Dual Deep Instantiation
and Its ConceptBase Implementation

Bernd Neumayr\textsuperscript{1}, Manfred A. Jeusfeld\textsuperscript{2}, Michael Schrefl\textsuperscript{1}, and Christoph Schütz\textsuperscript{1}

\textsuperscript{1} Johannes Kepler University Linz, Austria
\texttt{\{neumayr,schrefl,schuetz\}@dke.uni-linz.ac.at}
\textsuperscript{2} University of Skövde, Sweden
\texttt{manfred.jeusfeld@acm.org}

Abstract. Application integration requires the consideration of instance data and schema data. Instance data in one application may be schema data for another application, which gives rise to multiple instantiation levels. Using deep instantiation, an object may be deeply characterized by representing schema data about objects several instantiation levels below. Deep instantiation still demands a clear separation of instantiation levels: the source and target objects of a relationship must be at the same instantiation level. This separation is inadequate in the context of application integration. Dual deep instantiation (DDI), on the other hand, allows for relationships that connect objects at different instantiation levels. The depth of the characterization may be specified separately for each end of the relationship. In this paper, we present and implement set-theoretic predicates and axioms for the representation of conceptual models with DDI.

Keywords: Conceptual Data Modeling, Metamodeling, Deep Characterization, Powertypes.

1 Introduction

Conceptual modeling for application integration and the accompanying data interchange must consider both schema data, such as classes and associations, and instance data, such as objects and links. What is represented as instance data in one application may be represented as schema data in another application. For example, in an application for managing a mobile phone catalog, a class \texttt{MobilePhone} is instantiated by objects representing particular mobile phone models, such as \texttt{SamsungGS4}. In an application for managing an inventory of mobile phones, the same phone model may be represented by a class \texttt{SamsungGS4} which is instantiated by objects representing individual mobile phones, such as \texttt{SarahsPhone}. In a compact and integrated representation, a single model element, e.g., \texttt{SamsungGS4}, should represent both: an individual object, e.g., the mobile phone model \texttt{SamsungGS4}, and a class, e.g., individual phones of model \texttt{SamsungGS4}.
Such a schema-instance duality is inherent to conceptual models with Deep Instantiation (DI) \[4\]. DI extends the traditional notion of instantiation with the concept of potency: each attribute of a class may have a potency that specifies at how many instantiation levels below the class the attribute is to be instantiated. For example, in Fig. 1 MobilePhone has an attribute serialNr with a potency of 2 which is to be instantiated by the instantiations, e.g., SarahsPhone, of its instantiations, e.g., SamsungGS4.

A schema-instance duality is also inherent to relationships in conceptual models with DI. For example, a cpu relationship between phone model SamsungGS4 and processor model Cortex-A15 is, first, an instantiation of the cpu relationship between Processor and MobilePhone and, second, acts as schema for relationships between individual phones, such as SarahsPhone, and individual processors of type Cortex-A15.

The flexibility, however, of modeling with DI is restricted by the tenets of strict metamodeling: DI only allows relationships between objects at the same level \[3, 4, 5, 9\]. A common problem with strict metamodeling is illustrated by the following situation which cannot be represented. Every instantiation of MobilePhone has a designer being a person. Likewise, every instantiation of an instantiation of MobilePhone, e.g., SarahsPhone, has an owner which is also a person.

Dual Deep Instantiation (DDI) lifts the restrictions of strict metamodeling. DDI, unlike traditional DI, distinguishes between source potency and target potency. DDI relationships may connect objects at different instantiation levels. Every relationship has both a source and target potency to separately indicate the depth of characterization for both sides of the relationship. For example, MobilePhone relates to Person, first, via a relationship owner with a source potency of 2 and a target potency of 1, and, second, via a relationship designer with a source potency of 1 and a target potency of 1. Relationship owner is to be instantiated between instantiations of instantiations of MobilePhone, such as SarahsPhone, and instantiations of Person. Relationship designer is to be instantiated between instantiations of MobilePhone, such as SamsungGS4, and instantiations of Person.

This paper introduces a concise set-theoretic formalization of “basic” DDI (Sect. 2). Section 3 extends basic DDI by object specialization resulting in the full axiomatic specification of DDI. The running mobile phone example motivates the constructs throughout sections 2 and 3. The implementation (Sect. 4) shows the satisfiability of the DDI axioms under the Datalog semantics of ConceptBase \[12, 14\]. We have created a collection of sample DDI models that show the absence of
redundancy in the DDI axiom system. The implementation also shows that the consistency checking for DDI models can be automated at run time. Section 5 reviews related work before we conclude with a summary.

