Deductive reconstruction of MLT* for multi-level modeling

Manfred A. Jeusfeld
University of Skövde / IIT
Skövde, Sweden
manfred.jeusfeld@acm.org

João Paulo A. Almeida
Federal University of Espírito Santo
Vitória, ES, Brazil
jpalmeida@ieee.org

Victorio A. Carvalho
Federal Institute of Espírito Santo
Colatina, ES, Brazil
victorio@ifes.edu.br

Claudenir M. Fonseca
Free University of Bozen-Bolzano
Bolzano, Italy
cmoraisfonseca@unibz.it

Bernd Neumayr
Johannes Kepler University Linz
Linz, Austria
neumayr@dke.uni-linz.ac.at

ABSTRACT

In the last two decades, about a dozen proposals were made to extend object-oriented modeling by multiple abstraction levels. One group of proposals designates explicit levels to objects and classes. The second group uses the powertype pattern to implicitly establish levels. From this group, we consider two proposals, DeepTelos and MLT*. Both have been defined via axioms and both give a central role to the powertype pattern. In this paper, we reconstruct MLT* with the deductive axiomatization style used for DeepTelos. The resulting specification is executed in a deductive database to check MLT* multi-level models for errors and complete them with derived facts that do not have to be explicitly asserted by modelers. This leverages the rich rules of MLT* with the deductive approach underlying DeepTelos. The effort also allows us to clearly establish the relation between DeepTelos and MLT*, in an attempt to clarify the relations between approaches in this research domain. As a byproduct, we supply MLT-Telos as a fully operational deductive implementation of MLT* to the research community.

CCS CONCEPTS

• **Software and its engineering → Syntax**; **Semantics**; **Domain specific languages**; **Visual languages**.

KEYWORDS

Multi-level modeling, Object-oriented modeling, MLT*, Powertype, ConceptBase, Datalog, DeepTelos

ACM Reference Format:
Manfred A. Jeusfeld, João Paulo A. Almeida, Victorio A. Carvalho, Claudenir M. Fonseca, and Bernd Neumayr. 2020. Deductive reconstruction of MLT* for multi-level modeling. In *ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems (MODELS '20 Companion), October 18–23, 2020, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3417990.3421410

MODELS '20 Companion, October 18–23, 2020, Virtual Event, Canada

ACM ISBN 978-1-4503-8135-2/20/106…$15.00
https://doi.org/10.1145/3417990.3421410

1 INTRODUCTION

Multi-level modeling approaches developed over last two decades can be roughly divided into two flavors. In the first one, objects and classes are *explicitly designated* by the modeler to a particular "level". Attributes and relations get so-called "potencies" that specify how many levels lower they can be instantiated. Example realizations of this flavor are Melanee [19], MetaDepth [9], and FMMLx [12]. The second flavor uses the powertype pattern [6, 22, 23] to form sequences of powertype relations between certain classes to establish levels implicitly. Two examples of this approach are MLT* [1] and DeepTelos [18]. The multi-level community has undertaken a number of attempts to compare and unify the existing proposals for multi-level modeling. One method is to compare the proposals and tools by feature lists. Another method is to define demanding challenges, for which the researchers behind the multi-level modeling tools submit solutions. The complexity and expressiveness of the solutions is then evaluated by the community. In this paper, we follow a different approach. We review the axioms of the multi-level theory MLT* and implement them in the same system, ConceptBase [13, 15], that was used to define DeepTelos. Differences between MLT* and DeepTelos have be dealt with in this process. First of all, the axioms and definitions of MLT* are expressed in first-order logic with classical open-world semantics, while ConceptBase intentionally restricts itself to Datalog-neg [8] with minimal Herbrand model semantics and closed-world assumption for negation. This means that ConceptBase trades some expressive power for suitability as a deductive system.

Rendering MLT* in ConceptBase allows the operationalization of certain rules for multi-level modeling structures that were not defined before for DeepTelos. Thus, we aim for a combination that leverages the best of both worlds in the MLT-Telos implementation: a rich set of multi-level mechanisms and run-time deduction. Further, this exercise allows for a better understanding of the commonalities and differences of DeepTelos and MLT*. The attempt to unify two existing approaches to multi-level modeling contributes to reducing unnecessary diversity in this research domain.

Section 2 presents the foundations of MLT* and DeepTelos. Section 3 introduces MLT-Telos as an implementation of MLT* using the deductive capabilities of ConceptBase. Section 4 evaluates the implementation by showing how MLT* example models from the literature can be checked for errors and completed automatically. Section 5 shows MLT-Telos is a superset of DeepTelos. The paper concludes with a discussion of the results and future research.

2 FOUNDATIONS OF MLT* AND DEEPTELOS

Both MLT* (with its modeling language ML2) and DeepTelos are axiomatically defining extensions to the object-oriented modeling paradigms to better support multiple classification levels, such as objects, classes, meta-classes and so forth. Unlike the potency-based multi-level modeling approaches, MLT* and DeepTelos tackle the problem by additional logical axioms on instantiation and specialization.

2.1 MLT*

MLT* is defined by first-order logic axioms with classical semantics. In this logic, the axioms are used to make formal proofs of theorems that are then true in all interpretations that fulfill the axioms. MLT* is intended as a reference theory, establishing admissible multi-level structures that arise from the basic axioms and definitions. The theory is built up from a primitive instantiation relation.

The first axiom of MLT* states that any constant in any interpretation of MLT* is an entity[1]:

$$\forall x\ \mathsf{Entity}(x) \tag{1}$$

As a second example, types are defined as being those objects that have at least least one instance:

$$\forall x\ \mathsf{Type}(x) \leftrightarrow \exists y\ \mathsf{iof}(y, x) \tag{2}$$

The predicate $\mathsf{iof}(y, x)$ is true when y is an instance of x. The predicate $\mathsf{Type}(x)$ is true when x is a type, i.e. it has at least one possible instance. This rules out necessarily uninstantiated types. MLT* also defines a predicate "Individual" to qualify entities that have no instance (those that are not types). The predicate "Entity" thus subsumes "Type" and "Individual".

A central construct of MLT* is the powertype relation, which holds whenever all instances of a type t_1 (the so-called powertype) are specializations[2] of a base type t_2 (following Cardelli [6]):

$$\begin{aligned}\forall t_1, t_2\ \mathsf{isPowertypeOf}(t_1, t_2) \leftrightarrow \mathsf{Type}(t_1) \wedge \\ \forall t_3(\mathsf{iof}(t_3, t_1) \leftrightarrow \mathsf{specializes}(t_3, t_2))\end{aligned} \tag{3}$$

In total, MLT* comprises around 20 such axioms [1]. The other axioms describe categorizations (powertypes following Odell [22]), in particular partitions, and the difference between ordered types (those that can be strictly stratified into orders) and orderless types (those that defy stratification). There are also various theorems that are proven from the axiomatization (using model finders and automated theorem provers). For example, it follows that a (Cardelli) powertype is unique given a base type; that two types that partition the same base type cannot specialize each other, etc. The formally-proven theorems were shown to rule out problematic multi-level hierarchies in a realistic setting [4].

