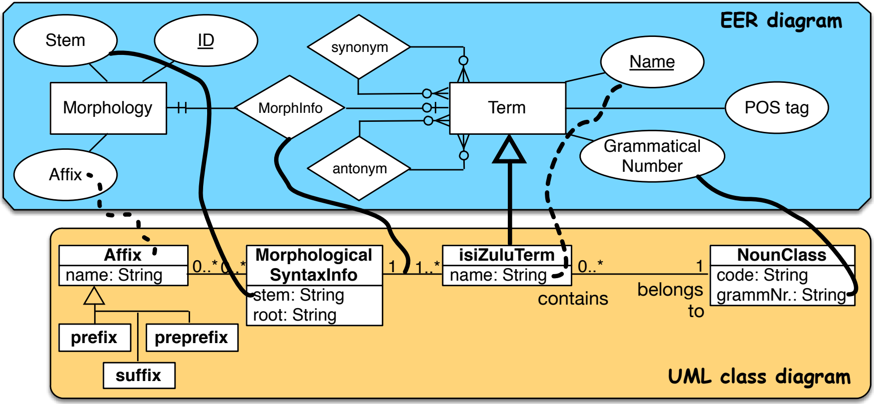
$R \sqsubseteq [i]A \sqcap [j]C$
$S \sqsubseteq [i]D \sqcap [j]B$
$A \sqsubseteq =1\, [i]R$
$C \sqsubseteq =1\, [j]R$
$B \sqsubseteq \exists[i]R$
$\forall x, y(R(x, y) \rightarrow A(x) \land C(y))$
$\forall x, y(S(x, y) \rightarrow D(x) \land B(y))$
$\forall x(A(x) \rightarrow \exists^{=1} y(R(x, y)))$
$\forall y(C(y) \rightarrow \exists^{=1} x(R(x, y)))$
$\forall y(B(y) \rightarrow \exists x(S(x, y)))$

Conceptual data models–EER diagram, inferences

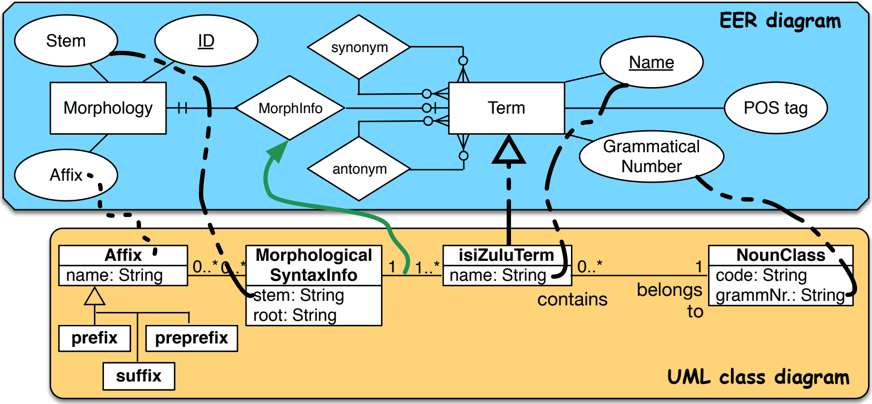


$R \sqsubseteq [i]A \sqcap [j]C$
$S \sqsubseteq [i]D \sqcap [j]B$
$A \sqsubseteq = 1\,[i]R$
$C \sqsubseteq = 1\,[j]R$
$B \sqsubseteq \exists[i]R$
$\forall x, y(R(x, y) \rightarrow A(x) \wedge C(y))$
$\forall x, y(S(x, y) \rightarrow D(x) \wedge B(y))$
$\forall x(A(x) \rightarrow \exists^{=1} y(R(x, y)))$
$\forall y(C(y) \rightarrow \exists^{=1} x(R(x, y)))$
$\forall y(B(y) \rightarrow \exists x(S(x, y)))$

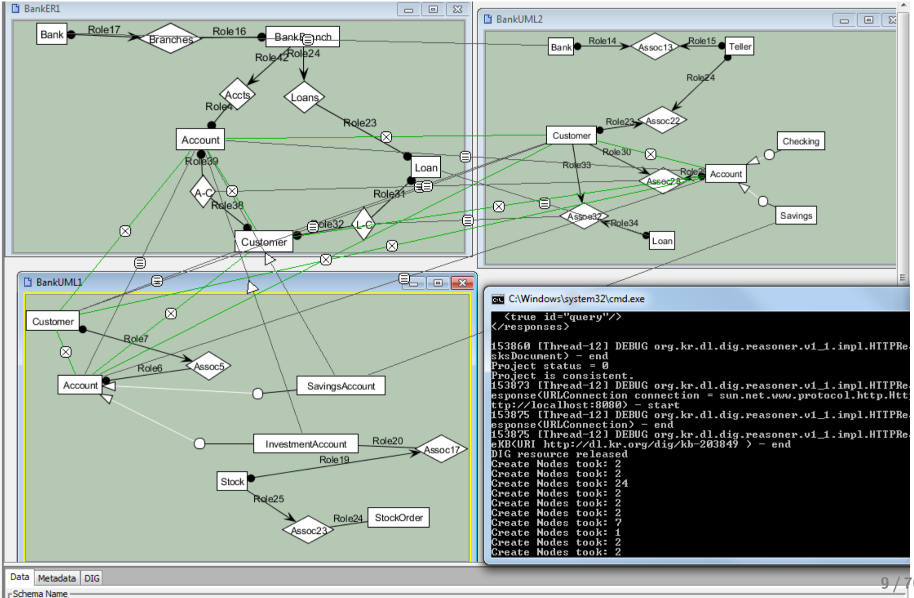
Example scenario: isiZulu termbank (simplified)



After logical and ontological analysis

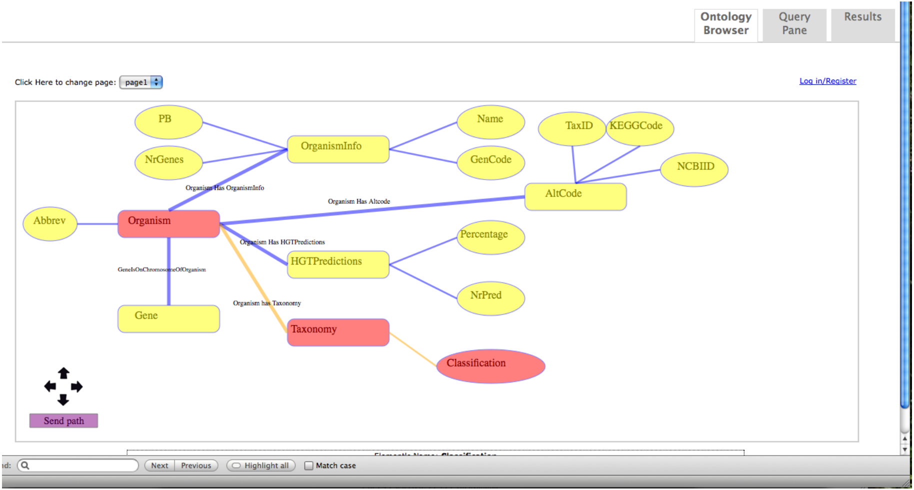


Example: ICOM tool [Fillottrani et al.(2012)]



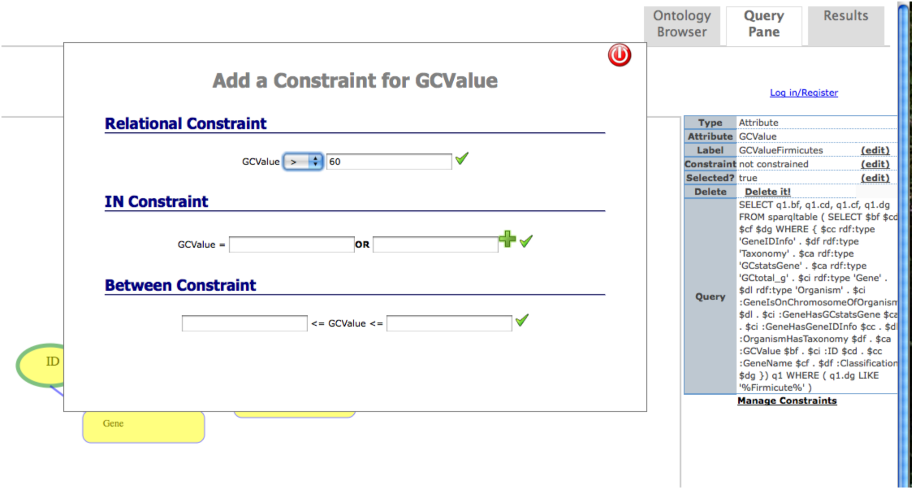
Graphical queries for Ontology-Based Data Access

[Calvanese et al.(2010)]



Graphical queries for Ontology-Based Data Access

[Calvanese et al.(2010)]



Typical *computational* usages for conceptual models

- **Reasoning over conceptual models to improve their quality**
  - With logic-based reconstructions in, and reasoners for, among others: DL [Artale et al.(2007), Berardi et al.(2005), Keet(2009)]) and OWL [Wagih et al.(2013)], OCL [Queralt et al.(2012)], CLIF [Pan and Liu(2010)], Alloy [Braga et al.(2010)], Z [Jahangard Rafsanjani and Mirian-Hosseinabadi(2011)] (and many more)
- **Use of conceptual models during runtime**
  - Verification and validation [Cabot et al.(2008), Nizol et al.(2014)] (e.g., scalable test data generation [Smaragdakis et al.(2009)])
  - Designing [Bloesch and Halpin(1997)] and executing [Calvanese et al.(2010)] queries with the model's vocabulary; VQF/QBD [Soylu et al.(2017)]
  - Querying databases during the stage of query compilation [Toman and Weddell(2011)]
- Ontology-based data access and integration (tries both)

Some common questions/problems/assumptions

- Diagrams are notational variants?
- Design a unifying logic-based reconstruction for one/all CDMLs
- Which CDML features are actually used?
- Decidable, scalable, computationally well-behaved reasoning, querying, ...

$\Rightarrow$ Suitable formalism, logic-based reconstruction

Some common questions/problems/assumptions

- Diagrams are notational variants?
  - They're not
- Design a unifying logic-based reconstruction for one/all CDMLs
- Which CDML features are actually used?
- Decidable, scalable, computationally well-behaved reasoning, querying, ...

$\Rightarrow$ Suitable formalism, logic-based reconstruction

Some common questions/problems/assumptions

- Diagrams are notational variants?
  - They're not
- Design a unifying logic-based reconstruction for one/all CDMLs
  - Achieved for fragments only
- Which CDML features are actually used?
- Decidable, scalable, computationally well-behaved reasoning, querying, ...

$\Rightarrow$ Suitable formalism, logic-based reconstruction

Some common questions/problems/assumptions

- Diagrams are notational variants?
  - They're not
- Design a unifying logic-based reconstruction for one/all CDMLs
  - Achieved for fragments only
- Which CDML features are actually used?
  - Most, but few often
- Decidable, scalable, computationally well-behaved reasoning, querying, ...