2 Basics of Dual Deep Instantiation

We first discuss the setting in which DDI is introduced and then formally describe the modeling primitives of DDI, auxiliary predicates, axioms checking well-formedness of DDI models, and consistent instantiation. Table 1 contains the full DDI specification, which we elaborate in the text by referring to the line numbers (No. 1 – No. 31), with the part relating to specialization (No. 17 – No. 27) being explained in Sect. 3. The formal description is illustrated by a sample DDI model depicted in Fig. 2. We also discuss criticism of the DI approach and relate DI and DDI to modeling using the powertype pattern.

DDI is introduced in a setting that extends the following class-instance model (of the ER model or UML) to arbitrary instantiation levels:

1. Entity sets (classes) are related by binary relationship sets (associations), where one end of the relationship is generally referred to as source and the other end as target role. For conciseness of presentation, we omit role names and depict/denote sources at the left and targets at the right hand side.

2. Entities (instances) of entity sets are related by relationship occurrences (links), whereby a corresponding relationship set must exist between their entity sets, i.e., a relationship set \( r \) between two entity-sets \( x \) and \( y \) acts - considered as function and inverse - as two referential integrity constraint, a domain constraint for the source and a range-constraint for the target.

3. Relationships occurrences (links) and relationship sets (associations) carry a label, which is the same for the relationship occurrence and the relationship set to which it belongs.

4. Values are treated as objects, and consequently, attributes as relationships.

![Diagram](image_url)  
**Fig. 2.** Modeling a Mobile Phone Catalog and Inventory using DDI
A DDI model contains a set of objects, $O$ (No. 1), which are organized in instantiation hierarchies with an arbitrary number of instantiation levels, where $in(x, y)$ expresses that $x$ is an instantiation of $y$ (No. 5). For sake of the later extension to specialization, we introduce for the time being member and isa as synonym to in (No. 9 and No. 11). Object $x$ is called an $n$-member of $y$ if $x$ is gained by $n$ times of instantiation from $y$, or it is a 0-member of $y$ if $x = y$ (No. 10). We synonymously say $x$ is at the $n$-th instantiation level of $y$ if $x$ is a $n$-member of $y$ and we say $x$ is a $i..j$-member of $y$ if $x$ is a $n$-member of $y$ with $i \leq n \leq j$, and $x$ is a *-member of $y$ if $x$ is an $n$-member of $y$ with an arbitrary $n$ (‘$x$ is a *-member of $y$’ is equivalent to $isa^*(x, y)$).

Objects in DDI combine aspects of classes and of instances. An object acts as hook for the definition of instance data and schema data about objects in the instantiation subtree rooted in the object, tying together descriptions about objects at different instantiation levels. Objects further down in the instantiation tree instantiate and specialize these data. In this regard, instantiation hierarchies combine aspects of classification hierarchies and of generalization hierarchies.

Example 1 (Objects). Object SarahsPhone (see Fig. 2) is an instantiation of SamsungGS4 which is an instantiation of MobilePhone. SamsungGS4 is at the first instantiation level (is a 1-member) of MobilePhone and SarahsPhone is at the second instantiation level (is a 2-member) of MobilePhone. Now, looking at the instantiation hierarchy rooted in object ExchangeableTender: object Euro is an instantiation of ExchangeableTender and represents the Euro currency, a particular type of exchangeable tender, object MEuro represents a particular unit of Euro and is at the second instantiation level of ExchangeableTender, and, finally, object 137inUnit-MEuro is an instantiation of MEuro and represents a particular value in a particular unit of the Euro currency and is at the third instantiation level of ExchangeableTender.

We will now discuss the kinship between deep instantiation and powertypes [7], not taking into account DDI relationships and, thus, talking of DI objects instead of DDI objects; we will come back to DDI relationships and powertypes at the end of this section. We accentuate the kinship between DI and powertypes by giving a label to each instantiation level to indicate that objects at
some instantiation level are in a 1-to-1 relationship to instances of the class related to that level. For example (see Fig. 3), the instantiation levels of ExchangeableTender are labeled Type, Unit, and Value. These levels are related to classes arranged in a cascaded powertype pattern: TypeOfExchangeableTender is powertype of UnitOfExchangeableTender which in turn is powertype of ValueOfExchangeableTender. The DI object ExchangeableTender can be regarded as a multi-faceted construct that represents these three classes. Likewise, instantiation between DI objects is multi-faceted. For example, the instantiation of DI object ExchangeableTender to Euro represents the instance-of relationship between Euro and TypeOfExchangeableTender as well as the specialization of UnitOfExchangeableTender and ValueOfExchangeableTender to UnitOfEuro and EuroValue, respectively. Note: labels and sortals associated with levels are outside the set-theoretic formalization of DDI and are thus omitted in Table 1.