2.2 DeepTelos

DeepTelos [18] is based on the 30 axioms of O-Telos [14], which are a deductive Interpretation of the formalization of the knowledge representation language Telos [21]. The O-Telos axioms define and constrain the use of instantiation, specialization and attribution. Due the underpinning by O-Telos, the definition of DeepTelos only requires six axioms in its latest incarnation [16]. Reformulated with the predicates used in MLT*, the central axiom of DeepTelos is

$$\begin{aligned}\forall t_1, t_2, t_3\ \mathsf{iof}(t_1, \mathsf{Proposition}) \wedge \mathsf{iof}(t_2, \mathsf{Proposition}) \\ \wedge\ \mathsf{iof}(t_3, t_1) \wedge \mathsf{isPowertypeOf}(t_1, t_2) \rightarrow \mathsf{specializes}(t_3, t_2)\end{aligned} \tag{4}$$

There are however two important differences. First, all variables in formalization of O-Telos and DeepTelos are bound to finite extensions based on the deductive database consisting of a single base relation "Proposition", subsuming all objects, explicit attributes/links, explicit instantiations, and explicit specializations. The axioms of O-Telos and DeepTelos are then forming the intentional database using a fixpoint operator to compute the unique minimal Herbrand model [8]. Since the base relation is finite, all intentional (=derived) relations are finite as well. In the above formula, all variables must be bound to finite extensions, here the class "Proposition".

DeepTelos is implemented via deductive rules and integrity constraints in ConceptBase [13, 15]. With deductive rules, the right hand side of the rule (here $\mathsf{specializes}(t_3, t_2)$) stands for the concluded/derived predicate and the left hand side for the condition. While classical logic allows arbitrary re-ordering of predicates conforming to equivalences, this is not the case for deductive rules. Formula (3) in MLT* states the equivalence of a left hand side sub-formula to the right-hand side sub-formula, i.e. one can conclude in both directions. Such formulas are completely defining predicates such as "isPowertypeOf". In contrast, formula (4) derives new facts for the predicate "specializes" based on the left-hand side. There may well, be further deductive rules with the same right-hand side predicate "specializes". In DeepTelos, one declares facts of "isPowertypeOf" in the database and then formula (4) defines how new "specializes" facts are derived from it.

Besides deductive rules, ConceptBase also supports the definition of integrity constraints. Integrity constraints must be true in all database states. Technically, integrity constraints are transformed into a denial form deriving a predicate "inconsistent". The system makes sure that "inconsistent" has an empty extension at all times.

3 MLT-TELOS DEFINED WITHIN O-TELOS

The previous section exposed the main difference of MLT* and DeepTelos. MLT* is rooted in classic first-order logic allowing to prove theorems for any interpretation that satisfies the MLT* axioms. DeepTelos is based on ConceptBase which adopts Datalog-neg semantics where every program has a unique minimal Herbrand mode [8] computed by an efficient deductive database engine. DeepTelos is thus more suitable for analyzing actual models (instances of the DeepTelos constructs) by means of queries, rather than proving theorems. This section proposes MLT-Telos as a language to represent multi-level models using the MLT* constructs of instantiation, specialization, powertypes, and categorizations. The semantics of these constructs are expressed by rules and constraints on top of the O-Telos axioms (which are also expressed as rules and constraints). The purpose of this effort is to create an implementation of a multi-level modeling environment that *approximates* the original MLT* axioms and allows to analyze multi-level models for modeling errors via efficient Datalog engines. From a logic point of view, the minimal Herbrand model is just one of all possible interpretations in classical logic. However, if a multi-level

[1] Infinite interpretations are neither ruled out nor necessitated.
[2] The formal definition of "specializes" in MLT* [1] is shown later in section 3.2.2.

model is inconsistent[3] in its minimal Herbrand model, then it is also inconsistent in general. The deductive approach promoted by O-Telos and DeepTelos is therefore suitable to tackle the goal of this paper. O-Telos is the variant of Telos as implemented by ConceptBase. We thus shortly review the deductive features implemented in ConceptBase.

3.1 ConceptBase as implementation of O-Telos

ConceptBase stores all factual information as "propositions", i.e. quadruples of the form $P(o, x, n, y)$, where o is the system-generated identifier of the proposition, x is the source object identifier, n its label, and y the destination identifier. A proposition can represent named objects (and classes), explicit attributions and links, explicit instantiations, and explicit specializations. So, ConceptBase is effectively a deductive database with a single base relation $P(o, x, n, y)$.

Predicates for attributions, relations, instantiations and specializations are defined by deductive rules, some of them being defined in terms of the proposition predicate $P(o, x, n, y)$ and predicates based on $P(o, x, n, y)$. The most important predicates are:

- **(x in c)** The object x is an instance of the object c. This corresponds to the "iof" predicate of MLT*.
- **(c isA d)** The object c is a specialization or subclass of the object c. This corresponds to the "specializes" predicate of MLT*.
- **(x m y)** The objects x and y are linked by a relation m. The label "m" must be defined at some class of x.

The complete list of predicates and their precise definition is provided by the ConceptBase user manual [17]. O-Telos regards constants such as numbers and strings as objects. There is *ex ante* no difference between attributes and relations, though one can add extra rules and constraints to distinguish the two. Deductive rules and integrity constraints are described in a restricted first-order logic syntax, where all variables must be bound to classes with finite extensions. Subsequently, we use ConceptBase's textual syntax to denote deductive rules and constraints. For example, axiom (4) is written in O-Telos as a deductive rule:

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

"Proposition" is the most general class in O-Telos. It subsumes all objects [14]. Deductive rules must have a single conclusion predicate and all variables in this predicate must be universally quantified and bound to finite extensions in the rule body. Integrity constraints do not have this restriction. They are internally transformed to rules for deriving the "inconsistent" predicate. Besides rules and constraints, ConceptBase also supports query classes [17]. They are syntactically sugared-up representations of deductive rules. Safe negation is supported with the following restrictions: First all, variables in a negated predicate must be bound by a positive predicate in conjunction to the negated predicate. Second, if negated predicates occur in recursive deductive rules, then the rule set must be stratified [3] to avoid non-unique minimal models. We do not discuss this further since MLT-Telos does not contains such rules.

[3] A more precise statement is: The model violates at least one explicit integrity constraint. Logical consistency is a broader concept than deductive integrity constraint violation.

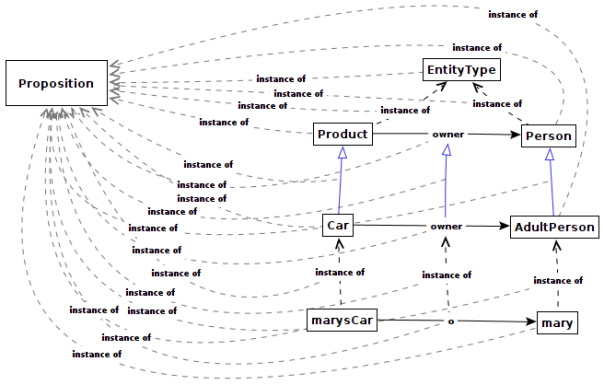

Figure 1: Example O-Telos model

Figure 1 shows a simple example of a model in O-Telos. The class "Proposition" has all explicit objects as instance, including classes. Refinement of attributes/relations is expressed by a specialization link, here between the owner relation of "Car" and the owner relation of "Product".