$\Rightarrow$ Suitable formalism, logic-based reconstruction

Some common questions/problems/assumptions

- Diagrams are notational variants?
  - They're not
- Design a unifying logic-based reconstruction for one/all CDMLs
  - Achieved for fragments only
- Which CDML features are actually used?
  - Most, but few often
- Decidable, scalable, computationally well-behaved reasoning, querying, ...
  - Many logics used, for the bits and pieces of the CDML that fits that logic

⇒ Suitable formalism, logic-based reconstruction

Some common questions/problems/assumptions

- Diagrams are notational variants?
  - They're not
- Design a unifying logic-based reconstruction for one/all CDMLs
  - Achieved for fragments only
- Which CDML features are actually used?
  - Most, but few often
- Decidable, scalable, computationally well-behaved reasoning, querying, ...
  - Many logics used, for the bits and pieces of the CDML that fits that logic

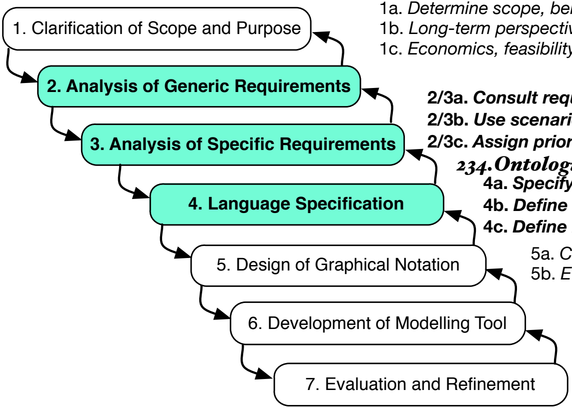
⇒ Suitable formalism, logic-based reconstruction

- 'Suitable'? For what?
- What does a 'good' (or at least not 'bad') formalisation (of CDMLs) look like?

Outline

Guidance on language design

- A logic/language can be seen as a ‘product’/solution that solves a problem
- In analogy of other products: is there a development process, with requirements to meet etc.?
- No methodology for design of a logic
- There is one for design of Domain Specific languages (DSLs) [Frank(2013)]
- Adapt that for our purpose



1a. *Determine scope, benefits*
1b. *Long-term perspective*
1c. *Economics, feasibility*

2/3a. *Consult requirements catalogue*
2/3b. *Use scenarios*
2/3c. *Assign priorities*

234.Ontological analysis of language features

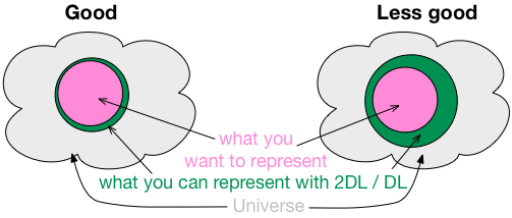
4a. *Specify syntax and semantics*
4b. *Define glossary*
4c. *Define metamodel*

5a. *Create sample diagrams*
5b. *Evaluate notation*

7a. *Test cases*
7b. *Analyse against requirements*
7c. *Analyse effect of use against current practice*

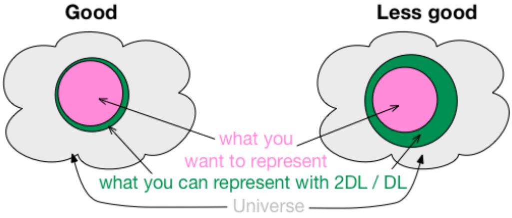
"234. Ontological analysis of language features"

- Affordances and features of the logic concern:
  - Ability to represent the conceptualisation/reality more or less precisely with more or less constraints; e.g.
    - $Human \sqsubseteq \exists hasPart.Eye$ or $Human \sqsubseteq = 2\ hasPart$ (OWL DL)
    - $Human \sqsubseteq = 2\ hasPart.Eye$ (OWL 2 DL)



"234. Ontological analysis of language features"

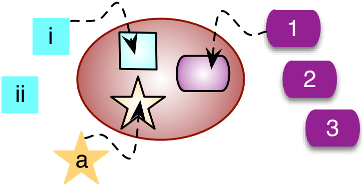
- Affordances and features of the logic concern:
  - Ability to represent the conceptualisation/reality more or less precisely with more or less constraints; e.g.
    - $Human \sqsubseteq \exists hasPart.Eye$ or $Human \sqsubseteq = 2\ hasPart$ (OWL DL)
    - $Human \sqsubseteq = 2\ hasPart.Eye$ (OWL 2 DL)



  - whether the language contributes to support, or even shape, the conceptualisation and one's data analysis for the conceptual data model, or embeds certain philosophical assumptions and positions

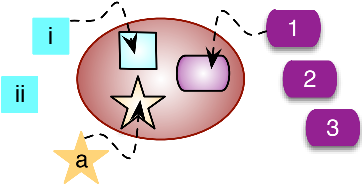
Choices – ontology

- Whether the roles that objects play are fundamental components of relationships (positionalist) or not (standard view); i.e.: if roles should be elements of the language; e.g.
  - $\exists teaches \sqsubseteq Course$ and $\exists teaches^{-} \sqsubseteq Prof$ (most DLs, FOL)
  - $teach \sqsubseteq [lect]Prof \sqcap [taught]Course$ ($\mathcal{DLR}$ family, DBs)



Choices – ontology

- Whether the roles that objects play are fundamental components of relationships (positionalist) or not (standard view); i.e.: if roles should be elements of the language; e.g.
  - $\exists teaches \sqsubseteq Course$ and $\exists teaches^{-} \sqsubseteq Prof$ (most DLs, FOL)
  - $teach \sqsubseteq [lect]Prof \sqcap [taught]Course$ ($\mathcal{DLR}$ family, DBs)



- 4D view on the world (space-time worms) or 3D objects with optional temporal extension
- Inherent vagueness (rough, fuzzy), or the world is crisp

Choices – (im)precision in elements

- Whether refinements on the kinds of general elements—that then have their own representation element—would result in a different (better) conceptual model. e.g.:
  - Add element for aggregation or parthood (in addition to not just Relationship and subsumption)
  - not just Object type but also, say, sortal with rigid property ($\forall x \phi(x) \rightarrow \Box \phi(x)$) or class with anti-rigid property ($\forall x \phi(x) \rightarrow \neg \Box \phi(x)$), with stereotypes or separate graphical elements
  - If binary relationships only (cf. *n*-aries), would the modeller would assume there are only binaries in the world?

Choices – (im)precision in elements

- Whether refinements on the kinds of general elements—that then have their own representation element—would result in a different (better) conceptual model. e.g.:
  - Add element for aggregation or parthood (in addition to not just Relationship and subsumption)
  - not just Object type but also, say, sortal with rigid property ($\forall x \phi(x) \rightarrow \Box \phi(x)$) or class with anti-rigid property ($\forall x \phi(x) \rightarrow \neg \Box \phi(x)$), with stereotypes or separate graphical elements
  - If binary relationships only (cf. *n*-aries), would the modeller would assume there are only binaries in the world?
- 'truly conceptual' or or also somewhat computational; i.e., to represent only *what* vs. *what & how*
  - data types of attributes (UML) or not (ER), with attribute being $A \mapsto C \times \texttt{Datatype}$

The choices in UML, ER, ORM

- Ontology: positionalist, 3D, crisp world
- Features: *n*-aries, UML with aggregation, just object types, ER no datatypes
- Data showed that UML has disproportionally
  - fewer *n*-aries (look across is ambiguous)
  - more aggregation (if the construct is there, modellers see it everywhere?)

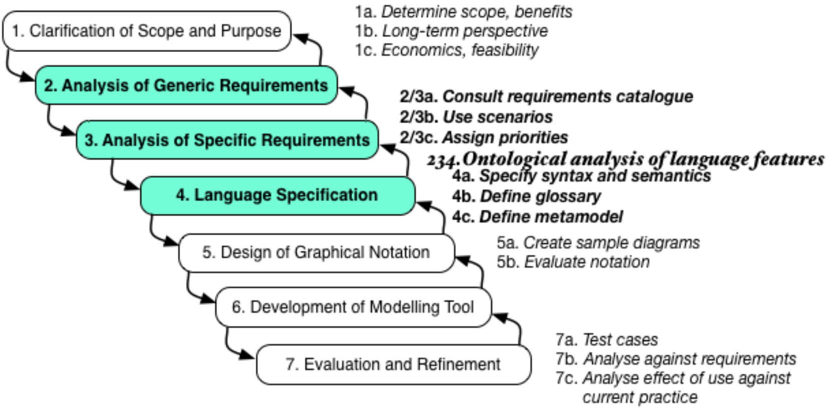
Table: Popular logics for logic-based reconstructions of CDMLs assessed against a set of requirements (1/2).

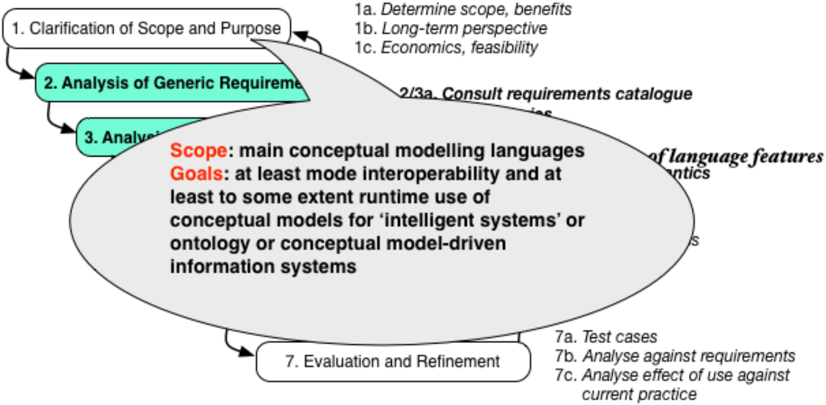
DL-Lite$_{\mathcal{A}}$	$\mathcal{DLR}_{ifd}$	**OWL 2 DL**	**FOL**
Language features			
– standard view	+ positionalist	– standard view	– standard view
– with datatypes	– with datatypes	– with datatypes	+ no datatypes
– no parthood primitive	– no parthood primitive	– no parthood primitive	– no parthood primitive
– no *n*-aries	+ with *n*-aries	– no *n*-aries	+ with *n*-aries
+ 3D	+ 3D	+ 3D	+ 3D
– very few features; large feature mismatch	+ little feature mismatch	± some feature mismatch, with overlapping sets	+ little feature mismatch
– formalisation to complete	+ formalisation exist	– formalisation to complete	± formalisation exist

Table: Popular logics for logic-based reconstructions of CDMLs assessed against a set of requirements (2/2).