Coming back to the set-theoretic formalization of DDI: We impose two restrictions on object instantiation. First, acyclic instantiation (No. 12) is a basic assumption of the DDI approach. Second, the restriction to single instantiation (No. 13) is mainly due to space limitations. The extension to multiple instantiation/inheritance is ongoing research.

Relationships in DDI combine aspects of relationship occurrences (links) and relationship sets (associations). An $a^{i\to j}$ relationship (where $i$ is the source potency, $j$ the target potency, and $a$ the relationship label) between source object $x$ and target object $y$ is a multi-faceted construct. It represents, first, a link between $x$ and $y$, and, second, can be further instantiated between $0..i$-members of $x$ in the source role and $0..j$-members of $y$ in the target role, and represents, in this regard, a set of multiple associations, one association for each combination of possible source and target instantiation levels. An $a^{0\to 0}$ relationship cannot be further instantiated. We say, a relationship between source object $x$ and target object $y$ instantiates a relationship between source object $c$ and target object $d$ if they have the same relationship label and $x$ is a *-member of $c$ and $y$ is a *-member of $d$.

Relationships at more abstract instantiation levels implicitly impose domain and range referential integrity constraints on relationships at more concrete instantiation levels. An intermediary relationship (i.e., a relationship that instantiates another relationship and can itself be further instantiated) refines these constraints. This is akin to the redefinition of associations in the UML (see [6] for an in-depth treatment) where associations are refined with regard to association ends. To allow a similar flexibility in DDI, relationships are asserted by three kinds of so-called A-facts (No. 6): DR-facts (No. 4) which express associated domain and range constraints (No. 7, No. 8), R-facts (No. 2) which express an associated range constraint (No. 7), and D-facts (No. 3) which express an associated domain constraint (No. 8). In the visual illustration of DDI, DR-facts are depicted by solid lines, R-facts and D-facts by dotted arcs directed to the target object or the source object, respectively. We demand that each relationship label is introduced with a single unique DR-fact (No. 14) such that all other relationships with that relationship label are instantiations thereof.
Table 1. DDI modeling primitives, predicates, well-formedness and consistency rules