Model 1: O-Telos code for figure 1

```
EntityType end
Product in EntityType with attribute owner: Person end
Person in EntityType end
AdultPerson isA Person end
Car isA Product with attribute owner: AdultPerson end
mary in AdultPerson  end
marysCar in Car with owner o: mary end
```

The textual syntax for O-Telos code is a syntactic sugar for a list of facts such as `(Product in EntityType)`, `(AdultPerson isA Person)` and so forth. Details are explained in the user manual [17]. Note that the grey instantiation links to "Proposition" in figure 1 are derived by O-Telos axioms.

3.2 Definition of MLT-Telos

We follow the MLT* axiomatization described in [1] and discuss here the main design decisions for MLT-Telos since the mapping to O-Telos is not lossless.

3.2.1 The basic constructs of MLT.* MLT* defines "Entity" (any object), "Type" (objects that have instances) and "Individual" (objects that have no instances) by means of axioms. To avoid confusion with predefined O-Telos objects, we shall use all-caps labels in MLT-Telos for these concepts. All subsequent declarations are incremental. We use the ConceptBase frame syntax for declaring MLT-Telos objects. It is a more compact representation of facts expressed in the predicates `(x in c)`, `(c isA d)`, and `(x m y)`. The first step is to define classes for the three predefined types:

Listing 1: MLT-Telos predefined types

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

All three concepts are types[4], hence they are declared here as explicit instances of "TYPE". Types can be powertypes of other types and they can specialize other types:

Listing 2: Powertype and specialization

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

The attribute categories "irreflexive", "antisymmetric", "symmetric", and "transitive" come with predefined deductive rules and integrity constraints in ConceptBase. The same holds for the cardinality categories "single" (at most one filler), "revsingle" (at most one object has this value), and "necessary" (at least one filler). Transitivity is supported by deductive rules due to its minimal model semantics. For instance, the "specializes" relation is defined to be reflexive, antisymmetric and transitive. The rules and constraints associated to the attribute categories are made available at http://conceptbase.sourceforge.net/mlt-telos/SOURCES/SystemoHome.sml.

3.2.2 Mapping of MLT axioms.* Deductive rules and integrity constraints in Datalog-neg must be "range-restricted", i.e. all variables must be bound to finite extensions. For this reason, the MLT* definition of "Entity"

$$\forall x\ \mathsf{Entity}(x) \tag{5}$$

cannot be matched in MLT-Telos. Instead, MLT-Telos defines "Entity" to subsume all instances of "TYPE" and "INDIVIDUAL". These two classes shall have either explicit instances or instances derived by rules. At any point of type, the extension of "ENTITY" is finite in MLT-Telos, unlike as in MLT* which allows infinite interpretations. Subsequently, we follow the MLT* axiom definitions from [1]. The type "Individual" in MLT* is defined by axiom (6).

$$\forall x\ \mathsf{Individual}(x) \leftrightarrow \neg\exists y\ \mathsf{iof}(y, x) \tag{6}$$

In MLT-Telos, this is mapped to two constraints. The difference is not so much in the syntax but in the semantics. In MLT-Telos, any of the finitely many instances of "INDIVIDUAL" must fulfill the condition. Further, the variable "y" in the first constraint is bound to "ENTITY", which has a finite extension in MLT-Telos.

Listing 3: Individual objects

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

The MLT-Telos constraints may look syntactically similar to the corresponding MLT* axiom 6. However, the MLT* axiom makes a statement about all interpretations that satisfy the MLT* theory. The logical equivalence operator can also be read in both directions. In MLT-Telos, the constraint is checked only against the unique minimal Herbrand model of the current database[5]. The database consists of deductive rules, integrity constraints, and facts. The latter can also be viewed as deductive rules with no body. In MLT-Telos, instances of the class "INDIVIDUAL" need either to be declared explicitly, e.g. `(mary in Person)` or derived by deductive rules. In both cases, the number of instances is finite and all objects occur as constants in the database. We could add a rule like

```
forall x/Entity not exists y/ENTITY (y in x)
 ==> (x in INDIVIDUAL)
```

This rule would return all (finitely many) instances of "ENTITY" that currently have no instance. Such a rule would not be very useful for modeling purposes since a new definition of a type initially has no instance in the MLT-Telos database and would consequently be classified as an individual object.

The next MLT* axiom demands that each type must have at least one individual as transitive instance. The axiom uses the predicate "iof_trans" to denote the transitive closure of "iof":

$$\forall t\ \mathsf{Type}(t) \rightarrow \exists x\ \mathsf{Individual}(x) \wedge \mathsf{iof\_trans}(x, t) \tag{7}$$

In MLT-Telos, we first define "iof_trans" as the transitive closure of the explicit instantiation `:(x in t):` and then a so-called query class that returns all types that have no individual as transitive instance:

Listing 4: Transitive instances

```
ENTITY in Class with
  attribute,transitive iof_trans: ENTITY
  rule h_1: $ forall x/INDIVIDUAL t/TYPE :(x in t):
          ==> (x iof_trans t) $
end

TYPEwithoutINDIVIDUAL in QueryClass isA TYPE with
  constraint
    a_1: $ not exists x/INDIVIDUAL (x iof_trans this) $
end
```

The variable "this" in listing 4 stands for any instance of the superclass "TYPE" of the query. The definitions in listing 5 allow to temporarily have types in MLT-Telos that have no individual as transitive instance. This approach is suitable for modeling environment since a newly defined type initially has no extension in the minimal model semantics of MLT-Telos. Instances must be declared explicitly or must be derived from other explicit facts in the database.

The next two axioms defines the "specializes" and "properSpecializes" predicate in MLT*.

$$\forall t_1, t_2\ \mathsf{specializes}(t_1, t_2) \leftrightarrow\ (\mathsf{Type}(t_1) \wedge \forall e\ \mathsf{iof}(e, t_1) \rightarrow \mathsf{iof}(e, t_2)) \tag{8}$$

$$\forall t_1, t_2\ \mathsf{properSpecializes}(t_1, t_2) \leftrightarrow \mathsf{specializes}(t_1, t_2) \wedge t_1 \neq t_2 \tag{9}$$

In MLT-Telos, we only implement the direction from left to right, i.e. if a type specializes another type, then the supertype inherits the instances of the subtype. This is achieved by the following two deductive rules:

[4]We shall uses the terms type and classes as synonyms in MLT-Telos, because MLT-Telos types can only have a finite extension of explicit and derived instances.

[5]Datalog stems from the database discipline. A deductive database consists of rules, constraints (the latter are also mapped to rules) and facts (tuples in relational databases). Facts can be regarded as rules with no body.