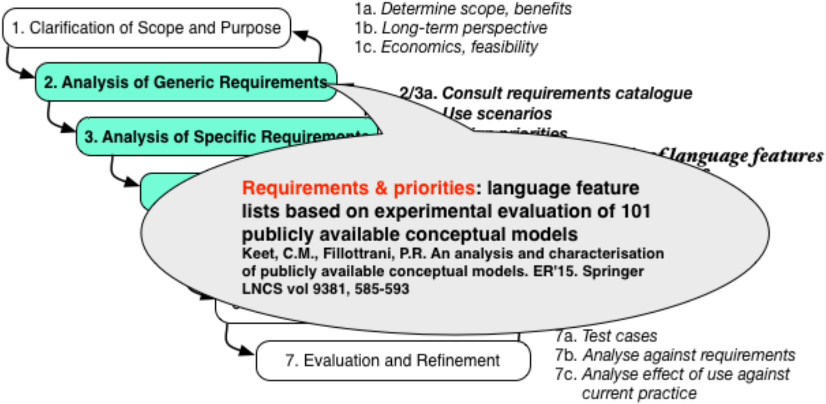
DL-Lite$_{\mathcal{A}}$	$\mathcal{DLR}_{\mathit{ifd}}$	**OWL 2 DL**	**FOL**
Computation and implementability			
+ PTIME (TBox); AC^0 (ABox)	± EXPTIME-complete	± N2EXPTIME-complete	– undecidable
+ very scalable (TBox and ABox)	± somewhat scalable (TBox)	± somewhat scalable (TBox)	– not scalable
+ several reasoners	– no implementation	+ several reasoners	– few reasoners
+ linking with ontologies doable	– no interoperability	+ linking with ontologies doable	– no interoperability with existing infrastructures
+ 'integration' with OntoIOP	– no integration with OntoIOP	+ 'integration' with OntoIOP	+ 'integration' with OntoIOP
+ modularity infrastructure	– modularity infrastructure	+ modularity infrastructure	– modularity infrastructure

Outline


1. Clarification of Scope and Purpose
2. Analysis of Generic Requirements
3. Analysis of Specific Requirements
4. Language Specification
5. Design of Graphical Notation
6. Development of Modelling Tool
7. Evaluation and Refinement
1a. Determine scope, benefits
1b. Long-term perspective
1c. Economics, feasibility
2/3a. Consult requirements catalogue
2/3b. Use scenarios
2/3c. Assign priorities
234. Ontological analysis of language features
4a. Specify syntax and semantics
4b. Define glossary
4c. Define metamodel
5a. Create sample diagrams
5b. Evaluate notation
7a. Test cases
7b. Analyse against requirements
7c. Analyse effect of use against current practice

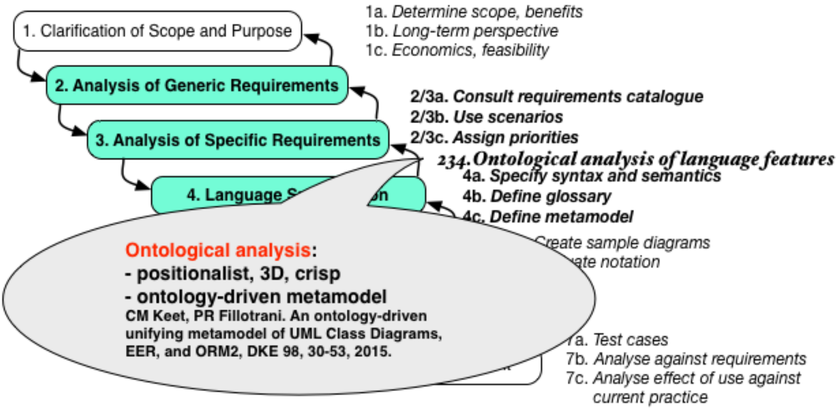


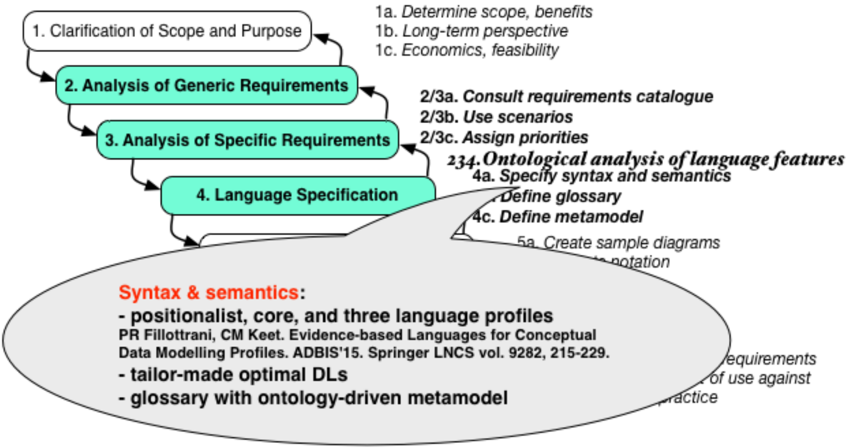
Scope: main conceptual modelling languages
Goals: at least mode interoperability and at least to some extent runtime use of conceptual models for 'intelligent systems' or ontology or conceptual model-driven information systems



Requirements & priorities: language feature lists based on experimental evaluation of 101 publicly available conceptual models

Keet, C.M., Fillottrani, P.R. An analysis and characterisation of publicly available conceptual models. ER'15. Springer LNCS vol 9381, 585-593


1. Clarification of Scope and Purpose
1a. Determine scope, benefits
1b. Long-term perspective
1c. Economics, feasibility
2. Analysis of Generic Requirements
2/3a. Consult requirements catalogue
2/3b. Use scenarios
2/3c. Assign priorities
3. Analysis of Specific Requirements
234. Ontological analysis of language features
4a. Specify syntax and semantics
4b. Define glossary
4c. Define metamodel
7a. Test cases
7b. Analyse against requirements
7c. Analyse effect of use against current practice
Ontological analysis:
- positionalist, 3D, crisp
- ontology-driven metamodel
CM Keet, PR Fillotrani. An ontology-driven unifying metamodel of UML Class Diagrams, EER, and ORM2, DKE 98, 30-53, 2015.


1. Clarification of Scope and Purpose
1a. Determine scope, benefits
1b. Long-term perspective
1c. Economics, feasibility
2. Analysis of Generic Requirements
2/3a. Consult requirements catalogue
2/3b. Use scenarios
2/3c. Assign priorities
3. Analysis of Specific Requirements
234. Ontological analysis of language features
4a. Specify syntax and semantics
Define glossary
4c. Define metamodel
4. Language Specification
5a. Create sample diagrams
notation
equirements
of use against
actice
Syntax & semantics:
- positionalist, core, and three language profiles
PR Fillottrani, CM Keet. Evidence-based Languages for Conceptual Data Modelling Profiles. ADBIS'15. Springer LNCS vol. 9282, 215-229.
- tailor-made optimal DLs
- glossary with ontology-driven metamodel

Logic foundation for profiles

- How to formalise the diagrams in which logic?

⇒ *Which DL (or other logic) is most appropriate, and why?*

⇒ Analyse contents of publicly available conceptual data models[2]

- Try as high a coverage of the most used features

[2] Fillottrani, P.R., Keet, C.M. Evidence-based Languages for Conceptual Data Modelling Profiles. ADBIS'15. Morzy et al. (Eds.). Springer LNCS vol. 9282, 215-229.

Considerations in the formalisation

- Positionalist relations and relationships complicates formalisation (computationally more costly), and implementation ($\mathcal{DLR}$ has one very much proof-of-concept implementation [Calvanese et al.(2011)])
- Did both positionalist and standard core, with algorithm

Definition (Positionalist core profile)

Given a conceptual model in any of the analysed CDMLs, we construct a *knowledge base* in $\mathcal{DC}_p$ by applying the rules:

- we take the set all of object types A, binary relationships P, datatypes T and attributes a in the model as the basic elements in the knowledge base.
- for each binary relationship P formed by object types A and B, we add to the knowledge base the assertions $\geq 1[1]P \sqsubseteq A$ and $\geq 1[2]P \sqsubseteq B$.
- for each attribute a of datatype T within an object type A, including the transformation of ORM's Value Type following the rule given in [Fillottrani and Keet(2014)], we add the assertion $A \sqsubseteq \exists a.T \sqcap \leq 1a$.
- subsumption between two object types A and B is represented by the assertion $A \sqsubseteq B$.

Continues on next slide....

Definition (Positionalist core profile)

Given a conceptual model in any of the analysed CDMLs, we construct a *knowledge base* in $\mathcal{DC}_p$ by applying the rules:

… *continued from previous slide*

- for each object type cardinality $m..n$ in relationship P with respect to its i-th component A, we add the assertions $A \sqsubseteq \leq n[i]P \sqcap \geq m[i]P$.
- we add for each mandatory constraints of a concept A in a relationship P the axiom $A \sqsubseteq \geq 1[1]P$ or $A \sqsubseteq \geq 1[2]P$ depending on the position played by A in P. This is a special case of the previous one, with $n = 1$.
- for each single identification in object type A with respect to an attribute a of datatype T we add the axiom id $A\, a$.

Positionalist Core profile in DL syntax

$\mathcal{DC}_p$ can be represented by the following DL syntax. Starting from atomic elements, we can construct binary relations R, arbitrary concepts C and axioms X according to the rules:

$$C \longrightarrow \top \,|\, A \,|\, \leq k[i]R \,|\, \geq k[i]R \,|\, \forall a.T \,|\, \exists a.T \,|\, \leq 1\, a \,|\, C \sqcap D$$
$$R \longrightarrow \top_2 \,|\, P \,|\, (i : C)$$
$$X \longrightarrow C \sqsubseteq D \,|\, \mathrm{id}\; C\; a$$

where $i = 1, 2$ and $0 < k$. For convenience of presentation, we use the numbers 1 and 2 to name the role places, but they can be any number or string and do not impose an order.

Positionalist Core profile in DL, semantics (1/2)

Definition

An $\mathcal{DC}_p$ *interpretation* $\mathcal{I} = (\cdot^{\mathcal{I}}_{\mathcal{C}}, \cdot^{\mathcal{I}}_{\mathcal{T}}, \cdot^{\mathcal{I}})$ for a knowledge base in $\mathcal{DC}_p$ consists of a set of objects $\Delta^{\mathcal{I}}_{C}$, a set of datatype values $\Delta^{\mathcal{I}}_{T}$, and a function $\cdot^{\mathcal{I}}$ satisfying the constraints shown in Table 3. It is said that $\mathcal{I}$ *satisfies* the assertion $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; and it *satisfies* the assertion id C a iff exists T such that $C^{\mathcal{I}} \subseteq (\exists a.T \sqcap \leq 1a)^{\mathcal{I}}$ (mandatory 1) and for all $v \in T^{\mathcal{I}}$ it holds that $\#\{c | c \in C^{\mathcal{I}} \wedge (c, v) \in a^{\mathcal{I}}\} \leq 1$ (inverse functional).