<table>
<thead>
<tr>
<th>DDI Core: Modeling Primitives and Predicates</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sorts &amp; Asserted Predicates:</strong></td>
</tr>
<tr>
<td>(1) $O$: objects, $\mathbb{N}$: natural numbers including 0, $L$: relationship labels</td>
</tr>
<tr>
<td>(2) $R \subseteq O \times \mathbb{N} \times L \times \mathbb{N} \times O$</td>
</tr>
<tr>
<td>(3) $D \subseteq O \times \mathbb{N} \times L \times \mathbb{N} \times O$</td>
</tr>
<tr>
<td>(4) $DR \subseteq O \times \mathbb{N} \times L \times \mathbb{N} \times O$</td>
</tr>
<tr>
<td>(5) $in \subseteq O \times O$</td>
</tr>
<tr>
<td><strong>Auxiliary Predicates:</strong></td>
</tr>
<tr>
<td>(6) $A(x, i, a, j, y) :\iff R(x, i, a, j, y) \lor DR(x, i, a, j, y) \lor D(x, i, a, j, y)$</td>
</tr>
<tr>
<td>(7) $R'(x, i, a, j, y) :\iff R(x, i, a, j, y) \lor DR(x, i, a, j, y)$</td>
</tr>
<tr>
<td>(8) $D'(x, i, a, j, y) :\iff D(x, i, a, j, y) \lor DR(x, i, a, j, y)$</td>
</tr>
<tr>
<td>(9) $member(x, c) :\iff in(x, c)$</td>
</tr>
<tr>
<td>(10) $nmember(x, c, n) :\iff (n = 0 \land x = c) \lor \exists m \exists d : (n = m + 1) \land nmember(x, d, m) \land member(d, c)$</td>
</tr>
<tr>
<td>(11) $isa(x, c) :\iff in(x, c)$</td>
</tr>
<tr>
<td><strong>Well-formedness Criteria and Syntactic Restrictions:</strong></td>
</tr>
<tr>
<td>(12) $isa^+(x, c) \rightarrow x \neq c$</td>
</tr>
<tr>
<td>(13) $in(x, c) \land in(x, d) \rightarrow c = d$</td>
</tr>
<tr>
<td>(14) $A(x, i, a, j, y) \land A(s, k, a, l, t) \rightarrow \exists c \exists m \exists d : DR(c, m, a, n, d) \land isa^+(x, c) \land isa^+(s, c) \land isa^+(y, d) \land isa^+(t, d)$</td>
</tr>
<tr>
<td>(15) $A(x, i, a, j, y) \land A(s, k, a, l, t) \land isa^+(x, s) \rightarrow j \leq l$</td>
</tr>
<tr>
<td>(16) $A(x, i, a, j, y) \land A(s, k, a, l, t) \land isa^+(y, t) \rightarrow i \leq k$</td>
</tr>
<tr>
<td><strong>DDI with Specialization: Additional Modelling Primitives and Predicates</strong></td>
</tr>
<tr>
<td><strong>Additional Asserted Predicate:</strong></td>
</tr>
<tr>
<td>(17) $spec \subseteq O \times O$</td>
</tr>
<tr>
<td><strong>Auxiliary Predicates (Additions and Redefinitions):</strong></td>
</tr>
<tr>
<td>(18) $member(x, c) :\iff \exists \exists \exists d : spec^+(x, s) \land in(s, d) \land spec^+(d, c)$</td>
</tr>
<tr>
<td>(19) $nmember(x, c, n) :\iff (n = 0 \land spec^+(x, c)) \lor \exists m \exists d : (n = m + 1) \land nmember(x, d, m) \land member(d, c)$</td>
</tr>
<tr>
<td>(20) $isa(x, c) :\iff in(x, c) \lor spec(x, c)$</td>
</tr>
<tr>
<td>(21) $concrete(x) :\iff \exists y : spec(y, x)$</td>
</tr>
<tr>
<td><strong>Well-formedness Criteria and Syntactic Restrictions:</strong></td>
</tr>
<tr>
<td>(22) $in(x, c) \rightarrow concrete(c)$</td>
</tr>
<tr>
<td>(23) $spec(x, s) \land spec(x, z) \rightarrow s = z$</td>
</tr>
<tr>
<td>(24) $spec(x, s) \rightarrow \exists c : in(x, c)$</td>
</tr>
<tr>
<td>(25) $A(x, i, a, j, y) \land A(s, k, a, l, t) \land spec^+(x, s) \rightarrow \neg spec^+(t, y)$</td>
</tr>
<tr>
<td>(26) $R'(x, i, a, j, y) \land R'(x, i, a, j, t) \rightarrow \neg spec^+(y, t)$</td>
</tr>
<tr>
<td>(27) $D'(x, i, a, j, y) \land D'(s, i, a, j, y) \rightarrow \neg spec^+(x, s)$</td>
</tr>
<tr>
<td><strong>Consistent Instantiation and Specialization:</strong></td>
</tr>
<tr>
<td>(28) $A(x, i, a, j, y) \land A(c, k, a, l, d) \land nmember(x, c, n) \rightarrow n = k - i$</td>
</tr>
<tr>
<td>(29) $A(x, i, a, j, y) \land A(c, k, a, l, d) \land nmember(y, d, n) \rightarrow n = l - j$</td>
</tr>
<tr>
<td>(30) $A(x, i, a, j, y) \land R'(c, k, a, l, d) \land (j \leq l) \land isa^+(x, c)$</td>
</tr>
<tr>
<td>$\rightarrow \exists d' : R'(c, k, a, l, d') \land isa^+(y, d')$</td>
</tr>
<tr>
<td>(31) $A(x, i, a, j, y) \land D'(c, k, a, l, d) \land (i \leq k) \land isa^+(y, d)$</td>
</tr>
<tr>
<td>$\rightarrow \exists d' : D'(c', k, a, l, d) \land isa^+(x, c')$</td>
</tr>
</tbody>
</table>
**Example 2 (Relationships).** Relationship label cpu is introduced with relationship cpu\(^{2-2}\) between Processor and MobilePhone and expresses that relationships labeled cpu may exist only between 0..2-members of Processor in the source role and 0..2-members of MobilePhone in the target role. This relationship, being asserted as a DR-fact, imposes domain and range constraints: 0..2-members of Processor may be related by relationships labeled cpu only to 0..2-members of MobilePhone and vice versa. This relationship is instantiated by relationship cpu\(^{1-1}\) between Cortex-A15 and SamsungGS4 which relates 0..1-members of Cortex-A15 to 0..1-members of SamsungGS4. This relationship, being asserted as a D-fact, imposes only a domain constraint but no range constraint: 0..1-members of SamsungGS4 in the target role may be related only to 0..1-members of Cortex-A15 in the source role, but 0..1-members of Cortex-A15 may still be related to other 0..2-members of MobilePhone. This relationship is in turn instantiated by relationship cpu\(^{0-0}\) between CortexA15#1 and SarahsPhone, which is not further instantiated. Relationship label totalRevenue is introduced with relationship totalRevenue\(^{1-3}\) between MobilePhone and ExchangeableTender. It expresses that relationships labeled totalRevenue may exist only between 0..1-members of MobilePhone and 0..3-members of ExchangeableTender. This relationship is instantiated by totalRevenue\(^{0-1}\) between SamsungGS4 and MEuro, which expresses that the total revenue with mobile phone model SamsungGS4 is given only in unit `millions of Euros`. This relationship is in turn instantiated by relationship totalRevenue\(^{0-0}\) between SamsungGS4 and 137inUnit-MEuro.