Listing 5: Semantics of specializes

```
forall t1,t2/TYPE e/Proposition (t1 specializes t2) and
   (e in t1) ==> (e in t2)

forall t1,t2/TYPE (t1 specializes t2) and (t1 <> t2)
   ==> (t1 properSpecializes t2)
```

The range "Proposition" for the variable e is subsuming all stored objects in an MLT-Telos database. This is another case where MLT* allows infinite extensions and MLT-Telos only considers the finite minimal model semantics. The "specializes" predicate of MLT-Telos has virtually the same semantics as the pre-defined O-Telos predicate "isA". We still define it here because O-Telos forbids for technical reasons user-defined rules deriving "isA" relations. Having a dedicated "specializes" predicate allows controlling its definition independent from such restrictions.

$$\begin{aligned}\forall t_1, t_2\ \mathsf{Type}(t_1) \wedge \mathsf{Type}(t_2) \rightarrow\ & ((t_1 = t_2) \\ & \leftrightarrow \forall x\ (\mathsf{iof}(x, t_1) \\ & \leftrightarrow \mathsf{iof}(x, t_2))))\end{aligned} \tag{10}$$

Axiom (10) defines type equality based on the extension of the types. While two classes in O-Telos can be checked in whether they have the same extension in the minimal Herbrand model, the axiom cannot be translated since two objects in O-Telos are the same if and only if they are identical, i.e. have the same name. Consequently, axiom (10) is not supported in MLT-Telos.

The next MLT* axiom codifies the meaning of the "isPowerTypeOf" relation as discussed in the previous section.

$$\begin{aligned}\forall t_1, t_2\ \mathsf{isPowertypeOf}(t_1, t_2) \leftrightarrow\ & \mathsf{type}(t_1) \wedge \forall t_3(\mathsf{iof}(t_3, t_1) \\ & \leftrightarrow \mathsf{specializes}(t_3, t_2))\end{aligned} \tag{11}$$

In MLT-Telos, this axiom is implemented in the same way as for DeepTelos by deductive rules:

Listing 6: Semantics of isPowerTypeOf

```
forall t1,t2,t3,m/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)
```

The first deductive rule derives new specializations when a powertype is instantiated. The second derives instantiations to the powertype if the target type "t2" of the powertype is specialized. Note that axiom (11) uses the equivalence to *define* the "isPowerTypeOf" relation. All substitutions $[v_1/t_1, v_2/t_2]$ that satisfy the right-hand side

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

also satisfy $\mathsf{isPowertypeOf}(t_1, t_2)$. MLT-Telos only supports the direction from left to right and even more, only derives new solutions for the "specializes" predicate. The rationale for this restriction is that we assume that the modeler declares "isPowerTypeOf" facts rather than deriving them from a given instance model.

Basic types in MLT* are establishing a stratified order of types based on the instantiations and specialization predicates. The corresponding axioms can also not be mirrored exactly in O-Telos.

Listing 7: Approximation of BasicType

```
BasicType in QueryClass isA TYPE with constraint
    a_1: $ exists x/TYPE (x isPowerTypeOf this) or
             (this isPowerTypeOf x) $
end
```

Instead, we approximate the meaning by a query class that is solely based on "isPowerTypeOf". Note that the corresponding MLT* predicate is defined by an equivalence, while it is declared by explicit facts in MLT-Telos.

The powertype construct in MLT-Telos is the inverse as the most-general-instance construct in DeepTelos. DeepTelos does not constrain the cardinalities. MLT-Telos follows MLT* by imposing a 1:1 cardinality constraint for the powertype construct. This appears to be the more solid approach since it avoids undesired duplicate types that always have the same extension. Unlike DeepTelos, MLT* defines categorizations of types, such as disjoint and/or complete decomposition of types. These constructs are implemented in MLT-Telos by first declaring them and then defining their semantics by a combination of rules and constraints:

Listing 8: Categorization constructs

```
TYPE with single
     categorizes: TYPE;   disjointlyCategorizes: TYPE;
     partitions: TYPE;    completelyCategorizes: TYPE
end
```

The five deductive rules in listing 9 related the four categorizes constructs to each other and to the "specializes" relation. The last rule is not mentioned as a formal axiom in the MLT* specification but is described in words the written elaboration of categorization in [11]. It becomes a deductive rule in MLT-Telos to handle categorizations correctly.

Listing 9: Categorization semantics

```
forall t1,t2/TYPE (t1 disjointlyCategorizes t2)
  ==> (t1 categorizes t2)

forall t1,t2/TYPE (t1 completelyCategorizes t2)
  ==> (t1 categorizes t2)

forall t1,t2/TYPE (t1 partitions t2)
  ==> (t1 disjointlyCategorizes t2)

forall t1,t2/TYPE (t1 partitions t2)
 ==> (t1 completelyCategorizes t2)

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

The two integrity constraints in listing 10 relate categorizations and powertypes. The constraints demand that the complete categorization of a type must take place at the subtype level of the types participating in an "isPowerTypeOf" relation. The two constraints could also have been realized as deductive rules that derive the required subtype relations. However, we prefer encoding them as integrity constraints and thereby require the modeler to use the two constructs in the correct way. Defining them as deductive rules would potentially lead to cyclic subtype relations, which would be captured by the generic "antisymmetric" constraint of the "specializes" construct.

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)
```

Incorrect categorizations are analyzed via query classes. The first query class in listing 11 returns types that are declared as complete categorizations but that are in fact incomplete (there are instances of the type that are not instance of a proper subclass of the type). The second query class in listing 11 returns non-disjoint decompositions. The example in figure 3 shows how to use the queries during modeling. As discussed earlier, the use of query classes allows violations of the constraints by returning the "violators" as the result of the query. In a modeling environment one would only call the query when the model is considered to be complete. Violations during the modeling is not causing an integrity constraint violation.

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
```

The complete sources of MLT-Telos plus example models and further documentation are available from http://conceptbase.sourceforge.net/mlt-telos/.

4 MLT-TELOS BY EXAMPLE

The rules, constraints and queries listed in the previous section can directly be used in the ConceptBase system to define and analyze example MLT-Telos models. We use the standard examples of MLT* as provided in [11] to test the implementations and discuss its limitations.

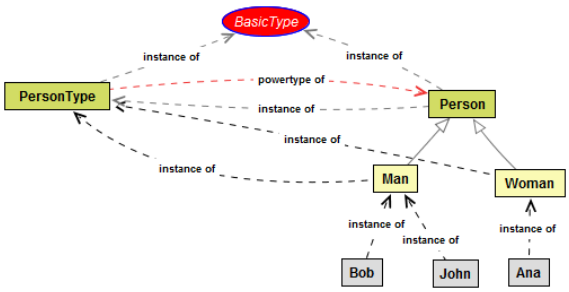

Figure 2: Simple powertype relation with basic types

The example in figure 2 declares a powertype relation between "PersonType" and "Person". The reverse instance-of relation is correctly derived[6] by MLT-Telos via the second rule of listing 6. The classes "Woman" and "Man" are declared as explicit instances of "PersonType". They are thus derived specializations of the class "Person" via the first MLT-Telos deductive rule of listing 6. The instances "Bob", "John" and "Ana" are declared as explicit instances of "Woman" and "Man", respectively, and are consequently derived instances of "Person". The query class "BasicType" classifies all types/classes that participate in powertype relations as discussed in the previous section. The source code of the example is in model 2. Note that the individuals need to be declared explicitly here because of the minimal model semantics of Datalog-neg. Likewise types need to be instantiated to the class "TYPE".

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
```

Figure 3 shows an example of the `partitions` construct. The classes "Woman" and "Man" are declared as explicit instances of "PersonByGender", which specializes "PersonType". Hence, "Woman" and "Man" are both derived instances of "PersonType", which is the powertype of "Person". Consequently, "Woman" and "Man" are then derived specializations of "Person".

