An ontology-driven unifying metamodel for UML
Class Diagrams, EER, and ORM2

C. Maria Keet

School of Mathematics, Statistics, and Computer Science,
University of KwaZulu-Natal, South Africa,
keit@ukzn.ac.za

joint work with Pablo Rúben Fillottrani
Universidad Nacional del Sur, Argentina

Dagstuhl Seminar “Automated Reasoning on Conceptual Schemas”,
Schloss Dagstuhl, Germany, May 20-24, 2013
Setting

- NRF/DST- and MINCyT-funded Project “Ontology-driven unification of conceptual modelling languages”
Setting

- NRF/DST- and MINCyT-funded Project “Ontology-driven unification of conceptual modelling languages”
- Aims of the project:
  - inter-model assertions among EER, UML v2.4.1, ORM2;
  - one formalization including all (structural, static) language features, where each of the languages is a fragment;
  - (converting among the representations, and reasoning across models);
  - some module management
Setting

- NRF/DST- and MINCyT-funded Project “Ontology-driven unification of conceptual modelling languages”
- Aims of the project:
  - inter-model assertions among EER, UML v2.4.1, ORM2;
  - one formalization including all (structural, static) language features, where each of the languages is a fragment;
  - (converting among the representations, and reasoning across models);
  - some module management

⇒ First step: identify commonalities and differences in terminology, syntax, semantics, and ontological commitments of the structural components of the three main languages (EER, UML Class Diagrams v2.4.1, ORM2)
⇒ Metamodel
### A motivation why first metamodelling

<table>
<thead>
<tr>
<th>$\mathcal{DLR}_{ifd}$</th>
<th>OWL 2 DL</th>
<th>FOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>– no implementation</td>
<td>+ several automated reasoners, relatively scalable</td>
<td>– few reasoners, not really scalable</td>
</tr>
<tr>
<td>– no interoperability</td>
<td>+ linking with ontologies doable</td>
<td>– no interoperability with existing infrastructures</td>
</tr>
<tr>
<td>– no integration</td>
<td>+ ‘integration’ with OntoIOP</td>
<td>+ ‘integration’ with OntoIOP</td>
</tr>
<tr>
<td>+ formalisation exist</td>
<td>– formalisation to do</td>
<td>± formalisation exist</td>
</tr>
<tr>
<td>+ little feature mismatch</td>
<td>– what to do with OWL 2 DL features not in the CM languages and vv.</td>
<td>+ little feature mismatch</td>
</tr>
<tr>
<td>– modularity infrastructure</td>
<td>+ modularity infrastructure</td>
<td>– modularity infrastructure</td>
</tr>
</tbody>
</table>
Ontology-driven

- Uncover ontological decisions embedded in the modelling language, among others:
  - Positionalism of relations (nature of relations)
  - Identification mechanisms (identity)
  - Attributes, ‘attribute-free’ or ‘attribute-hidden’ (attributions, quality properties)
  - Subrelations (meaning of a sub-relation)
  - Any differences/similarities for constraints (e.g., on when a relationship may be objectified)
Notes for the metamodel

- We use UML Class Diagram notation for the metamodel
- Not all constraints can be represented in that diagram, but added as textual constraint
- It has some redundancies (from a logic-based perspective), e.g., multivalued attributes
- Not all features may be ‘good’ features, but we do not judge about elegance (can be addressed in a formalization)
- Table with naming conventions for UML, EER, ORM2, and the metamodel terms
Principal static entities of the metamodel
Static structural components

Introduction

Constraint

- Uniqueness constraint
- Relationship constraint
- Disjointness constraint
- Equality constraint