The potency reduction axioms (No. 28, No. 29) state that potencies of relationships need to be consistent with the number of instantiation steps in related object instantiation hierarchies. We first look at source potencies: If object \(x\) is an \(n\)-member of object \(c\) and both are source of a relationship labeled \(a\) with source potency \(i\) and \(k\), respectively, then the difference between \(i\) and \(k\) must be equal to \(n\) which is the number of instantiation steps between \(x\) and \(c\) (No. 28). The same needs to hold analogously for target potencies (No. 29). For example, totalRevenue\(^{1-3}\) between MobilePhone and ExchangeableTender is instantiated consistently by totalRevenue\(^{0-1}\) between SamsungGS4 and MEuro, which is a 1-member of MobilePhone, and MEuro, which is a 2-member of ExchangeableTender.

The DDI approach requires a local stratification of relationships (No. 15, No. 16) to ensure that every relationship is direct instantiation of at most one other relationship. Local stratification is far more flexible than strict metamodeling and still reduces the overall complexity of the approach and avoids potential conflicts of domain and range constraints due to multiple instantiation. We first look at local stratification with regard to target potencies (No. 15): If object \(s\) is source of a relationship labeled \(a\) with target potency \(l\), then \(*\)-members of \(s\) must not be source of a relationship with the same label \(a\) and a target potency higher than \(l\). The same needs to hold, in the opposite direction, for source potencies (No. 16). For example, this holds for relationship soldPrice\(^{0-0}\) between SarahsPhone and 320inUnit-1Euro, which is direct instantiation of exactly one relationship, namely soldPrice\(^{1-2}\) between SamsungGalaxyS4 and Euro.
This would not be the case, if there also were a relationship soldPrice\(^{0−3}\) between SarahsPhone and ExchangeableTender.

An object may be source of multiple relationships that only differ in the target object and are identical with regard to relationship label and potencies. The range and/or domain restriction imposed by multiple such relationships is given by their union, i.e., a lower-level relationship must instantiate any one of these! For example, object SamsungGS4 is source of soldPrice\(^{1−2}\) relationships with target objects FFlyerPoints and Euro to indicate that SamsungGS4 phones may be paid for in frequent flyer points or in Euro. Consider range restrictions (No. 30): If for a \((i,j)\)-potency relationship labeled \(a\) with source \(x\) and target \(y\) there is a range-restricting relationship labeled \(a\) with a source object \(c\) that is the same as or is an ancestor of \(x\) and a target potency \(l\) that is higher or equal to \(j\), then \(c\) must be source of some range-restricting relationship, which the former relationship instantiates. Domain restrictions (No. 31) work analogously.

The representation of DI using the powertype pattern (see Fig. 3) can be extended to also cover DDI relationships, but one need to take recourse to additional, annotated constraints in OCL that reflect that relationships at higher
levels act as schema for relationships at lower levels (see Fig. 4). The semantic modeling construct DDI, in contrast, keeps schema and associated constraints at one place and implicitly captures involved constraints, a key characteristic of semantic data models [10].

3 DDI with Object Specialization

In this section we investigate the interplay between dual deep instantiation and object specialization and extend the axiomatic specification of DDI accordingly (Table 1, No. 17 – No. 27).

DDI objects at the same instantiation level may be organized in specialization hierarchies, where \( \text{spec}(x, y) \) expresses that object \( x \) is a specialization of object \( y \) (No. 17). The idea of specialization hierarchies in DDI is to factor out common features of objects at the same level to a joint more general object at that level. Features of an object are inherited unchanged by its specializations, unless overridden. We say, an object is \textit{concrete} if it has no specializations (No. 21) and \textit{abstract} otherwise. To simplify the approach we assume that only concrete objects are instantiated (No. 22), a restriction that is akin to the \textit{abstract superclass rule} [11]. Further, we only consider single specialization (No. 23) and assume that an object that is an instantiation of an object may not be at the same time a specialization of some object (No. 24).

