Deductive reconstruction of MLT* for multi-level modeling

Manfred A. Jeusfeld[1], João Paulo A. Almeida, Victorio A. Carvalho
Claudenir M. Fonseca, Bernd Neumayr

[1] University of Skövde, Sweden, manfred.jeusfeld@acm.org

- Understand the relationship between MLT* and DeepTelos (both power-type based)
- Specify MLT-Telos with ConceptBase/O-Telos
- Create an efficient implementation for MLT* to check large multi-level models

All sources and executable examples are published at http://conceptbase.cc/mlt-telos/

Summary for the impatient

MLT* ⊃ MLT-Telos ⊃ DeepTelos

Summary for the slightly less impatient

first order logic

Datalog with negation (minimal Herbrand interpretation)

MLT* ⊃ MLT-Telos ⊃ DeepTelos

MLT*

- axiomatic definition
- open-world semantics
- axioms need to be satisfied by models
- powertype and categorization
- implemented by Alloy

MLT-Telos

- rules and constraints
- closed-world semantics
- constraints may not be violated
- powertype and categorization
- implemented by ConceptBase

DeepTelos

- rules and constraints
- closed-world semantics
- constraints may not be violated
- powertype
- implemented by ConceptBase

partial implementation of

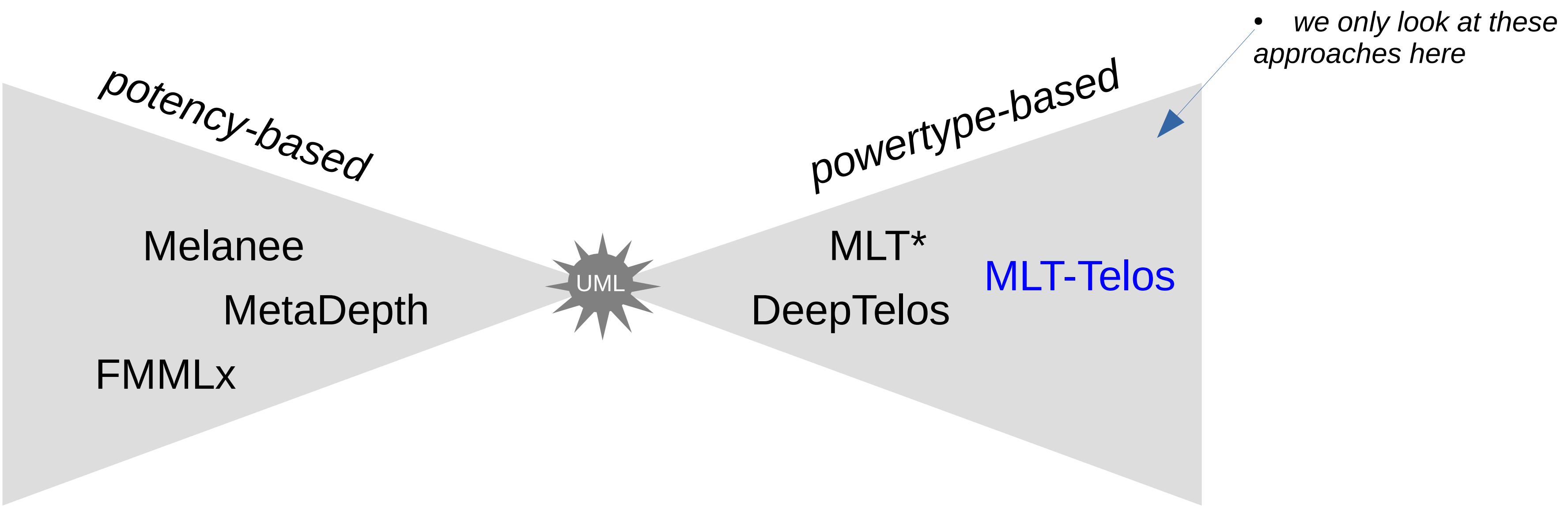
adds categorization

First-order vs. Datalog: A question of semantics

	reasoning		*query processing*	
definitions	First-order theory (formulas)	$\Sigma \Rightarrow \varphi$	Datalog program (rules + **facts**)	DB $\Rightarrow$ query
semantics	All interpretations		Unique Minimal Herbrand Interpretation	

First-order theories can be checked on consistency (by reasoning about all interpretations),
Datalog programs can be used to compute the unique minimal Herbrand interpretation.
Integrity checking in Datalog is a form of query processing.