Positionalist Core profile in DL, semantics (2/2)

Table: Semantics of $\mathcal{DC}_p$.

$$\top^{\mathcal{I}} \subseteq \Delta_C^{\mathcal{I}}$$
$$A^{\mathcal{I}} \subseteq \top^{\mathcal{I}}$$
$$\top_2^{\mathcal{I}} = \top^{\mathcal{I}} \times \top^{\mathcal{I}}$$
$$P^{\mathcal{I}} \subseteq \top_2^{\mathcal{I}}$$
$$T^{\mathcal{I}} \subseteq \Delta_T^{\mathcal{I}}$$
$$a^{\mathcal{I}} \subseteq \top^{\mathcal{I}} \times \Delta_T^{\mathcal{I}}$$
$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$

$$(\leq k[i]R)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} | \#\{(d_1, d_2) \in R^{\mathcal{I}}.d_i = c\} \leq k\}$$
$$(\geq k[i]R)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} | \#\{d_1, d_2) \in R^{\mathcal{I}}.d_i = c\} \geq k\}$$
$$(\exists a.T)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} | \exists b \in T^{\mathcal{I}}.(c, b) \in a^{\mathcal{I}}\}$$
$$(\forall a.T)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} | \forall v \in \Delta_T^{\mathcal{I}}.(c, v) \in a^{\mathcal{I}} \rightarrow v \in T^{\mathcal{I}}\}$$
$$(\leq 1\, a)^{\mathcal{I}} = \{c \in \Delta_C^{\mathcal{I}} | \#\{(c, v) \in a^{\mathcal{I}}\} \leq 1\}$$
$$(i : C)^{\mathcal{I}} = \{(d_1, d_2) \in \top_2^{\mathcal{I}} | d_i \in C^{\mathcal{I}}\}$$

Some observations

- All the entities in the core profile sum up to 87.57% of the entities in all the analysed models, covering 91,88% of UML models, 73.29% of ORM models, and 94.64% of ER/EER models
- Excluded due to their low incidence in the model set (despite overlap): Role (DL role component) and Relationship (DL role) Subsumption, and Completeness and Disjointness constraints

Some observations

- All the entities in the core profile sum up to 87.57% of the entities in all the analysed models, covering 91,88% of UML models, 73.29% of ORM models, and 94.64% of ER/EER models
- Excluded due to their low incidence in the model set (despite overlap): Role (DL role component) and Relationship (DL role) Subsumption, and Completeness and Disjointness constraints
- No completeness and disjointness, so reasoning is quite simple
- Can code negation only with cardinality constraints [Baader et al.(2008), chapter 3], but then we need to reify each negated concept as a new idempotent role, which is not possible to get from the $\mathcal{DC}_p$ rules
- Can embed $\mathcal{DC}_p$ into $\mathcal{DLR}$, but latter is more expressive than needed

Standard core profile

- Convert $\mathcal{DC}_p$ into a standard core, $\mathcal{DC}_s$

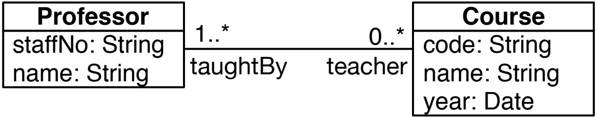
Definition

Given a conceptual model in any of the analysed CDMLs, we construct a *knowledge based* in $\mathcal{DC}_s$ by applying Algorithm 1 to its $\mathcal{DC}_p$ knowledge base.

- With inverse relations to keep connected both relationships generated by reifying roles
- DL syntax approximation (noting construction rules from $\mathcal{DC}_p$):

$$C \longrightarrow \top_1 \,|\, A \,|\, \forall R.A \,|\, \exists R.A \,|\, \leq k\, R \,|\, \geq k\, R \,|\, \forall a.T \,|\, \exists a.T \,|\, \leq 1\, a.T \,|\, C \sqcap D$$
$$R \longrightarrow \top_2 \,|\, P \,|\, P^-$$
$$X \longrightarrow C \sqsubseteq D \,|\, \mathrm{id}\; C\; a$$

Positionalist to standard choices



teacher and *taughtBy* are named association ends, not a name of the association (DL role). Options to formalise it:

- make each association end a DL role, `teacher` and `taughtBy`, then choose:
  - declare them inverse of each other with $\texttt{teacher} \equiv \texttt{taughtBy}^-$
  - do not declare them inverses
- 'bump up' either *teacher* or *taughtBy* to DL role, and use the other through a direct inverse and do not extend vocabulary with the other ($\texttt{teacher}$ and $\texttt{teacher}^-$ cf. adding also `taughtBy`)

Algorithm 1 *Positionalist Core to Standard Core*

P an atomic binary relationship; D_P domain of P; R_P range of P
if $D_P \neq R_P$ **then**
  Rename P to two 'directional' readings, Pe_1 and Pe_2
  Make Pe_1 and Pe_2 a DL relation (role)
  Type the relations with $\top \sqsubseteq \forall Pe_1.D_P \sqcap \forall Pe_1^-.R_P$
  Declare inverses with $Pe_1 \equiv Pe_2^-$
else
  if $D_P = R_P$ **then**
    if $i = 1, 2$ is named **then**
      $Pe_i \leftarrow i$
    else
      $Pe_i \leftarrow$ user-added label or auto generated label
    end if
    Make Pe_i a DL relation (role)
    Type one Pe_i, i.e., $\top \sqsubseteq \forall Pe_i.D_P \sqcap \forall Pe_i^-.R_P$
    Declare inverses with $Pe_i \equiv Pe_2^-$
  end if
end if

Some observations on $\mathcal{DC}_s$

- Simple, too
- Main reasoning problem still class subsumption and equivalence
- At most the DL $\mathcal{ALNI}$ (called $\mathcal{PL}_1$ in [Donini et al.(1991)])
- $\mathcal{PL}_1$ has polynomial subsumption; data complexity unknown
- Tweaking with interaction between role inclusions and number restrictions, and UNA: $DL\text{-}Lite_{core}^{(\mathcal{HN})}$ (NLOGSPACE)
- As aside: adding class disjointness, then reduction to $DL\text{-}Lite_{bool}^{(\mathcal{HN})}$ (NP-hard) [Artale et al.(2009)]

Profile for UML Class diagrams (1/2)

- Strictly extends $\mathcal{DC}_s$ with:
  - **Shared & composite** aggregate (no new semantics in UML v2.4.1)
  - Association **subsumption**, as DL role inclusion $R \sqsubseteq S$
  - **Attributive Property Cardinality** (as for DL roles) and Attribute **Value Constraint** (define new data types)

Definition

A *knowledge base* in $\mathcal{DC}_{UML}$ from a given conceptual model in UML is obtained by adding to its $\mathcal{DC}_s$ knowledge base the following formulas and axioms:

- for each attribute cardinality $m..n$ in an attribute a of datatype T within an object type A we add the assertion $A \sqsubseteq \leq n\, a.T \sqcap \geq m\, a.T$.
- for each binary relationship subsumption between relationships R and S we add the axiom $R \sqsubseteq S$.

The syntax is as in $\mathcal{DC}_s$, with the additions highlighted in bold face for easy comparison:

$$
\begin{aligned}
C &\longrightarrow \top \mid A \mid \forall R.A \mid \exists R.A \mid \leq k\, R \mid \geq k\, R \mid \forall a.T \mid \exists a.T \\
C &\longrightarrow \leq \mathbf{k}\, a.T \mid \geq \mathbf{k\, a.T} \mid C \sqcap D \\
R &\longrightarrow \top_2 \mid P \mid P^- \\
X &\longrightarrow C \sqsubseteq D \mid \mathbf{R} \sqsubseteq \mathbf{S} \mid \mathrm{id}\; C\; a
\end{aligned}
$$

Definition

A $\mathcal{DC}_{UML}$ *interpretation* for a $\mathcal{DC}_{UML}$ knowledge base is a $\mathcal{DC}_s$ interpretation $\mathcal{I}$ that also satisfies $R \sqsubseteq S$ if and only if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$, with $(\leq k\, a.T)^{\mathcal{I}} = \{c \in \Delta^{\mathcal{I}}_C \mid \#\{a \in T^{\mathcal{I}} \mid (c, a) \in a^{\mathcal{I}}\} \leq k\}$ and $(\geq k\, a.T)^{\mathcal{I}} = \{c \in \Delta^{\mathcal{I}}_C \mid \#\{a \in T^{\mathcal{I}} \mid (c, a) \in a^{\mathcal{I}}\} \geq k\}$.

Profile for UML Class diagrams

- 99.44% of all the elements in the analysed UML models are covered by this profile.
- $\mathcal{ALNI}$ + role hierarchies $\rightarrow$ $\mathcal{ALNHI}(\mathcal{D})$ that has not been studied yet
- If with UNA and some restrictions on role inclusion and cardinalities, then $DL\text{-}Lite_{core}^{(\mathcal{H})\mathcal{N}}$ suffices (NLOGSPACE for subsumption and AC^0 for data complexity [Artale et al.(2009)])

Profile for ER and EER

- Core Profile plus:
  - **Composite** and **Multivalued attribute**: multivalued with attribute cardinality, composite with union datatype derivation operator
  - **Weak Object Type**, **Weak Identification**: use functionality constraints on roles as in $\mathcal{DLR}_{ifd}$ [Calvanese et al.(2001)] or in $\mathcal{CFD}$ [Toman and Weddell(2009)]
  - **Ternary relationships**
  - **Associative Object type**: use reification
  - **Multiattribute identification**: new composite attribute with single identification
- 99.06% of all the elements in the set of (E)ER models belong to this profile

Which DL language, complexity?

- The only DL with arbitrary *n*-aries and the advanced id constraints is $\mathcal{DLR}_{ifd}$, which is positionalist
- DL role components are not strictly needed for (E)ER; *n*-ary DL without DL role components but with id: the $\mathcal{CFD}$ family + positionalist to standard view Algorithm 1
- Giving in a little more
  - (binaries and with UNA): $DL\text{-}Lite^{\mathcal{N}}_{core}$ [Artale et al.(2009)] (NLOGSPACE for the satisfiability problem), use Algo 2
  - (no composite att, no weak entity types): similar result for ER_{ref} [Artale et al.(2007)]

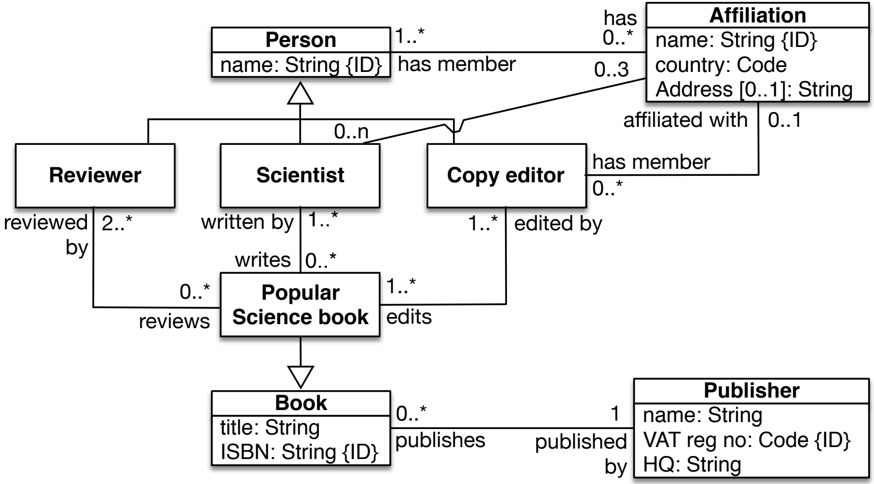
Algorithm 2 *Equivalence-preserving n-ary into a binary conversion*