The predicates \( \text{isa}, \text{member}, \) and \( \text{nmember} \) are redefined to also cover specialization hierarchies. We say \( x \ text{isa} y \) if \( x \) is an instance of \( y \) or \( x \) specializes \( y \) (No. 20). We say \( x \) is a member of \( y \) if \( x \) relates to \( y \) by a chain of \( \text{isa} \) with exactly one instantiation step (No. 18). We say \( x \) is an \( n \)-member of \( y \) if \( x \) relates to \( y \) by a chain of \( \text{isa} \) with \( n \) instantiation steps (No. 19).

\textbf{Example 3.} Euro and Yen in Fig. 5 are \textit{concrete} types of exchangeable tender. They are made specializations of \textit{Currency}, which is an \textit{abstract} type of exchangeable tender. MEuro and 1Euro are members of Euro as well as of \textit{Currency}, and

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{ddi_object_specialization.png}
\caption{DDI with Object Specialization}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{cross_over_relationships.png}
\caption{Cross-over of Relationships}
\end{figure}
2-members of ExchangeableTender. Euro is a 0-member of itself and of Currency and a 1-member of ExchangeableTender. Relationship label legalTender is introduced with relationship legalTender\(^1\rightarrow^0\) between Country and Currency to express that only types of exchangeable tender that are specializations of Currency can be legal tender in individual countries. Yen is the legal tender in Japan.

The local stratification requirement is extended to exclude also a “cross-over” of several relationships with the same label with respect to the specialization hierarchies of their source and target objects (No. 25) as depicted in Fig. 6. Further, to ensure that the represented range constraints for an object with the same target potency are redundancy free, they need to be to orthogonal target objects (two objects are orthogonal if they are not in a direct or indirect specialization relationship) in a specialization hierarchy (No. 26). This holds analogously for domain constraints (No. 27).

Consistent instantiation and specialization has already been defined in a way (No. 30, No. 31) through the use of isa to cover both, instantiation and specialization. Inherited facts may be overridden but in a co-variant manner, such that any relationship facts in which a source object participates meets the range constraints of its ancestors with regard to instantiation and specialization (No. 30) and conversely, any relationship fact in which a target objects participates meets the domain constraints of its isa-ancestors (No. 31). While the notation of these axioms do not explicitly distinguish between instantiation and specialization, any DDI model must differentiate between them: predicate in corresponds to an element-set relation, and predicate spec to a subset-superset relation. The potency reduction axioms (No. 28, No. 29) ensure that on instantiation, source or target potencies of relationships, are accordingly reduced, but stay the same on specialization. Thus, axioms No. 30 and No. 31 can take this as given and need not discriminate between instantiation and specialization. For example, the target potencies of legalTender are both 0 for Currency and its specialization Yen, but source potencies are reduced from 1 to 0 because Japan is an instantiation of Country.

The presented base model can be extended to relationships that are organized in a specialization-hierarchy to model for example that a privatePhoneNo-relationship of a person to a phone number is also a phoneNo-relationship, by explained along our example - (1) introducing a specialization relation between privatePhoneNo and phoneNumber, (2) applying all domain and range constraints for phoneNo also to privatePhoneNo, and (3) treating each asserted privatePhoneNo relationship fact also as phoneNo relationship fact. We omit a further treatment for space limitations.

4 Implementation in ConceptBase

ConceptBase [12, 14] is a metamodeling system based on Datalog and the Telos data model [21]. All explicit information (objects, classes, metaclasses) are represented with a single extensional predicate \(P(id, from, label, to)\). The principles of instantiation, specialization and attribution are defined by axioms [13]. However, they lack the DDI notion of potency. The implementation strategy is to
create first a Telos metamodel to store DDI models. Secondly, the DDI predicates are defined by deductive rules. Finally, the DDI axioms are encoded as integrity constraints. The result is a prototypical environment for storing and analyzing the consistency of DDI models.