Two multi-level modeling approaches



- classes, attributes and associations get explicit level/potency numbers

- classes can have powertype associations
- levels are implicit (derived/optional)

Both principal approaches are capable of passing the multi-level modeling challenges, see MULTI'19

MLT* has three predefined types

- *these are facts in the ConceptBase system*

Listing 1: MLT-Telos predefined types

```
ENTITY in TYPE end
TYPE in TYPE isA ENTITY end
INDIVIDUAL in TYPE isA ENTITY end
```

These are the three MLT* predefined types ENTITY, TYPE, and INDIVIDUAL as O-Telos classes.

O-Telos provides instantiation ('in'), specialization ('isA'), and attribution/relations, see http://conceptbase.sourceforge.net/userManual81/cbm003.html

Structural definitions for MLT-Telos as O-Telos frames

Listing 2: Powertype and specialization

```
TYPE with
  irreflexive,antisymmetric,single,revsingle
    isPowerTypeOf: TYPE
  reflexive,antisymmetric,transitive
    specializes: TYPE
  attribute
    properSpecializes: TYPE
end
```

The 'isPowerType' and 'specializes' relations are defined for TYPE.
The categories like 'irreflexive' come with predefined rules and constraints realizing their semantics, see also
http://conceptbase.sourceforge.net/mlt-telos/SOURCES/System-oHome.sml

From First-order to Datalog: a lossy translation

MLT* definition of ”individuals” and 'types':

$$\forall x\ \mathrm{Individual}(x) \leftrightarrow \neg\exists y\ \mathrm{iof}(y, x)$$
$$\forall x\ \mathrm{Type}(x) \leftrightarrow \exists y\ \mathrm{iof}(y, x)$$

Approximation in MLT-Telos by constraints:

Listing 3: Individual objects

```
forall x/INDIVIDUAL not exists y/ENTITY (y in x)
forall x/INDIVIDUAL not (x in TYPE)
```

In Datalog, facts like (x in INDIVIDUAL) must be either explicitely stated or derived from explicit facts by rules. In MLT*, we have true definitions of the terms.

The powertype relation in MLT*

MLT* definition of 'isPowerTypeOf':

$$\forall t_1, t_2 \text{ isPowertypeOf}(t_1, t_2)$$
$$\leftrightarrow \text{type}(t_1) \wedge \forall t_3(\text{iof}(t_3, t_1) \leftrightarrow \text{specializes}(t_3, t_2))$$

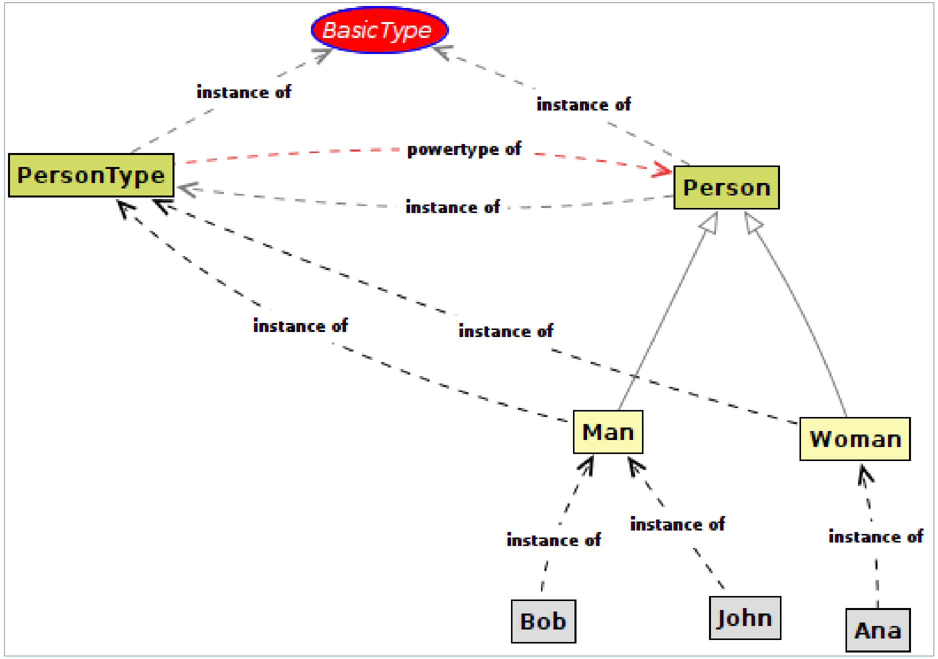
Approximation in MLT-Telos by deductive rules:

Listing 6: Semantics of isPowerTypeOf

```
forall t1,t2,t3  /TYPE (t1 isPowerTypeOf t2) and
  (t3 in t1) and not (t3 isA t2) ==> (t3 specializes t2)

forall t1,t2,t3/TYPE (t1 isPowerTypeOf t2) and
  (t3 specializes t2) ==> (t3 in t1)
```


Example 1: Simple powertype relation



Model 2: Listing for figure 2

```
Person in TYPE end
PersonType in TYPE with isPowerTypeOf type: Person end
Man in TYPE,PersonType end    Woman in TYPE,PersonType end
John in INDIVIDUAL,Man end    Bob in INDIVIDUAL,Man end
Ana in INDIVIDUAL,Woman end
```

The explicit facts in the listing lead to derived facts, in particular the 'specializes' relations between 'Person', 'Man', and 'Woman' (rule 1 of listing 6).
Note also the derived instantiation of 'Person' to 'PersonType' (rule 2 of listing 6).

Categorization in MLT*

MLT* supports besides the powertype construct the categorization of classes, e.g. the disjoint and complete categorization of the class 'Person' by gender.

Approximation in MLT-Telos by constraints and rules (excerpt):

Listing 10: Specializations deduced from categorization

```
forall t1,t2,s1,s2/TYPE (t1 isPowerTypeOf t2) and
 (s1 completelyCategorizes s2) and (s1 specializes t1)
  ==> (s2 specializes t2)

forall t1,t2,s1,s2/TYPE (t1 isPowerTypeOf t2) and
 (s1 categorizes s2) and (s2 specializes t2)
  ==> (s1 specializes t1)
```

Using queries to check disjointness and completeness

Listing 11: Completeness and disjointness

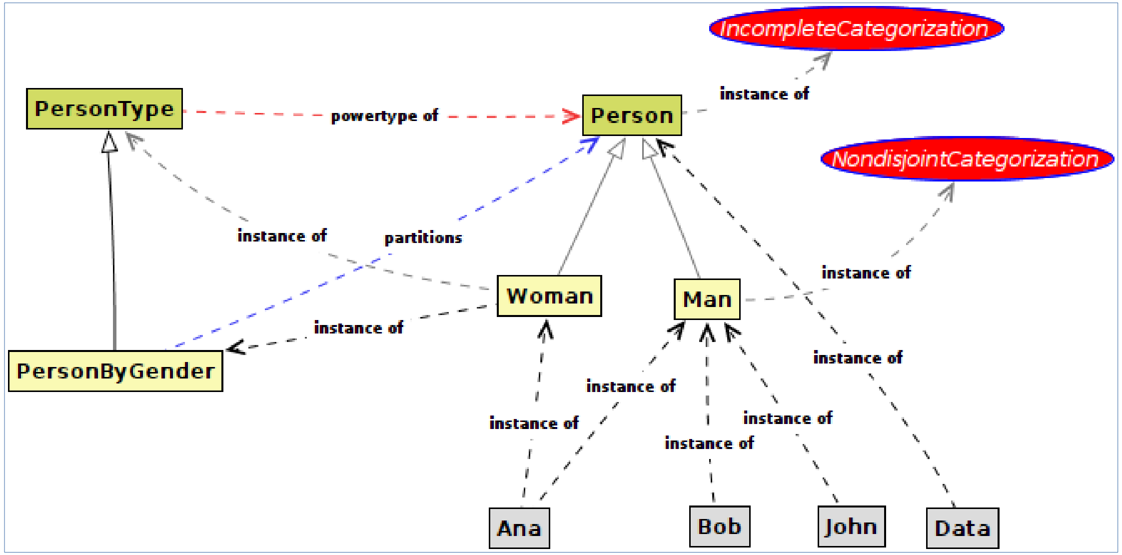
```
IncompleteCategorization in QueryClass isA TYPE with
  computed_attribute entity: ENTITY
  constraint isIncomplete : $ exists t1/TYPE
     (t1 completelyCategorizes this) and (~entity in this) and
        not (exists t2/TYPE  (t2 properSpecializes this) and
        (~entity in t2)) $
end

NondisjointCategorization in QueryClass isA TYPE with
  computed_attribute entity: ENTITY; type: TYPE
  constraint isNondisjoint: $ exists t1,t2/TYPE
        (t1 disjointlyCategorizes t2) and
        (this properSpecializes t2) and
        (~type properSpecializes t2) and (this <> ~type) and
        (~entity in ~type) and (~entity in this) $
end
```