D_R: domain of R; R_R range of R; n set of R-components
Reify R into $R' \sqsubseteq \top$
for all i, $3 \geq i \geq n$ **do**
  $Re_i \leftarrow$ user-added label or auto generated label
  Make Re_i a DL role,
  Type Re_i as $\top \sqsubseteq \forall Re_i.R' \sqcap \forall Re_i^-.R_R$, where R_R is the player ((E)ER entity type) in n
  Add $R' \sqsubseteq \exists Re_i.\top$ and $R' \sqsubseteq \leq 1\, Re_i.\top$
end for
Add external identifier $\top \sqsubseteq \leq 1\, (\sqcup_i Re_i)^-.R'$

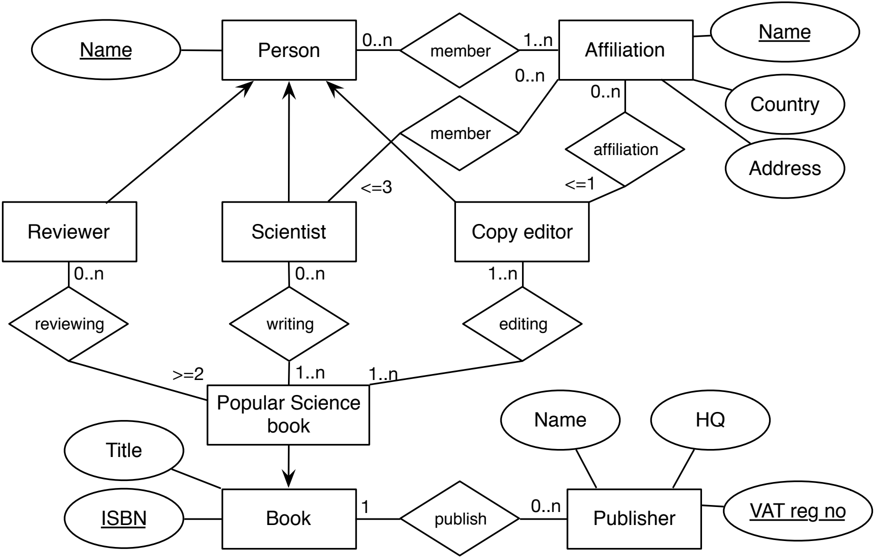
ORM and ORM2

- Profile covers 98.69% of all the elements in the analysed ORM an ORM2 models
- Main required extras: *n*-aries, identification constraints, and the **argument positions**
- Some extra processing due to fact type readings cf naming roles
- Then complexity of the ORM/2 Profile is still not clear:
  - EXPTIME-complete $\mathcal{DLR}_{ifd}$ is the easiest fit, but contains more than is strictly needed (disj., compl., complex fd)
  - PTIME $\mathcal{CFDI}_{nc}^{\forall-}$ [Toman and Weddell(2014)] may be a better candidate [Fillottrani et al.(2015)] provided positionalist to standard view translation; but no arbitrary number restrictions, disjunctive mandatory on ORM roles (so 96.5% coverage)

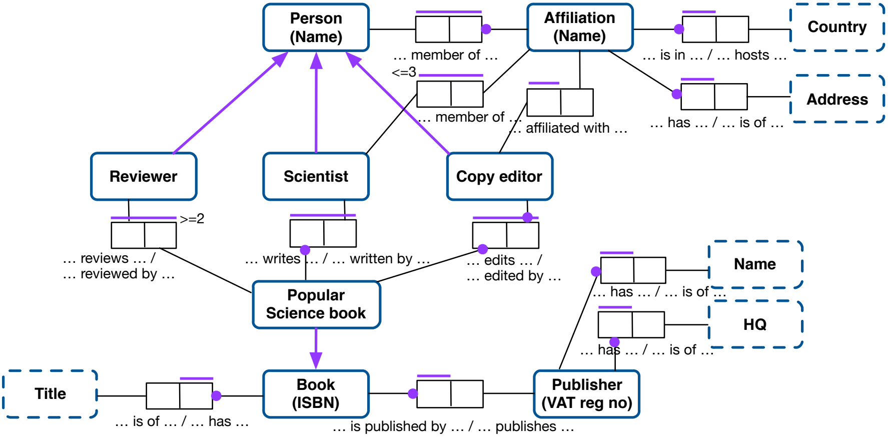
Sample diagrams using all $\mathcal{DC}_s$ features



Sample diagrams using all $\mathcal{DC}_s$ features



Sample diagrams using all $\mathcal{DC}_s$ features



Or as business rules (fragment shown)

- Each popular science book is reviewed by at least 2 reviewers.
- Each reviewer may review a popular science book.
- Each book must be published by exactly one publisher.
- Each publisher has one HQ.

Steps UML diagram to $\mathcal{DC}_s$

- (Recall $\mathcal{DC}_s$ is obtained from $\mathcal{DC}_p+$ Algorithm 1)
- Obtain set of OTs ({`Person, ...`}) `and` DTs ({`Name, ...`})

Steps UML diagram to $\mathcal{DC}_s$

- (Recall $\mathcal{DC}_s$ is obtained from $\mathcal{DC}_p$+ Algorithm 1)
- Obtain set of OTs ({Person, ...}) and DTs ({Name, ...})
- For Relationships, use Algorithm 1:
  1. bump up the association end names to DL roles
  2. type the relationships with:

$$\top \sqsubseteq \forall \texttt{has\_member}.\texttt{Affiliation} \sqcap \forall \texttt{has\_member}^{-}.\texttt{Person}$$

$$\top \sqsubseteq \forall \texttt{has}.\texttt{Person} \sqcap \forall \texttt{has}^{-}.\texttt{Affiliation}$$

  3. declare inverses, $\texttt{has\_member} \equiv \texttt{has}^{-}$

Repeat for each association in UML diagram

Steps UML diagram to $\mathcal{DC}_s$

- (Recall $\mathcal{DC}_s$ is obtained from $\mathcal{DC}_p$+ Algorithm 1)
- Obtain set of OTs ({`Person, ...`}) and DTs ({`Name, ...`})
- For Relationships, use Algorithm 1:
  1. bump up the association end names to DL roles
  2. type the relationships with:

$$\top \sqsubseteq \forall \texttt{has\_member}.\texttt{Affiliation} \sqcap \forall \texttt{has\_member}^{-}.\texttt{Person}$$

$$\top \sqsubseteq \forall \texttt{has}.\texttt{Person} \sqcap \forall \texttt{has}^{-}.\texttt{Affiliation}$$

  3. declare inverses, $\texttt{has\_member} \equiv \texttt{has}^{-}$

  Repeat for each association in UML diagram
- Step 3 of $\mathcal{DC}_p$ definition: attributes. e.g., for `Person`'s `Name`:

$$\texttt{Person} \sqsubseteq \exists \texttt{Name}.\texttt{String} \sqcap \leq 1\ \texttt{Name}$$

Steps UML diagram to $\mathcal{DC}_s$

- (Recall $\mathcal{DC}_s$ is obtained from $\mathcal{DC}_p$+ Algorithm 1)
- Obtain set of OTs ({Person, ...}) and DTs ({Name, ...})
- For Relationships, use Algorithm 1:
  1. bump up the association end names to DL roles
  2. type the relationships with:

$$\top \sqsubseteq \forall \mathtt{has\_member}.\mathtt{Affiliation} \sqcap \forall \mathtt{has\_member}^{-}.\mathtt{Person}$$

$$\top \sqsubseteq \forall \mathtt{has}.\mathtt{Person} \sqcap \forall \mathtt{has}^{-}.\mathtt{Affiliation}$$

  3. declare inverses, $\mathtt{has\_member} \equiv \mathtt{has}^{-}$

  Repeat for each association in UML diagram
- Step 3 of $\mathcal{DC}_p$ definition: attributes. e.g., for Person's Name:

$$\mathtt{Person} \sqsubseteq \exists \mathtt{Name}.\mathtt{String} \sqcap \leq 1 \ \mathtt{Name}$$

- Step 4: subsumptions; e.g., $\mathtt{Popular\_science\_book} \sqsubseteq \mathtt{Book}$

Steps UML diagram to $\mathcal{DC}_s$

- (Recall $\mathcal{DC}_s$ is obtained from $\mathcal{DC}_p$+ Algorithm 1)
- Obtain set of OTs ({Person, ...}) and DTs ({Name, ...})
- For Relationships, use Algorithm 1:
  1. bump up the association end names to DL roles
  2. type the relationships with:

$$\top \sqsubseteq \forall \texttt{has\_member}.\texttt{Affiliation} \sqcap \forall \texttt{has\_member}^{-}.\texttt{Person}$$

$$\top \sqsubseteq \forall \texttt{has}.\texttt{Person} \sqcap \forall \texttt{has}^{-}.\texttt{Affiliation}$$

  3. declare inverses, $\texttt{has\_member} \equiv \texttt{has}^{-}$

  Repeat for each association in UML diagram
- Step 3 of $\mathcal{DC}_p$ definition: attributes. e.g., for Person's Name:

$$\texttt{Person} \sqsubseteq \exists \texttt{Name}.\texttt{String} \sqcap \leq 1 \ \texttt{Name}$$

- Step 4: subsumptions; e.g., $\texttt{Popular\_science\_book} \sqsubseteq \texttt{Book}$
- Step 5 and 6: cardinalities. e.g. $\texttt{Affiliation} \sqsubseteq \geq 1 \ \texttt{has\_member}$

Steps UML diagram to $\mathcal{DC}_s$

- (Recall $\mathcal{DC}_s$ is obtained from $\mathcal{DC}_p$+ Algorithm 1)
- Obtain set of OTs ({Person, ...}) and DTs ({Name, ...})
- For Relationships, use Algorithm 1:
  1. bump up the association end names to DL roles
  2. type the relationships with:

$$\top \sqsubseteq \forall \texttt{has\_member}.\texttt{Affiliation} \sqcap \forall \texttt{has\_member}^{-}.\texttt{Person}$$

$$\top \sqsubseteq \forall \texttt{has}.\texttt{Person} \sqcap \forall \texttt{has}^{-}.\texttt{Affiliation}$$

  3. declare inverses, $\texttt{has\_member} \equiv \texttt{has}^{-}$

  Repeat for each association in UML diagram
- Step 3 of $\mathcal{DC}_p$ definition: attributes. e.g., for Person's Name:

$$\texttt{Person} \sqsubseteq \exists \texttt{Name}.\texttt{String} \sqcap \leq 1\ \texttt{Name}$$

- Step 4: subsumptions; e.g., $\texttt{Popular\_science\_book} \sqsubseteq \texttt{Book}$
- Step 5 and 6: cardinalities. e.g. $\texttt{Affiliation} \sqsubseteq \geq 1\ \texttt{has\_member}$
- Finally, identifiers; e.g. ISBN for Book, adding id Book ISBN to the $\mathcal{DC}_s$ knowledge base