Figure 7 illustrates the OBJECT class and its attributes for the representation of DDI in Telos. The \textit{in} and \textit{spec} predicates from the DDI model translate into the \texttt{IN} and \texttt{SPEC} attributes of the OBJECT class, which are represented by arrows with the respective label. The \texttt{A} predicate from the DDI model translates into the \texttt{link} attribute of the OBJECT class. In ConceptBase, attributes are essentially relationships between objects and it is possible to define attributes for the relationships themselves, similar to association class in UML. The \texttt{link} attribute of the OBJECT class has a label (ALABEL) as well as an integer number for the source potency (sourceLevel) and the target potency (targetLevel). Similarly, attributes (or relationships) may specialize other attributes (or relationships). The \texttt{linkD}, \texttt{linkR}, and \texttt{linkDR} attributes of the OBJECT class specialize the \texttt{link} attribute. These attributes represent the \texttt{D}, \texttt{R}, and \texttt{DR} predicate, respectively; the \texttt{link} attribute derives from these predicates. The following Telos frame defines OBJECT including derived attributes:

\begin{verbatim}
OBJECT with
  attribute
      link: OBJECT; linkDR: OBJECT; linkD: OBJECT; linkR: OBJECT;
      IN: OBJECT; SPEC: OBJECT;
      predD: OBJECT; predR: OBJECT;
      SPEC_t: OBJECT; SPEC_rt: OBJECT;
      ISA: OBJECT
end
\end{verbatim}

Besides the illustration of the DDI metamodel in Telos, Fig. 7 illustrates an example instantiation which represents part of a mobile phone product

\begin{figure}
\centering
\includegraphics[width=\textwidth]{fig7.png}
\caption{Classes (and an instantiation) for the representation of the DDI model in Telos}
\end{figure}
catalog. The Telos individuals MobilePhone, SamsungGS4, ExchangeableTender, Euro, and MEuro are “linguistic” instances of the OBJECT class, i.e., the instantiation relationships between these individuals and the OBJECT class bear the usual ConceptBase semantics. The individuals SamsungGS4 and MobilePhone, MEuro and Euro, as well as Euro and ExchangeableTender are in an instantiation relationship according to DDI semantics represented by the IN relationship. MobilePhone instantiates the linkDR attribute of the OBJECT class. This relationship $m_1$ from MobilePhone to ExchangeableTender has 1 as sourceLevel, 3 as targetLevel, and totalRevenue as label. SamsungGS4 also instantiates the linkDR attribute of the OBJECT class. This relationship $m_2$ from SamsungGS4 to Euro has 0 as sourceLevel, 1 as targetLevel, and totalRevenue as label.

The IN (DDI instantiation) and SPEC (DDI specialization) attributes are used for the representation of extensional predicates. The attribute SPEC_t is the transitive closure of the SPEC attribute, SPECrt its the reflexive and transitive closure. The closures are implemented as simple deductive rules in ConceptBase. The attributes allow to add facts such as (SamsungGS4 IN MobilePhone) to the database as well as to define rules and constraints imposed on them. Distinguish the DDI instantiation fact with label IN from the Telos instantiation fact, e.g., (SamsungGS4 in OBJECT), which represents the DDI model within Telos. The predicates of the DDI model (Sect. 2) are implemented by deductive rules in ConceptBase. For example, the following deductive rule represents the isa defined by formula (20) in Table 1:

forall $x,y$/OBJECT ($x$ SPEC $y$) or ($x$ IN $y$) $\Rightarrow$ ($x$ ISA $y$)

Whereas the nmember predicate of the DDI model translates to a function, the $A(x, s, a, t, y)$ predicate requires a Telos formalization that uses the $A_i(x, link, m)$ predicate provided by ConceptBase which returns an instance $m$ of the link attribute of object $x$:

$A(x,s,a,t,y) :\Leftrightarrow \exists m/OBJECT!link A_i(x,link,m)$ and $To(m,y)$ and ($m$ label $a$) and ($m$ sourcelevel $s$) and ($m$ targetlevel $t$)

The axioms of the DDI model are transcribed using the formula for the $A$ predicate of the DDI model. The DDI axioms become integrity constraints in ConceptBase. As an example consider the translation of axiom (28) in Table 1:

forall $x,y,c,d$/OBJECT $a$/ALABEL $i,j,k,l,n$/Integer $m_x,m_c$/OBJECT!link
$A_i(x,link,m_x)$ and $To(m_x,y)$ and ($m_x$ label $a$) and ($m_x$ sourcelevel $i$) and ($m_x$ targetlevel $j$) and $A_i(c,link,m_c)$ and $To(m_c,d)$ and ($m_c$ label $a$) and ($m_c$ sourcelevel $k$) and ($m_c$ targetlevel $l$) and ($n = NMEMBER(x,c)$) $\Rightarrow$ ($n = k - i$)