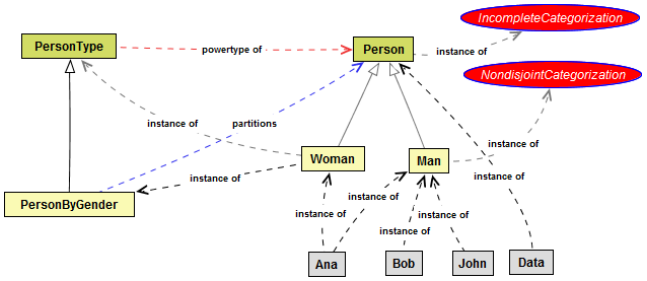

Figure 3: The partitions construct

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
```

The instances of "Woman" and "Man" are chosen to expose violations of both the disjointness and completeness criterion. The two criteria are realized as query classes "IncompleteCategorization" and "NondisjointCategorization". They are correctly exposing the

[6] Black links are explicit facts in the figures. Grey links are derived by rules. All figures are screendumps made with the ConceptBase user interface from graphical views of the model database.

violating types. For example, the answer to the query "NondisjointCategorization" is

```
Man in NondisjointCategorization with
  entity _: Ana  type _: Woman end
Woman in NondisjointCategorization with
  entity _: Ana  type _: Man end
```

There are two violators because both classes violate the criterion for disjoint decomposition. Query classes in ConceptBase are treated as classes: answers to the query are (derived) instances of the query. This explains the instantiation link to the query classes in figure 3.

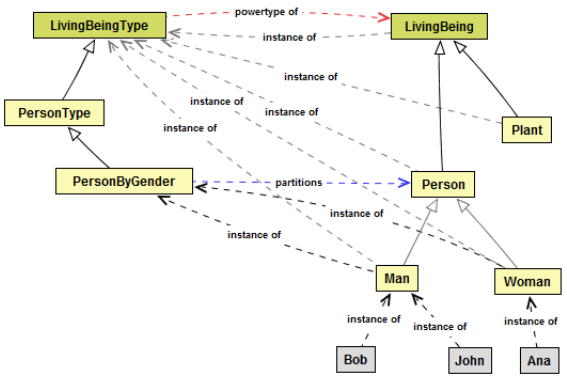

Figure 4: Partition under powertype

Figure 4 shows how to define a partition to a subclass of a powertype. The powertype relation is between "LivingBeingType" and "LivingBeing", whereas the partition relation is declared between the subclasses "PersonByGender" and "Person". The two classes "Woman" and "Man" are explicit instances of "PersonByGender". The partitions relations implies a categorization relations between "PersonByGender" and "Person", which then implies that "Woman" and "Man" are specializations of Person. The MLT-Telos source code of the example in figure 4 is:

Model 4: Listing for figure 4

```
LivingBeing in TYPE end
LivingBeingType in TYPE with isPowerTypeOf type: LivingBeing end
PersonType in TYPE with specializes t1: LivingBeingType end
Plant in TYPE with specializes t1: LivingBeing end
Person in TYPE with specializes t1: LivingBeing 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,Woman end     Data in INDIVIDUAL,Person end
```

Figure 4 shows that the MLT-Telos implementation of MLT* can derive instantiations to the powertype from explicit specializations. For example "Person" and "Plant" are explicit specializations of "LivingBeing". Their instantiation to the powertype "LivingBeingType" is derived via the second rule in listing 6. Compare in contrast the example in Figure 2. There, "Man" and "Woman" are explicit instances of "PersonType" and their specialization to "Person" is derived via the first rule of listing 6. The source code in model 4 corresponds to the explicit facts in the database. All other facts are derived by rules of O-Telos and MLT-Telos. The high ratio of derived facts versus explicit facts confirms that MLT-Telos supports the modeler in creating concise and redundancy-free models. Derived facts do not need to be declared.

MLT* supports types with more than one supertype. Figure 5 shows that this feature is also available in MLT-Telos. Here, the object "John" is instance of "AdultMan" and via inheritance of "Adult" and "Man". "AdultMan" is a derived instance of "PersonType" via the second rule for the "isPowerTypeOf" construct in listing 6. The classes "Man" and "Adult" are in contrast explicit instances of the powertype "PersonType". Hence, their specialization to "Person" is derived via the first rule in listing 6. MLT* distinguishes ordered types (types that are specializing basic types) from order-less types (all other types). Order-less types are then simply all types that are not ordered types.

Figure 6 shows the instances of "OrderedType". The types "2ndOT", "1stOT" and "INDIVIDUAL". are instances of "BasicType". MLT-Telos approximates "OrderedType" by a query class:

```
OrderedType in QueryClass isA TYPE with constraint
    d_8: $ exists b/BasicType (this specializes b) $
end
```

Arguably, this is only a roughly approximation of the MLT* concept of ordered types since the query class does not demand that the chain of powertype relations ends in "INDIVIDUAL".

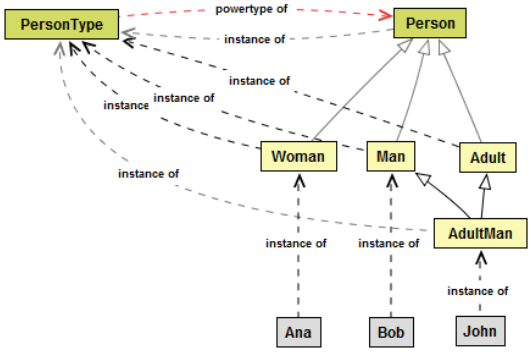

Figure 5: Multiple generalization in MLT-Telos

Model 5: Listing for figure 5