Outline

Profile comparison on language and complexity

Profile	Main features	Approx. DL	Subsumption complexity
$\mathcal{DC}_p$	positionalist, binary relationships, identifiers, cardinality constraints, attribute typing, mandatory attribute and its functionality	$\mathcal{DLR}$	EXPTIME
$\mathcal{DC}_s$	standard view, binary relationships, inverses	$\mathcal{ALNI}$	P
$\mathcal{DC}_{UML}$	relationship subsumption, attribute cardinality	$DL\text{-}Lite_{core}^{\mathcal{HN}}$	NLOGSPACE
$\mathcal{DC}_{EER}$	ternary relationships, attribute cardinality, external keys	$DL\text{-}Lite_{core}^{N}$	NLOGSPACE
		$\mathcal{CFD}$	P
$\mathcal{DC}_{ORM}$	entity type disjunction, relationships complement, relationship subsumption, complex identifiers ('multi attribute keys')	$\mathcal{DLR}_{ifd}$	EXPTIME
		$\mathcal{CFDI}_{nc}^{\forall-}$	P

Discussion

- ‘Uninteresting’ logics for automated reasoning over conceptual models
- But

Discussion

- 'Uninteresting' logics for automated reasoning over conceptual models
- But
- assuming that also the reconstructions of $\mathcal{DC}_p$ and $\mathcal{DC}_{ORM}$ will be lower than EXPTIME (tbd),
- They're good/excellent for use of conceptual models during runtime; e.g.:

Discussion

- ‘Uninteresting’ logics for automated reasoning over conceptual models
- But
- assuming that also the reconstructions of $\mathcal{DC}_p$ and $\mathcal{DC}_{ORM}$ will be lower than EXPTIME (tbd),
- They’re good/excellent for use of conceptual models during runtime; e.g.:
  - Scalable test data generation [Smaragdakis et al.(2009)]
  - Designing [Bloesch and Halpin(1997)] and executing [Calvanese et al.(2010)] queries with the model’s vocabulary
  - Querying databases during the stage of query compilation [Toman and Weddell(2011)]

Language design

- First attempt to scope and structure the the logic design process, with ontological considerations
- Can do with a broader systematic investigation on alternative design choices and their consequences
- Identified alternate choices effectively addressed by multiple compatible profiles with algorithms for conversions

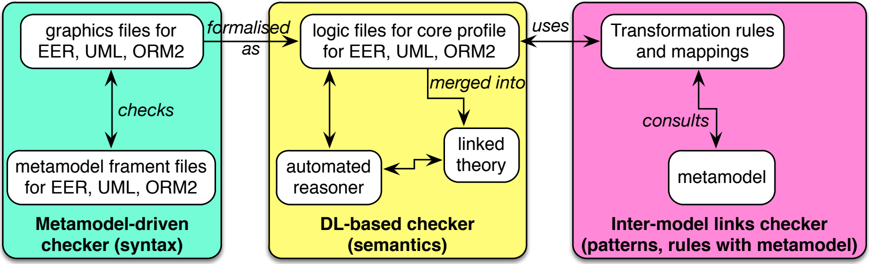
Language design

- First attempt to scope and structure the the logic design process, with ontological considerations
- Can do with a broader systematic investigation on alternative design choices and their consequences
- Identified alternate choices effectively addressed by multiple compatible profiles with algorithms for conversions
- 'good' logic
  - matches the implicit ontological commitments
  - that fits needs here is 'less good' in precision
  - turns out to be a *family* of compatible logics + algorithms

Toward applicability

- Profiles may be applied as back-end of CASE tool, OBDA
- Will allow modeller to model in their graphical notation of choice, yet be compatible with the rest
- Transformations and inter-model assertions of approximate entities and of modelling patterns

  [Fillottrani and Keet(2014), Khan et al.(2016), Fillottrani and Keet(2017)] details



Conclusions

- Ontology-informed language design process
- Used to define five optimal profiles
- Profiles are exceedingly suitable for runtime usage of conceptual models

Ongoing and future work

- Integrate these results into design tools (commenced, in ICOM)
- Module management, modularisation & rules (ongoing)
- Complexity of the profiles
- Suitable reasoner
- ‘Scalability’ of graphical representation and inferences

References I

A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev.
The DL-Lite family and relations.
Journal of Artificial Intelligence Research, 36:1–69, 2009.

Alessandro Artale, Diego Calvanese, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev.
Reasoning over extended ER models.
In Christine Parent, Klaus-Dieter Schewe, Veda C. Storey, and Bernhard Thalheim, editors, *Proceedings of the 26th International Conference on Conceptual Modeling (ER'07)*, volume 4801 of *LNCS*, pages 277–292. Springer, 2007. Auckland, New Zealand, November 5-9, 2007.

F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors.
The Description Logics Handbook – Theory and Applications.
Cambridge University Press, 2 edition, 2008.

D. Berardi, D. Calvanese, and G. De Giacomo.
Reasoning on UML class diagrams.
Artificial Intelligence, 168(1-2):70–118, 2005.

A. C. Bloesch and T. A. Halpin.
Conceptual Queries using ConQuer-II.
In *Proceedings of ER'97: 16th International Conference on Conceptual Modeling*, volume 1331 of *LNCS*, pages 113–126. Springer, 1997.

Bernardo F. B. Braga, João Paulo A. Almeida, Giancarlo Guizzardi, and Alessander Botti Benevides.
Transforming OntoUML into Alloy: towards conceptual model validation using a lightweight formal methods.
Innovations in Systems and Software Engineering, 6(1-2):55–63, 2010.

References II

Jordi Cabot, Robert Clarisó, and Daniel Riera.
Verification of UML/OCL class diagrams using constraint programming.
In *Model Driven Engineering, Verification, and Validation: Integrating Verification and Validation in MDE (MoDeVVA 2008)*, 2008.

Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini.
Identification constraints and functional dependencies in description logics.
In Bernhard Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 155–160. Morgan Kaufmann, 2001.
Seattle, Washington, USA, August 4-10, 2001.

Diego Calvanese, C. Maria Keet, Werner Nutt, Mariano Rodríguez-Muro, and Giorgio Stefanoni.
Web-based graphical querying of databases through an ontology: the WONDER system.
In Sung Y. Shin, Sascha Ossowski, Michael Schumacher, Mathew J. Palakal, and Chih-Cheng Hung, editors, *Proceedings of ACM Symposium on Applied Computing (ACM SAC'10)*, pages 1389–1396. ACM, 2010.
March 22-26 2010, Sierre, Switzerland.

Diego Calvanese, Domenico Carbotta, and Magdalena Ortiz.
A practical automata-based technique for reasoning in expressive description logics.
In *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11)*, 2011.

F. Donini, M. Lenzerini, D. Nardi, and W. Nutt.
Tractable concept languages.
In *Proc.of IJCAI'91*, volume 91, pages 458–463, 1991.

References III

P. R. Fillottrani and C. M. Keet.
Conceptual model interoperability: a metamodel-driven approach.
In A. Bikakis et al., editors, *Proceedings of the 8th International Web Rule Symposium (RuleML'14)*, volume 8620 of *LNCS*, pages 52–66. Springer, 2014.
August 18-20, 2014, Prague, Czech Republic.

Pablo R. Fillottrani and C. Maria Keet.
Patterns for heterogeneous tbox mappings to bridge different modelling decisions.
In E. Blomqvist et al., editors, *Proc. of ESWC'17*, volume 10249 of *LNCS*, pages 371–386. Springer, 2017.
30 May - 1 June 2017, Portoroz, Slovenia.

Pablo R. Fillottrani, Enrico Franconi, and Sergio Tessaris.
The ICOM 3.0 intelligent conceptual modelling tool and methodology.
Semantic Web Journal, 3(3):293–306, 2012.

Pablo Rubén Fillottrani, C. Maria Keet, and David Toman.
Polynomial encoding of orm conceptual models in $\mathcal{CFDI}_{nc}^{\forall -}$.
In D. Calvanese and B. Konev, editors, *Proceedings of the 28th International Workshop on Description Logics (DL'15)*, volume 1350 of *CEUR-WS*, pages 401–414, 2015.
7-10 June 2015, Athens, Greece.

Ulrich Frank.
Domain-specific modeling languages - requirements analysis and design guidelines.
In I. Reinhartz-Berger, A. Sturm, T. Clark, J. Bettin, and S. Cohen, editors, *Domain Engineering: Product Lines, Conceptual Models, and Languages*, pages 133–157. Springer, 2013.

References IV

Amir Jahangard Rafsanjani and Seyed-Hassan Mirian-Hosseinabadi.
A Z Approach to Formalization and Validation of ORM Models.
In Ezendu Ariwa and Eyas El-Qawasmeh, editors, *Digital Enterprise and Information Systems*, volume 194 of *CCIS*, pages 513–526. Springer, 2011.

C. Maria Keet.
Positionalism of relations and its consequences for fact-oriented modelling.
In R. Meersman, P. Herrero, and Dillon T., editors, *OTM Workshops, International Workshop on Fact-Oriented Modeling (ORM'09)*, volume 5872 of *LNCS*, pages 735–744. Springer, 2009.
Vilamoura, Portugal, November 4-6, 2009.

Zubeida C. Khan, C. Maria Keet, Pablo R. Fillottrani, and Karina Cenci.
Experimentally motivated transformations for intermodel links between conceptual models.
In J. Pokorný et al., editors, *20th Conference on Advances in Databases and Information Systems (ADBIS'16)*, volume 9809 of *LNCS*, pages 104–118. Springer, 2016.
August 28-31, Prague, Czech Republic.

Matthew Nizol, Laura K. Dillon, and R. E. K. Stirewalt.
Toward tractable instantiation of conceptual data models using non-semantics-preserving model transformations.
In *Proceedings of the 6th International Workshop on Modeling in Software Engineering (MiSE'14)*, pages 13–18. ACM Conference Proceedings, 2014.
Hyderabad, India, June 02-03, 2014.

Wen-Lin Pan and Da-xin Liu.
Mapping object role modeling into common logic interchange format.
In *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE'10)*, volume 2, pages 104–109. IEEE Computer Society, 2010.

References V

Anna Queralt, Alessandro Artale, Diego Calvanese, and Ernest Teniente.
OCL-Lite: Finite reasoning on UML/OCL conceptual schemas.
Data & Knowledge Engineering, 73:1–22, 2012.

Y. Smaragdakis, C. Csallner, and R. Subramanian.
Scalable satisfiability checking and test data generation from modeling diagrams.
Automation in Software Engineering, 16:73–99, 2009.

Ahmet Soylu, Evgeny Kharlamov, Dimitry Zheleznyakov, Ernesto Jimenez Ruiz, Martin Giese, Martin G. Skjaeveland, Dag Hovland, Rudolf Schlatte, Sebastian Brandt, Hallstein Lie, and Ian Horrocks.
OptiqueVQS: a visual query system over ontologies for industry.
Semantic Web Journal, page in press, 2017.

David Toman and Grant E Weddell.
Applications and extensions of PTIME Description Logics with functional constraints.
In *Proceedings of the 21st International Joint Conference on Artificial Intelligence IJCAI'09*, pages 948–954. AAAI Press, 2009.