- Internal uniqueness
- External uniqueness

- Disjoint roles
- Disjoint relationship types
- Role equality

- Disjoint entity types
- Join-equality constraint

- Join constraint
- Join-disjointness constraint

- Transitivity
- Antisymmetry
- Irreflexivity
- Reflexivity

- Asymmetry
- Purely-reflexive

- Acyclicity
- Intransitivity

- Strongly intransitive
- Frequency constraint

- Identification constraint
- Extended frequency constraint

- Object type equality

- Value constraint
- Role value constraint

- Value type constraint

- Completeness constraint
- Value comparison constraint

- Value constraint

- Mandatory constraint
- Inclusive mandatory

- Mandatory

- Disjunctive mandatory

- Object type equality

- Join-subset constraint
- Join-equality constraint

- Subset constraint
- Cardinality constraint

- Attibutive property cardinality
- Object type cardinality

- External identification
- Internal identification

- Single identification

- Internal identification

- External identification

- Join-disjointness constraint
UML, EER, and ORM are all **positionalist** [Keet(2009)]

- *n*-ary relationship
- Role that an entity plays in a relationship
- No order on the roles (or: ‘relationship’ as a set of roles), but one can add an order
- Relationship composed of roles

Optional predicate, with order and no roles

Cardinality

Nested object type
Principal relationships between Relationship, Role, and Entity type
Notes

- Relations between roles and a predicate can only exist if there is a relation between those roles and the relationship that that predicate is an ordering of (i.e., it is a join-subset).
- Entities that participate in the predicate must play those roles that compose the relationship of which that predicate is an ordered version of it.
Roles, relationships, and predicates

Subsumption and aggregation

A. subsumer

Entity

0..* subsumed by

has part

0..*

Entity type

0..*

part of

has components

0..*

Entity type

0..*

aggregates into

has components

0..*

Entity type

0..1

is composite of

B. Entity

sub 1

super 1

(disjoint)

Relationship

participant 2

participates in

0..*

0..*

Subsumption

PartWhole

0..* 0..*

Part

2..*

Shared aggregate

1

Composite aggregate

1

Attributive property

Composite attribute

part 1

1 whole

whole

1 part

1 whole

A. B.
What are attributes?

- An *attribute* \((A)\) is a binary relationship between a relationship or entity type \((R \cup E)\) and a data type \((D)\), i.e., \(A \mapsto R \cup E \times D\).
- An attribute is no more, and no less.
- For instance, one can have an attribute hasColour, that relates an object type to a string; e.g, hasColour \(\mapsto\) Flower \(\times\) String.
Attributes and value types

Attribution in Ontology and ontologies

- Principally as *quality property*, formalised as unary predicate
- Separate relation to endurants or perdurants
- Separate relation to ‘values’ (qualia) of an attribution
Attributes and value types

Attribution in Ontology and ontologies

- Principally as *quality property*, formalised as unary predicate
- Separate relation to endurants or perdurants
- Separate relation to ‘values’ (qualia) of an attribution
- Implemented as such in foundational ontologies, such as DOLCE and GFO
Attributes and value types

AttrIBUTION IN ONTOLOGY AND ONTOLOGIES

- Principally as *quality property*, formalised as unary predicate
- Separate relation to endurants or perdurants
- Separate relation to ‘values’ (qualia) of an attribution
- Implemented as such in foundational ontologies, such as DOLCE and GFO
- Practically, the same quality property can be related to more than one entity
Examples of ‘attributes’ in UML, EER, ORM

A. UML Class Diagram (two options)

<table>
<thead>
<tr>
<th>Apple</th>
<th>colour: string</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>weight: integer</td>
</tr>
</tbody>
</table>

Apple

<table>
<thead>
<tr>
<th>Apple</th>
<th>colour</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 String</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Apple</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 Integer</td>
</tr>
</tbody>
</table>

B. ORM 2 (two options)

Apple

Colour

Weight

Apple

Colour (name)

Weight (kg)

Apple

C. ER (Barker notation)

APPLE

* colour

* weight

D. EER (bubble notation)

Apple

Colour

Weight
Dimensional attributes and value types

- *dimension* for the value: implicit meaning in the values for some data types, which has to do with measurements
- e.g., hasHeight ↦ Flower × Integer does not contain any of that information, but somehow has to be included
### Dimensional attributes and value types

- **dimension** for the value: implicit meaning in the values for some data types, which has to do with measurements.