```
Person in TYPE end
PersonType in TYPE with isPowerTypeOf type: Person end
Man in TYPE,PersonType end
Woman in TYPE,PersonType end
Adult in TYPE,PersonType end
AdultMan in TYPE with specializes type1: Man; type2: Adult end
John in INDIVIDUAL,AdultMan end
Bob in INDIVIDUAL,Man end
Ana in INDIVIDUAL,Woman end
```

Having "INDIVIDUAL" as end of the powertype chain removes the need to explicitly declare an object as an instance of "INDIVIDUAL" if the class of the object is an instance of "1stOT". For example, the object "John" in model 5 is declared as an explicit

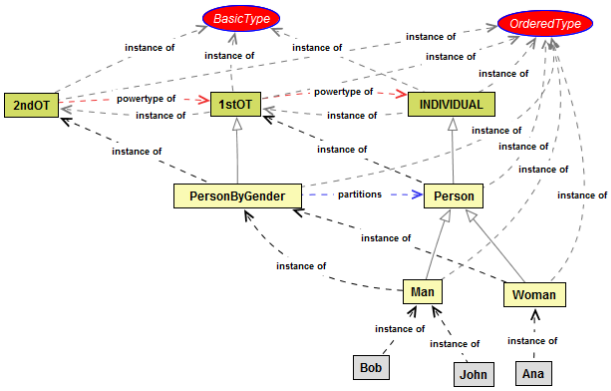

Figure 6: Ordered types

instance of "Man". Via specialization, it is then a derived instance of "INDIVIDUAL". This completes the pure MLT-Telos examples. We have shown that all MLT* constructs could be represented. The concepts of basic types and ordered types are only approximated due to the limited power of Datalog-neg compared to first-order logic. Some of the definitions of MLT* are stated as logical equivalences, see for example the axiom (11) for "isPowerTypeOf". The original MLT* axiom allows to derive "isPowerTypeOf" facts by reading the formula from right to left. In MLT-Telos, we only support the direction from left to right, i.e. "specializes" facts are derived from "isPowerTypeOf" facts.

Model 6: Listing for figure 6

```
1stOT in TYPE with isPowerTypeOf type1: INDIVIDUAL end
2ndOT in TYPE with isPowerTypeOf type1: 1stOT end
PersonByGender in TYPE, 2ndOT with partitions type: Person end
Person in TYPE, 1stOT end
Man in TYPE,PersonByGender with end
Woman in TYPE,PersonByGender with end
John in Man end    Bob in Man end    Ana in Woman end
```

All figures in this paper are screendumps of the ConceptBase user interface displaying graphical views on the models stored in the ConceptBase database. O-Telos treats classes as objects. So, the database also contains the O-Telos class definitions of MLT-Telos as presented in the listings of this paper. The graphical user interface was configured to use the graphical symbols familiar with UML with adaptions for MLT*.

5 MLT-TELOS VERSUS DEEPTELOS

This research was started because the authors wanted to understand how DeepTelos and MLT* are related, being both based on the powertype pattern. We tackled the problem by implementing MLT* in the same way how DeepTelos was implemented, i.e. as extension of O-Telos within the Datalog-neg framework provided by ConceptBase. DeepTelos has just five rules and one constraint. So, is MLT-Telos a proper super-set of DeepTelos? We answer this question by mapping DeepTelos models to MLT-Telos models via a few deductive rules.

MLT-Telos uses the dedicated class "TYPE" to define the constructs "isPowerTypeOf" and "specializes". The corresponding DeepTelos constructs are defined at the omega class "Proposition":

```
Proposition with
  attribute ISA: Proposition; IN: Proposition
end
```

This has the advantage that any object can use them while MLT-Telos requires the objects to be instances of "TYPE". Section 2 found that the MLT* construct "isPowerTypeOf" corresponds to the inverse of the DeepTelos construct "IN", and the MLT* construct "specializes" can be identified with the DeepTelos construct "ISA". The deductive rules in listing 12 realize a mapping of a DeepTelos model to MLT-Telos.

Listing 12: Rules for mapping DeepTelos to MLT-Telos

```
forall x,y/TYPE (x IN y) ==> (y isPowerTypeOf x)
forall x,y/TYPE (x ISA y) ==> (x specializes y)

forall x,y/Proposition (x IN y) ==> (x in TYPE)
forall x,y/Proposition (x IN y) ==> (y in TYPE)
forall x,y/Proposition (x ISA y) ==> (x in TYPE)
forall x,y/Proposition (x ISA y) ==> (y in TYPE)
forall x,y/TYPE z/Proposition (x IN y) and (z in y)
  ==> (z in TYPE)
forall x/Proposition y/TYPE (x isA y) and (x <> y) and
   not (x in QueryClass) ==> (x in TYPE)
```

The first two rules map the constructs from DeepTelos to MLT-Telos. The remaining rules make sure that the participating objects are instance of the MLT-Telos class "TYPE". The predicate "isA" is the regular O-Telos specialization. Query classes need to be excluded for technical reasons only.

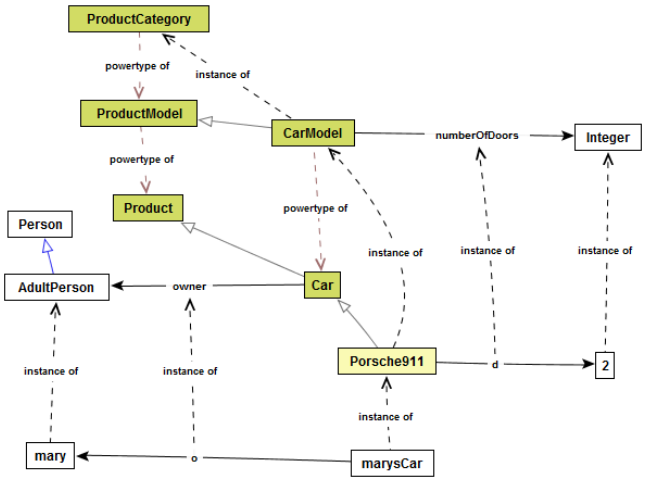

Figure 7: DeepTelos model interpreted as MLT-Model

Figure 7 shows the running DeepTelos example used in [18]. The powertype relations in the model are now derived from the rules of listing 12 based on the facts of example 5. Note that the attributes and relations defined at the classes "CarModel" and "Car" can be used for the instances "Porsche911" and "marysCar", respectively. The mapping rules do not cover the mapping of objects to the MLT-Telos class "INDIVIDUAL". This is a certain limitation but is not

harmful for this example model since DeepTelos has no notion of an individual object. The DeepTelos code for figure 7 is given in model 7.

Model 7: DeepTelos code for figure 7

```
ProductCategory end
ProductModel with IN c: ProductCategory end
Product with IN c: ProductModel end
CarModel in ProductCategory end   Car with IN c :CarModel end
Person end                        ProductCategory end
ProductModel with  attribute listPrice: Integer end
CarModel with attribute numberOfDoors: Integer end
Car with attribute owner: AdultPerson end
AdultPerson isA Person end
Product with attribute owner: Person end