David Toman and Grant E. Weddell.
Fundamentals of Physical Design and Query Compilation.
Synthesis Lectures on Data Management. Morgan & Claypool, 2011.

David Toman and Grant E. Weddell.
On adding inverse features to the description logic $\mathit{CFD}^{\forall}{}_{\mathrm{nc}}$.
In *PRICAI 2014: Trends in Artificial Intelligence - 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014.*, pages 587–599, 2014.

References VI

Heba M. Wagih, Doaa S. El Zanfaly, and Mohamed M. Kouta.
Mapping Object Role Modeling 2 schemes into $\mathcal{SROIQ}(d)$ description logic.
International Journal of Computer Theory and Engineering, 5(2):232–237, 2013.

Thank you!

For more information, papers, data sets, presentations and other files, please visit
http://www.meteck.org/SAAR.html

Bilateral project "ontology-driven unification of conceptual data modelling languages" funded by SA Dept. of Sci & Tech and AR's MINCyT

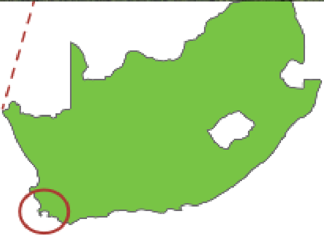
International Conference on Formal Ontology and Information Systems 2018 (FOIS'18)

17-21 September 2018, Cape Town

http://fois2018.cs.uct.ac.za/

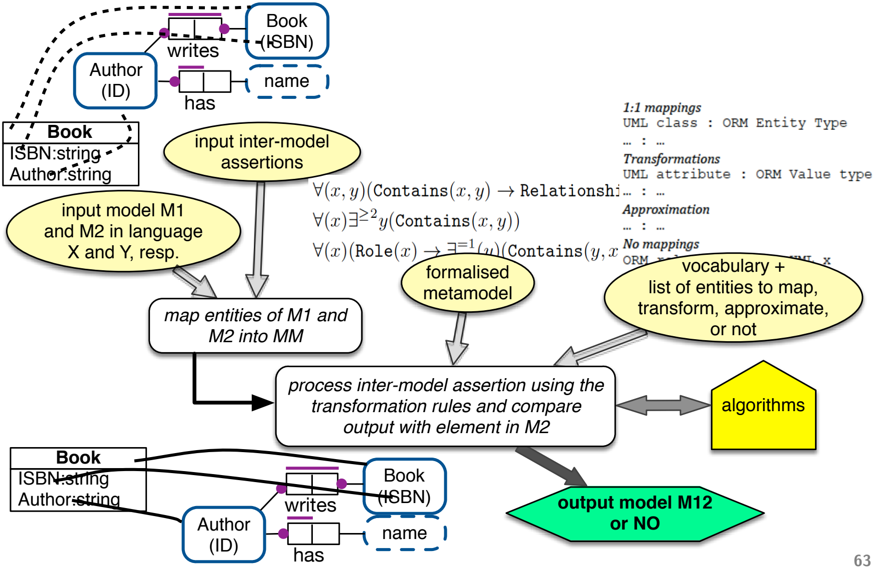
CS@UCT

CS@UCT

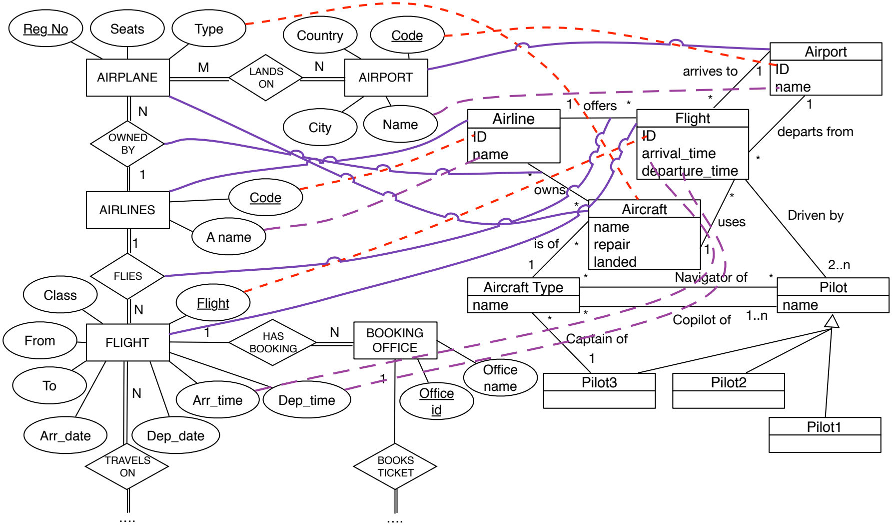


CS@UCT

Inter-model assertions [Fillottrani and Keet(2014)]



(still a small ‘toy’ example)



1:1 mapping rules and the metamodel (selection)

(R1) `Association` $\xrightarrow{\text{UML to MM}}$ Relationship

in: `Association(AssociationEnd : Class, AssociationEnd : Class)`

out: `AssociationEnd` → Role *// i.e., using (Ro1)*

out: `Association` → Relationship

out: `Class` → Object Type *// i.e., using (O1)*

out: Relationship(Role:Object type, Role:Object Type)

(1R) Relationship $\xrightarrow{\text{MM to UML}}$ `Association`

in: Relationship(Role:Object type, Role:Object Type)

out: Role → `AssociationEnd` *// i.e., using (1Ro)*

out: Relationship → `Association`

out: Object Type → `Class` *// i.e., using (1O)*

out:
`Association(AssociationEnd : Class, AssociationEnd : Class)`

Formalised metamodel (section), highlighted for step 2

$\forall(x,y)(\texttt{Contains}(x,y) \rightarrow \texttt{Relationship}(x) \wedge \texttt{Role}(y))$

$\forall(x)\exists^{\geq 2}y(\texttt{Contains}(x,y))$

$\forall(x)(\texttt{Role}(x) \rightarrow \exists(y)(\texttt{Contains}(y,x)))$

$\forall(x,y,z)(\texttt{Contains}(x,y) \wedge \texttt{Contains}(z,y) \rightarrow (x=z))$

$\forall(x,y,z)(\texttt{RolePlaying}(x,y,z) \rightarrow \texttt{Role}(x) \wedge \texttt{CardinalityConstraint}(y) \wedge \texttt{EntityType}(z))$

$\forall(x)(\texttt{Role}(x) \rightarrow \exists(y,z)(\texttt{RolePlaying}(x,y,z)))$

$\forall(x,y,z,v,w)(\texttt{RolePlaying}(x,y,z) \wedge \texttt{RolePlaying}(x,v,w) \rightarrow (y=v) \wedge (z=w))$

$\forall(x,y,z,v,w)(\texttt{RolePlaying}(x,y,z) \wedge \texttt{RolePlaying}(v,y,w) \rightarrow (x=v) \wedge (z=w))$

$\forall(x)(\texttt{CardinalityConstraint}(x) \rightarrow \exists(y)(\texttt{MinimumCardinality}(x,y) \wedge \texttt{Integer}(y)))$

$\forall(x)(\texttt{CardinalityConstraint}(x) \rightarrow \exists(y)(\texttt{MaximumCardinality}(x,y) \wedge \texttt{Integer}(y)))$

$\forall(x,y)(\texttt{Identifies}(x,y) \rightarrow (\texttt{IdentificationConstraint}(x) \wedge \texttt{ObjectType}(y)))$

$\forall(x)(\texttt{IdentificationConstraint}(x) \rightarrow \exists(y)(\texttt{Identifies}(x,y)))$

$\forall(x,y,z)((\texttt{Identifies}(x,y) \wedge \texttt{Identifies}(x,z)) \rightarrow (y=z))$

$\forall(x)(\texttt{ObjectType}(x) \rightarrow \exists(y)(\texttt{Identifies}(y,x)))$

$\forall(x,y,z)((\texttt{DeclaredOn}(x,y) \wedge \texttt{DeclaredOn}(x,z) \wedge \texttt{IdentificationConstraint}(x) \wedge (\neg(y=z)))$
$(\texttt{ValueProperty}(y) \leftrightarrow \neg\texttt{AttributiveProperty}(z)))$

$\forall(x)(\texttt{IdentificationConstraint}(x) \rightarrow \exists(y)(\texttt{DeclaredOn}(x,y)))$

$\forall(x,y)((\texttt{DeclaredOn}(x,y) \wedge \texttt{SingleIdentification}(x)) \rightarrow (\texttt{Attribute}(y) \vee \texttt{ValueType}(y)))$

$\forall(x)(\texttt{SingleIdentification}(x) \rightarrow \exists(y)(\texttt{DeclaredOn}(x,y))$

$\forall(x,y,z)((\texttt{SingleIdentification}(x) \wedge \texttt{DeclaredOn}(x,y) \wedge \texttt{DeclaredOn}(x,z)) \rightarrow (y=z))$

Highlighted section for step 3

$\forall(x,y)(\texttt{Contains}(x,y) \rightarrow \texttt{Relationship}(x) \land \texttt{Role}(y))$
$\forall(x)\exists^{\geq 2}y(\texttt{Contains}(x,y))$
$\forall(x)(\texttt{Role}(x) \rightarrow \exists(y)(\texttt{Contains}(y,x)))$
$\forall(x,y,z)(\texttt{Contains}(x,y) \land \texttt{Contains}(z,y) \rightarrow (x=z))$
$\forall(x,y,z)(\texttt{RolePlaying}(x,y,z) \rightarrow \texttt{Role}(x) \land \texttt{CardinalityConstraint}(y) \land \texttt{EntityType}(z))$
$\forall(x)(\texttt{Role}(x) \rightarrow \exists(y,z)(\texttt{RolePlaying}(x,y,z)))$
$\forall(x,y,z,v,w)(\texttt{RolePlaying}(x,y,z) \land \texttt{RolePlaying}(x,v,w) \rightarrow (y=v) \land (z=w))$
$\forall(x,y,z,v,w)(\texttt{RolePlaying}(x,y,z) \land \texttt{RolePlaying}(v,y,w) \rightarrow (x=v) \land (z=w))$
$\forall(x)(\texttt{CardinalityConstraint}(x) \rightarrow \exists(y)(\texttt{MinimumCardinality}(x,y) \land \texttt{Integer}(y)))$
$\forall(x)(\texttt{CardinalityConstraint}(x) \rightarrow \exists(y)(\texttt{MaximumCardinality}(x,y) \land \texttt{Integer}(y)))$
$\forall(x,y)(\texttt{Identifies}(x,y) \rightarrow (\texttt{IdentificationConstraint}(x) \land \texttt{ObjectType}(y)))$
$\forall(x)(\texttt{IdentificationConstraint}(x) \rightarrow \exists(y)(\texttt{Identifies}(x,y)))$
$\forall(x,y,z)((\texttt{Identifies}(x,y) \land \texttt{Identifies}(x,z)) \rightarrow (y=z))$
$\forall(x)(\texttt{ObjectType}(x) \rightarrow \exists(y)(\texttt{Identifies}(y,x)))$
$\forall(x,y,z)((\texttt{DeclaredOn}(x,y) \land \texttt{DeclaredOn}(x,z) \land \texttt{IdentificationConstraint}(x) \land (\neg(y=z)))$
$(\texttt{ValueProperty}(y) \leftrightarrow \neg\texttt{AttributiveProperty}(z)))$
$\forall(x)(\texttt{IdentificationConstraint}(x) \rightarrow \exists(y)(\texttt{DeclaredOn}(x,y)))$
$\forall(x,y)((\texttt{DeclaredOn}(x,y) \land \texttt{SingleIdentification}(x)) \rightarrow (\texttt{Attribute}(y) \lor \texttt{ValueType}(y)))$
$\forall(x)(\texttt{SingleIdentification}(x) \rightarrow \exists(y)(\texttt{DeclaredOn}(x,y))$
$\forall(x,y,z)((\texttt{SingleIdentification}(x) \land \texttt{DeclaredOn}(x,y) \land \texttt{DeclaredOn}(x,z)) \rightarrow (y=z))$