- e.g., `hasHeight \mapsto \text{Flower} \times \text{Integer}` does not contain any of that information, but somehow has to be included.

- How? e.g.:
  - `hasHeight \mapsto \text{Flower} \times \text{Integer} \times \text{cm}`
    - or perhaps with an approach along the line of:
  - `hasHeight \mapsto \text{Flower} \times \text{Height}`
  - `mapped\_to \mapsto \text{Height} \times \text{Integer}`
  - `hasDimension \mapsto \text{Integer} \times \text{cm}`
Dimensional attributes and value types

- *dimension* for the value: implicit meaning in the values for some data types, which has to do with measurements
- e.g., `hasHeight ↦ Flower × Integer` does not contain any of that information, but somehow has to be included
- How? e.g.:
  - `hasHeight ↦ Flower × Integer × cm`
  - or perhaps with an approach along the line of:
    - `hasHeight ↦ Flower × Height`
    - `mapped_to ↦ Height × Integer`
    - `hasDimension ↦ Integer × cm`
- Within the scope of unification, add the notion of dimension (not a whole system of recording measurement data for a specific scenario)
Attributes and value types

Attributes in the metamodel

```
Relationship
  Attribute
     {disjoint}
  Dimensional attribute

Attributive property
  0..* domain
  0..* {or} 0..*

Object type
  0..* domain

Data type
  range 1

Dimension
  1

dimensional attribution
  0..*
  1
```
Value types in the metamodel
Related work: data and schema level

- Mapping and transformation algorithms using a common hypergraph, for small subsets of ER, UML, and ORM; set-based semantics vs. a model-theoretic semantics [Boyd and McBrien(2005)]
- Physical schema layer [Bowers and Delcambre(2006)]
Related work: conceptual data modelling and software engineering

- Compare the languages through their metamodels in ORM, highlight differences [Halpin(2004)]
- Metamodel for a part of ER and a part of NIAM in CoCoA and implemented in MViews, Pounamu [Venable and Grundy(1995), Grundy and Venable(1996), Zhu et al.(2004)]; omits, a.o., value types, composite attributes, and e.g., NIAM is forced to have the attributes as in ER
- (linking/integrating conceptual models represented in the same conceptual data modelling language [Atzeni et al.(2008), Fillottrani et al.(2012)])
Related work: Knowledge representation and reasoning

- Approach: mainly, choose a logic and show it fits neatly/sufficiently with one or more conceptual data modelling languages
- Partial unifications, using e.g., $ALUNI$ [Calvanese et al.(1999)], several $DL$-$Lite$ fragments [Artale et al.(2007)], $DLR_{ifd}$ [Keet(2008)]
- They cannot simply be linked up and implemented
Related work: Knowledge representation and reasoning

- Approach: mainly, choose a logic and show it fits neatly/sufficiently with one or more conceptual data modelling languages
- Partial unifications, using e.g., $ALUNI$ [Calvanese et al.(1999)], several $DL$-$Lite$ fragments [Artale et al.(2007)], $DLR_{ifd}$ [Keet(2008)]
- They cannot simply be linked up and implemented
- Distributed Ontology Language [Mossakowski et al.(2012)] and system that is currently being standardised by ISO (http://ontoiop.org).
Conclusions

- Unifying, ontology-driven metamodel capturing most of ORM/FBM, EER, and static UML v2.4.1 w.r.t their static, structural, entities, their relationships, and constraints
- The only intersection among all these conceptual data modelling languages are role, relationship, and object type
- Adhere to the positionalist commitment of the meaning of relationship
- Attributions are represented differently in each language, but, ontologically, they denote the same notions
- Several implicit aspects, such as dimensional attribute and its reusability and relationship versus predicate, have been made explicit
- Common constraints: disjointness, completeness, simple mandatory, object type cardinality
Current work

- Two papers submitted
- Near future: formalisation in FOL, then OWL 2 DL subset
- Extension of tool that will aid the process of complex systems design and information integration
References I


References II