Porsche911 in CarModel,Class with numberOfDoors d: 2 end
mary in AdultPerson  end
marysCar in Porsche911 with owner o: mary end
```

A limitation of DeepTelos and MLT-Telos is the lack if integration of the "ISA" and "specializes" constructs with the O-Telos "isA". The latter is defined and constrained by a number of O-Telos axioms, in particular about refining attributes and relations for subclasses. Since the axioms for the O-Telos "isA" predicate are hard-coded into the ConceptBase system, a full integration of the specialization predicates requires some changes to the ConceptBase code.

6 DISCUSSION

This paper has investigated how MLT* and DeepTelos are related. The investigation started by comparing the main constructs of the two approaches. We then mapped the MLT* axioms to a set of MLT-Telos rules, constraints, and queries. The resulting specification was tested with standard MLT* examples. Finally, DeepTelos was mapped to MLT-Telos to test whether it is a proper subset. The results are as follows:

(1) MLT* and DeepTelos are both based on Cardelli's powertype construct [6].
(2) MLT* uses first-order predicate logic with classical interpretation to completely define the semantics of this multi-level approach. In contrast, DeepTelos constrains the construct by deductive rules and integrity constraints that are then interpreted by the unique minimal Herbrand model.
(3) MLT* can be used to prove theorems on multi-level modeling satisfied by all models that satisfy the theory. MLT-Telos can check a given model on whether it violates the constraints.
(4) MLT-Telos is an approximation of MLT* by rules, integrity constraints and queries. All MLT* constructs were covered. The definition of the predicates "isPowerTypeOf", "specializes" and "categorizes" is incomplete by only realizing one direction of the MLT* equivalences. Specifically, the "isPowerTypeOf" relation is declared in MLT-Telos, not defined as in MLT*.
(5) The examples from the standard MLT* papers were encoded in MLT-Telos. MLT-Telos can detect constraint violations such as incomplete or non-disjoint decompositions.
(6) The mapping rules for DeepTelos expose DeepTelos as a proper subset of MLT-Telos. DeepTelos models can directly be viewed as MLT-Telos models, but not vice versa.
(7) The 1:1 cardinality constraint for "isPowerTypeOf" in MLT* should also be applied to DeepTelos to exclude redundant classes that always have the same extension.
(8) DeepTelos showed that it can cover the standard multi-level modeling challenges [16]. Since DeepTelos is a proper subset, MLT-Telos and MLT* can easily pass the same challenges.

We did not yet discuss the issue of linguistic versus ontological instantiation [19]. One can argue that the classes "ENTITY", "INDIVIDUAL", and "TYPE" make up the linguistic level in MLT-Telos. Historically, O-Telos almost exclusively used the linguistic instantiation for creating domain-specific languages. Since MLT-Telos extends O-Telos, both usages of instantiation are available and can be mixed, see also [18].

6.1 Implementation Considerations

The implementation of MLT-Telos revealed some limitations of the ConceptBase system. First, the inability to create deductive rules deriving the O-Telos specialization predicate "isA" limits the extent to which the MLT-Telos predicate "specializes" can be used for refining attributes at subclasses. This restriction is due to the legacy implementation of the "isA" by software code rather than user-defined deductive rules. A solution requires adaptions to the ConceptBase code. A second restriction is on the rule compiler of ConceptBase. If a rule contains predicates `(x in c)` with a variable "c", then certain existentially quantified sub-formulas are not compiled. Since all rules are range-restricted [5] and can be mapped to Datalog-neg [20], this restriction can be overcome by generating auxiliary rules for the subformulas.

Instead of the ConceptBase system, we could also have used another Datalog engine such as DLV [2]. DLV uses Prolog-style clauses for deductive rules, augmented by disjunctive heads. ConceptBase offers a first-order syntax for deductive rules and integrity constraints and internally translates them to Datalog rules. A pragmatic reason for ConceptBase is the support of the O-Telos axioms that are the basis for both MLT-Telos and DeepTelos. A certain disadvantage of ConceptBase is the cumbersome way to define predicates other than the instantiation, specialization and attribution/relation predicates.

ConceptBase implements an incremental integrity checker based on the simplification method [5]. This method partially evaluates rules and constraints at compile time with patterns for insertions and deletions of facts. A similar technique is employed to partially evaluate formulas that have variables at the class position of instantiation predicates. The latter technique is also needed to provide an effective stratification test for complex recursive rule sets. MLT-Telos is an example of such a complex rules set.

6.2 Future Work

The focus of this paper was to compare two powertype-based approaches to multi-level modeling. A thorough comparison of powertype-based approaches with potency-based approaches, such as [10, 12], is subject to future work. Powertype-based methods create instantiation levels via chains of powertype links. Hence, the abstraction levels of types and individuals can be derived from their relation to such chains. An open question is how to translate potencies of attributes and relations from a powertype-based

multi-level model, and how to map a potency-based model to a powertype-based model.

MLT* does not explicitly define how to handle attributes and relations of multi-level objects (despite some treatment of those in the first version of the theory [7]). In traditional conceptual modeling languages, attributes and relations are distinguished from objects (entities). O-Telos and DeepTelos materialize attribute and relations, i.e. explicit attributes and relations are objects. Consequently, DeepTelos allows to specify powertype relations between attributes and relations. Does it make sense to define this feature in MLT* as well, and then pass it over to MLT-Telos?

We plan to apply MLT-Telos to a greater number of multi-level models to test its ability to identify modeling errors, in particular for industrial product and process modeling. Additional queries may be added to MLT-Telos to identify sub-optimal model fragments. On the implementation side, a full integration of the specialization constructs of DeepTelos with the O-Telos specialization is desirable. Further, we plan to test the approach with large multi-level models to evaluate the performance of the deductive implementation in ConceptBase.

7 CONCLUSIONS