Transformations approach [Khan et al.(2016)]

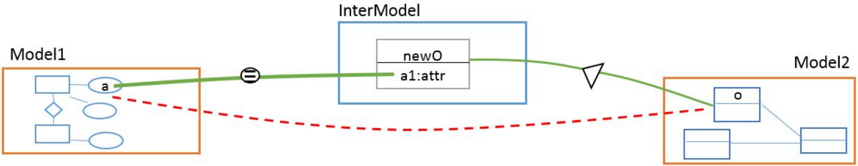
- Assume the models, called `Model1` and `Model2`, are syntactically correct
- Allow equivalence, disjointness, and subsumption axioms between homogeneous *metamodel* entities
- Introduce a third model, called `Intermodel`, that keeps all intermediate metamodel entities that are necessary to implement the transformation rule, adhering to constraints of metamodel
- Note: the original link has no direction, i.e., without source and target models

Transformations approach [Khan et al.(2016)]

- Assume the models, called `Model1` and `Model2`, are syntactically correct
- Allow equivalence, disjointness, and subsumption axioms between homogeneous *metamodel* entities
- Introduce a third model, called `Intermodel`, that keeps all intermediate metamodel entities that are necessary to implement the transformation rule, adhering to constraints of metamodel
- Note: the original link has no direction, i.e., without source and target models
- ATL uses notion of 'source' and 'target'
  - ⇒ Solution: `Model1` and `Model2` both serve as 'source', and `Intermodel` as 'target'

Attribute $\leftrightarrow$ Object type transformation

- An attribute $A \mapsto C \times \mathtt{D}$ becomes an object type A' with a new stub attribute $a \mapsto A' \times \mathtt{D}$ and has a relationship R to an object type C

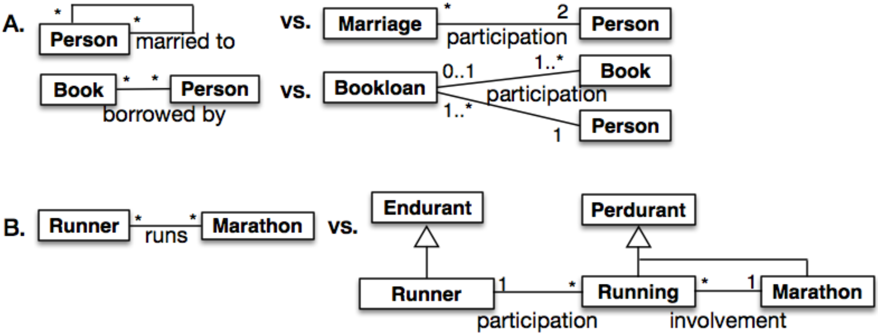


In ATL

```
rule Att<-->OT {
 from
     a : Model1.MM!Attribute (a.range(dt)),
     o : Model2.MM!ObjectType
 to
     newO : InterModel.MM!ObjectType ( newO.hasAttribute <-- a1 ),
     a1 : InterModel.MM!Attribute ( a1.domain <-- newO,
              a1.range<--dt, a1.of <-- co ),
     e : InterModel.MM!EqualityConstraint( e.declaredOn(a),
               e.declaredOn(a1)),
     co : InterModel.MM!CardO ( co.cardinalityConstraint <-- cc,
             co.attribute <-- a1, co.objectType <-- newO ),
     sid : InterModel.MM!SingleIdentification ( sid.declaredOn <-- a1,
             sid.identifies <-- newO, sid.mandatory <-- mc),
     m : InterModel.MM!Mandatory ( m.declaredOn <-- a1.contains ),
     cc : InterModel.MM!CardinalityConstraint (cc.maximumCardinality <-- 1,
             cc.minimumCardinality <-- 1),
     s : InterModel.MM!Subsumption ( s1.super <-- newO, s1.sub <-- o )
}
```

Bridging different modelling decisions [Fillottrani and Keet(2017)]

'Case A': class or object property? (and Case B with perdurants)



- Main issue: to reify or not to reify?
- And: are the more precise cardinality constraints needed?

Formalisation—preliminaries

- Language of pattern instantiation (OWL in this case)
- Language for patterns with vocabulary $\mathcal{V}$, meta-level (second-order) elements (or stereotypes)
- Ontology pattern, with name, elements from $\mathcal{V}$, pattern axiom components, pattern's full formalisations; e.g.:
  - pattern name: *basic all-some*
  - pattern elements: $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{R}_1$
  - pattern axiom components: $\sqsubseteq$, $\exists$
  - pattern's full formalisation $\mathcal{C}_1 \sqsubseteq \exists \mathcal{R}.\mathcal{C}_2$

  Example instantiation: $\texttt{Professor} \sqsubseteq \exists \texttt{teaches.Course}$
- Homogeneous mapping: subsumption or equivalence relating two homogeneous elements (e.g., `Teacher` in O and `Instructor` in O')

Definition (Ontology Pattern Alignment, *OPA*)

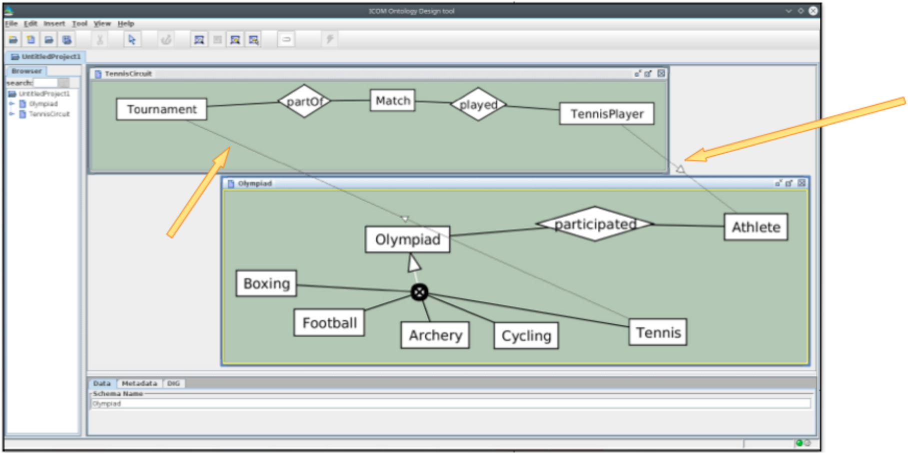
An *ontology pattern alignment OPA* consists of two ontology patterns, P and P', such that its signature Σ is a subset of the signature of the respective ontologies O and O', i.e., $\Sigma(P) \subseteq \Sigma(O)$ and $\Sigma(P') \subseteq \Sigma(O')$, and alignment axioms

- alignment pattern name;
- pattern elements;
- alignment patterns' context, consisting of:
  - O's pattern P
  - O''s pattern P'
- alignment pattern axiom component(s) from V_X;
- pattern alignment's formalisation, composed of:
  - a (possibly empty) set of mappings between homogeneous elements in P and P'
  - a set of axioms made from components in V_X connecting heterogeneous elements in P and P'

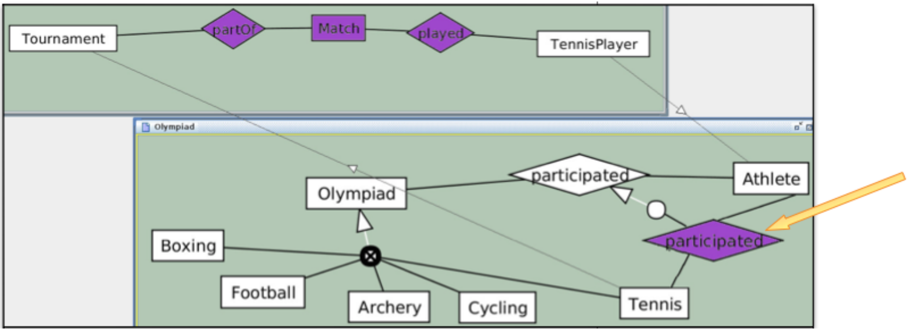
Class vs. Object Property (case A)

- *alignment pattern name*: *class-OP*
- *pattern elements*: $\mathcal{C}_1, \mathcal{C}_2, \mathcal{R}_1$ from O, $\mathcal{C}'_3, \mathcal{C}'_4, \mathcal{C}'_5, \mathcal{R}'_2, \mathcal{R}'_3$ from O'
- *alignment patterns' contexts*:
  - *pattern P in O*: $\exists\mathcal{R}_1.\mathcal{C}_2 \sqsubseteq \mathcal{C}_1$ and $\exists\mathcal{R}_1^-.\mathcal{C}_1 \sqsubseteq \mathcal{C}_2$;
  - *pattern P' in O'*: $\exists\mathcal{R}'_2.\mathcal{C}'_4 \sqsubseteq \mathcal{C}'_3$, $\exists\mathcal{R}'^-_2.\mathcal{C}'_3 \sqsubseteq \mathcal{C}'_4$, $\exists\mathcal{R}'_3.\mathcal{C}'_5 \sqsubseteq \mathcal{C}'_3$, $\exists\mathcal{R}'^-_3.\mathcal{C}'_3 \sqsubseteq \mathcal{C}'_5$, $\mathcal{C}'_3 \sqsubseteq (\exists\mathcal{R}'_2)$, and $\mathcal{C}'_3 \sqsubseteq (\exists\mathcal{R}'_3)$.
- *pattern's full formalisation*:
  - *homogeneous mappings*: between $\mathcal{C}_1$ and $\mathcal{C}'_4$ and between $\mathcal{C}_2$ and $\mathcal{C}'_5$, which may be subsumption or equivalence relations.
  - *heterogeneous alignments*: $\exists\mathcal{R}_1 \sqsubseteq \mathcal{C}'_3$, $\exists\mathcal{R}_1^- \sqsubseteq \mathcal{C}'_3$, $\mathcal{C}'_3 \sqsubseteq \exists\mathcal{R}_1 \sqcap \exists\mathcal{R}_1^- \sqcap (\leq 1\,\mathcal{R}_1) \sqcap (\leq 1\,\mathcal{R}_1^-)$.

Example: mapping and searching (‘Case A’: Class↔OP)



Example: checking and accept/reject alignment



back to main