A large portion of the predicates in the formula are due to the mechanic expansion of the quinary predicate $A$. The function NMEMBER recursively computes the number of DDI instantiations between two objects. It is compared with the arithmetic expression $k - i$. ConceptBase offers a validation of axioms within the expressiveness of a Datalog-neg [1] fixpoint model. An axiom represented as a constraint – or formally, a rule that derives inconsistency – can be violated by a given database
state, i.e. an example DDI model. Around 50 example models were created to validate the non-trivial satisfiability of the axioms under Datalog semantics. The examples also showed that the implementation of DDI coincided with the behavior we expected. The tested example models include the examples used in this paper. In order to demonstrate that the DDI axioms are free of redundancy, we provide for each axiom \( a \) an example “negative” DDI model that violates \( a \) but no other axiom. Besides the validation, the implementation was also used in the evolution of the DDI axioms: a revised set of axioms can be checked against the given example DDI models to analyze the resulting behavior. All example DDI models, the full DDI implementation, and further documentation are available online at [http://conceptbase.cc/ddi](http://conceptbase.cc/ddi).

### 5 Related Work

*VODAK* [15] was the first system to introduce what was later called deep instantiation/characterization [8,17]. The capability of deep instantiation was achieved by VODAK in that objects where in parallel to types also organized into classes and meta-classes. Meta-classes defined own-types, instance-types and instance-instances-types, basically representing a cascading structure of what later became known as *power type* design pattern [8]. Type instantiation is between an object and its class’s instance type and is inferred from the functional association of an object to a class to provide for a lossless and redundancy-free representation of functional data dependencies according to database design theory. Different to DDI, the semantics of the relationships between objects at different instantiation levels was captured behaviorally by methods attached to types and not by implicit constraints as in DDI.

Telos [21] and its implementation *ConceptBase* [12,14] support a loose form of deep characterization. An attribute may be instantiated recursively many times and thereby acts as referential integrity constraint for its direct and indirect instances. The Telos axiomatization, however, does not specify the depth of characterization and attributes carry only a range but no domain constraint.

Deep instantiation [4] extends the approach of VODAK to an arbitrary number of instance\(^n\)-types, but rather than explicitly introducing types, attributes receive a potency number indicating to which implicit instance\(^n\)-type they belong. Deep instantiation was complemented by a Java implementation [16]. The discussion of deep instantiation of connectors, (relationships) [3,9] focuses on their graphical rendering and is limited to strict metamodeling [5]. The modeling language Nivel [2] provides a formal semantics for deep instantiation. Materialization [24] is akin to deep instantiation and allows for deep characterization by introducing different types of attribute propagation. Similar to deep instantiation, materialization focuses on attributes and not on bi-directional relationships.

MetaDepth [18,20] is a meta-modelling tool that incorporates deep instantiation. The practical experiences with MetaDepth [19] fit well with our own findings. They also recognized the need to relate objects at different meta-levels and do so by *deep references*. However, this novel modeling construct is not
elaborated in detailed and only described informally. Further, deep references in MetaDepth are uni-directional instead of bi-directional relationships as known from the E/R model or from associations in the UML, consequently they only carry a range referential integrity constraint but no domain constraint.

*M-Objects and M-Relationships* [22] achieve deep characterization by a hierarchy of named concretization (instantiation) levels. Different to DDI, with m-relationships it is not possible to separately refine domain or range constraints and they do not come with an implementation that allows for automatic consistency checking. A comparison with different techniques for deep characterization, then called ‘multi-level abstraction’, is given in [23].

This paper contributes to the state of the art in deep characterization by providing a simple, formal representation and associated consistency axioms that cover (i) DDI with instantiation and specialization, (ii) domain or range redefinition of DDI relationships as it is known for UML associations, (iii) DDI model restrictions that on the one hand lift the requirement of global stratification through strict-meta modelling but on the other hand ensure local stratification (as explained in Sect. 2), and (iv) associated consistency checks for instantiation and specialization with multiple dominating domain- or range constraints. This paper further contributes to the debate [7] about the ontological adequacy of deep instantiation by clearly differentiating (see Sect. 2) between the multifaceted instantiation-of relationship between DDI objects at different instantiation levels and the instance-of relationship between an object and its class.

### 6 Conclusion

We have introduced the semantic modeling primitive DDI along a small running example, integrating different applications, a mobile phone catalog and associated inventories for different phone models. We have demonstrated that DDI allows for a concise representation of integrity constraints without need to take recourse to sophisticated constraint annotations. We currently work on extending the DDI approach to specialization of relationships as shortly exemplified in Sect. 3 and on introducing cardinality constraints at multiple instantiation levels.

### References


http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/d1228997/O-Telos-Axioms.pdf