MLT-Telos is an implementation of MLT* that allows to identify modeling errors via integrity constraint violations. MLT-Telos is able to leverage MLT* rules to automatically derive a number of elements of the multi-level model. As shown in this paper, a significant number of elements of the multi-level model can be derived rather than explicitly defined. This deductive capability can be employed to support the designer in an intelligent multi-level modeling environment. We have shown that MLT-Telos is a proper *superset* of DeepTelos. It realizes the powertype constructs in the same way as DeepTelos but adds the categorization construct. The definition of MLT-Telos can be included in a ConceptBase database, resulting in a (limited, but focused) implementation of MLT*. The sources of MLT-Telos and all example models are made available at http://conceptbase.sourceforge.net/mlt-telos/.

ACKNOWLEDGMENTS

João Paulo A. Almeida is funded by CNPq (grants 312123/2017-5 and 407235/2017-5) and CAPES (23038.028816/2016-41). Manfred A. Jeusfeld is partially funded by KK Stiftelsen (grant VF-KDO, HIS DNR 20180011).

REFERENCES

[1] João Paulo A. Almeida, Claudenir M. Fonseca, and Victorio Albani de Carvalho. 2017. A Comprehensive Formal Theory for Multi-level Conceptual Modeling. In *Conceptual Modeling - 36th International Conference, ER 2017, Valencia, Spain, November 6-9, 2017, Proceedings (Lecture Notes in Computer Science)*, Heinrich C. Mayr, Giancarlo Guizzardi, Hui Ma, and Oscar Pastor (Eds.), Vol. 10650. Springer, 280–294. https://doi.org/10.1007/978-3-319-69904-2_23

[2] Mario Alviano, Wolfgang Faber, Nicola Leone, Simona Perri, Gerald Pfeifer, and Giorgio Terracina. 2010. The Disjunctive Datalog System DLV. In *Datalog Reloaded - First International Workshop, Datalog 2010, Oxford, UK, March 16-19, 2010. Revised Selected Papers (Lecture Notes in Computer Science)*, Vol. 6702. Springer, 282–301. https://doi.org/10.1007/978-3-642-24206-9_17

[3] Krzysztof R. Apt, Howard A. Blair, and Adrian Walker. 1988. Towards a Theory of Declarative Knowledge. In *Foundations of Deductive Databases and Logic Programming*, Jack Minker (Ed.). Morgan Kaufmann, 89–148. https://doi.org/10.1016/b978-0-934613-40-8.50006-3

[4] Freddy Brasileiro, João A. Almeida, Victorio A. Carvalho, and Giancarlo Guizzardi. 2016. Applying a Multi-Level Modeling Theory to Assess Taxonomic Hierarchies in Wikidata. In *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, 975–980. https://doi.org/10.1145/2872518.2891117

[5] François Bry, Hendrik Decker, and Rainer Manthey. 1988. A Uniform Approach to Constraint Satisfaction and Constraint Satisfiability in Deductive Databases. In *Advances in Database Technology - EDBT'88, Proc. International Conference on Extending Database Technology, Venice, Italy, March 14-18, 1988 (Lecture Notes in Computer Science)*, Joachim W. Schmidt, Stefano Ceri, and Michele Missikoff (Eds.), Vol. 303. Springer, 488–505. https://doi.org/10.1007/3-540-19074-0_69

[6] Luca Cardelli. 1988. Structural Subtyping and the Notion of Power Type. In *Conference Record of the Fifteenth Annual ACM Symposium on Principles of Programming Languages, San Diego, California, USA, January 10-13, 1988*, Jeanne Ferrante and P. Mager (Eds.). ACM Press, 70–79. https://doi.org/10.1145/73560.73566

[7] Victorio A. Carvalho and João Paulo A. Almeida. 2018. Toward a well-founded theory for multi-level conceptual modeling. *Software and Systems Modeling* (2018), 1–27. https://doi.org/10.1007/s10270-016-0538-9

[8] Stefano Ceri, Georg Gottlob, and Letizia Tanca. 1989. What you Always Wanted to Know About Datalog (And Never Dared to Ask). *IEEE Trans. Knowl. Data Eng.* 1, 1 (1989), 146–166. https://doi.org/10.1109/69.43410

[9] Juan de Lara and Esther Guerra. 2010. Deep Meta-modelling with MetaDepth. In *Objects, Models, Components, Patterns, 48th International Conference, TOOLS 2010, Málaga, Spain, June 28 - July 2, 2010. Proceedings (Lecture Notes in Computer Science)*, Jan Vitek (Ed.), Vol. 6141. Springer, 1–20. https://doi.org/10.1007/978-3-642-13953-6_1

[10] Juan de Lara, Esther Guerra, Ruth Cobos, and Jaime Moreno-Llorena. 2014. Extending Deep Meta-Modelling for Practical Model-Driven Engineering. *Comput. J.* 57, 1 (2014), 36–58. http://dx.doi.org/10.1093/comjnl/bxs144

[11] Claudenir M. Fonseca, João Paulo A. Almeida, Giancarlo Guizzardi, and Victorio Albani de Carvalho. 2018. Multi-level Conceptual Modeling: From a Formal Theory to a Well-Founded Language. In *Conceptual Modeling - 37th International Conference, ER 2018, Xi'an, China, October 22-25, 2018, Proceedings (Lecture Notes in Computer Science)*, Juan Trujillo, Karen C. Davis, Xiaoyong Du, Zhanhuai Li, Tok Wang Ling, Guoliang Li, and Mong-Li Lee (Eds.), Vol. 11157. Springer, 409–423. https://doi.org/10.1007/978-3-030-00847-5_29

[12] Ulrich Frank. 2014. Multilevel Modeling - Toward a New Paradigm of Conceptual Modeling and Information Systems Design. *Business & Information Systems Engineering* 6, 6 (2014), 319–337. https://doi.org/10.1007/s12599-014-0350-4

[13] Matthias Jarke, Rainer Gallersdörfer, Manfred A. Jeusfeld, Martin Staudt, and Stefan Eherer. 1995. ConceptBase - A Deductive Object Base for Meta Data Management. *J. Intell. Inf. Syst.* 4, 2 (1995), 167–192. http://dx.doi.org/10.1007/BF00961873

[14] Manfred A. Jeusfeld. 2005. Complete List of O-Telos Axioms. Online: http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/d1228997/O-Telos-Axioms.pdf.

[15] Manfred A. Jeusfeld. 2009. Metamodeling and method engineering with ConceptBase. In *Metamodeling for Method Engineering*, Manfred A. Jeusfeld, Matthias Jarke, and John Mylopoulos (Eds.). MIT Press, 89–168.

[16] Manfred A. Jeusfeld. 2019. DeepTelos for ConceptBase: A Contribution to the MULTI Process Challenge. In *22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS Companion 2019, Munich, Germany, September 15-20, 2019*. IEEE, 66–77. https://doi.org/10.1109/MODELS-C.2019.00016

[17] Manfred A. Jeusfeld. 2020. ConceptBase.cc User Manual - Version 8.1. Online: http://conceptbase.sourceforge.net/userManual81/.

[18] Manfred A. Jeusfeld and Bernd Neumayr. 2016. DeepTelos: Multi-level Modeling with Most General Instances. In *Conceptual Modeling - 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016, Proceedings*. 198–211. https://doi.org/10.1007/978-3-319-46397-1_15

[19] Arne Lange and Colin Atkinson. 2018. Multi-level modeling with MELANEE. In *Proceedings of MODELS 2018 Workshops co-located with ACM/IEEE 21st International Conference on Model Driven Engineering Languages and Systems (MODELS 2018), Copenhagen, Denmark, October, 14, 2018 (CEUR Workshop Proceedings)*, Regina Hebig and Thorsten Berger (Eds.), Vol. 2245. CEUR-WS.org, 653–662. http://ceur-ws.org/Vol-2245/multi_paper_3.pdf

[20] John W. Lloyd and Rodney W. Topor. 1984. Making Prolog more Expressive. *J. Log. Program.* 1, 3 (1984), 225–240. https://doi.org/10.1016/0743-1066(84)90011-6

[21] John Mylopoulos, Alexander Borgida, Matthias Jarke, and Manolis Koubarakis. 1990. Telos: Representing Knowledge About Information Systems. *ACM Trans. Inf. Syst.* 8, 4 (1990), 325–362. https://doi.org/10.1145/102675.102676

[22] James J. Odell. 1998. *Advanced object-oriented analysis and design using UML*. Cambridge University Press, Chapter Power types, 23–32.

[23] Alain Pirotte, Esteban Zimányi, David Massart, and Tatiana Yakusheva. 1994. Materialization: A Powerful and Ubiquitous Abstraction Pattern. In *VLDB'94, Proc. 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo (Eds.). Morgan Kaufmann, 630–641. http://www.vldb.org/conf/1994/P630.PDF