Queries in ConceptBase are mapped to Datalog rules.

Here, they are used to return 'violators' of corresponding MLT* axioms.

Example 2: The partitions construct (disjoint and complete categorization)



Model 3: Listing for figure 3

```
Person in TYPE end
PersonType in TYPE with isPowerTypeOf type: Person end
PersonByGender in TYPE with specializes t1: PersonType
                            partitions t2: Person end
Man in TYPE,PersonByGender end
Woman in TYPE,PersonByGender end
John in INDIVIDUAL,Man end        Bob in INDIVIDUAL,Man end
Ana in INDIVIDUAL,Man,Woman end  Data in INDIVIDUAL,Person end
```

Answers to the queries of listing 11:

```
Person in IncompleteCategorization with
   entity
    COMPUTED_entity_id_3470 : Data
end

Man in NondisjointCategorization with
   entity
    COMPUTED_entity_id_3466 : Ana
  type
    COMPUTED_type_id_3457 : Woman
end
```

The two queries show how models can be checked for errors without having to reject them due to the violation of integrity constraints.

DeepTelos vs. MLT-Telos

- The paper lists the deductive rules to map a DeepTelos model into an MLT-Telos model.
- Hence, DeepTelos is a subset of MLT-Telos. It lacks the categorization constructs, the 'powertype' construct is virtually the same.

- MLT* does not specifically define how to handle attributes and relations defined for classes.
- MLT-Telos can take over this aspect from DeepTelos.

- DeepTelos passed the MULTI'19 challenge. Hence, MLT-Telos will do as well!

Limitations

- We did not yet test large models. The MLT-Telos implementation may turn out to be inefficient for testing large multi-level models.

- O-Telos (the axiomatically defined language used by ConceptBase) defines its own specialization construct 'isA'. This is not completely integrated with the MLT-Telos construct 'specializes' though they are virtually the same. This leads to a few problems when specializing attributes or relations.

- The fundamental limitation of MLT-Telos is that it uses the minimal Herbrand interpretation of Datalog. For example, MLT-Telos cannot derive powertype relations from a given model. Datalog distiguishes conditions and conclusions in deductive rules. First-order logic does not. The position of a predicate in a formula does not qualify it as being derived.

- Definition of 'BasicClass' in MLT-Telos is fairly simple. Similar with 'OrderedType'.

Benefits of MLT-Telos

- Specification (O-Telos source code) fits on one page.
- Potential for an efficient implementation due to the solid Datalog engine of ConceptBase.
- Number of query classes to check correctness can be extended by query classes that check the quality (fitness for use) of a multi-level model.

All sources and executable examples are published under CC BY-SA at
http://conceptbase.sourceforge.net/mlt-telos/
The web page contains further information on the implementation.
Check them out by installing the free ConceptBase system.

Summary: MLT* ⊃ MLT-Telos ⊃ DeepTelos

Next: Understand relation to potency-based approaches
Then: Large models

http://conceptbase.cc/mlt-telos/

  Kinnekulle