On the Properties of Metamodeling in OWL

Boris Motik

University of Manchester

Manchester, UK

bmotik <at> cs.man.ac.uk

August 25, 2007

Abstract

A common practice in conceptual modeling is to separate the conceptual from the data model. Although very intuitive, this approach is inadequate for many complex domains, in which the borderline between the two models is not clear-cut. Therefore, OWL Full, the most expressive of the Semantic Web ontology languages, allows us to combine the conceptual and the data model by a feature we refer to as *metamodeling*. In this paper, we show that the semantics of metamodeling adopted in OWL Full leads to the undecidability of basic inference problems due to the free usage of the built-in vocabulary. Based on this result, we propose two alternative semantics for metamodeling: the *contextual* and the *HiLog* semantics. We present several examples showing how to use the latter semantics to axiomatize the interaction between concepts and metaconcepts. Finally, we show that $\mathcal{SHOIQ}(\mathbf{D})$—the description logic underlying OWL DL—is still decidable when extended with metamodeling under either semantics.

Keywords: knowledge representation, description logics, metamodeling, decidabilty, OWL, Semantic Web

Note: This is an extended version of a paper with the same name that has been presented at ISWC 2005 [15].

1 Introduction

The Web Ontology Language (OWL) [20] is a family of ontology languages developed and standardized by the World Wide Web Consortium in order to enable semantic annotation of Web resources. Three variants of OWL with varying expressive power have been defined. OWL Lite and OWL DL are based on description logics (DLs)—a family of knowledge representation languages—$\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, respectively. These DLs are well-understood, both semantically and computationally, and their basic inference problems are known to be decidable in ExpTime and NExpTime, respectively. Furthermore, practical decision procedures for both languages have been developed and successfully implemented in several reasoners, such as FaCT++ [24], Pellet [19], RACER [6], or KAON2 [17]. OWL Full is the most expressive of the OWL family of languages. Its syntax is very similar to OWL DL and OWL Lite, but its semantics is not based on DLs; rather, it has been designed to capture the style of conceptual modeling we describe next.

OWL Lite and OWL DL, as well as all DLs known to us, follow a common practice in conceptual modeling that strictly separates the *conceptual* from the *data* model. The conceptual model is analogous to a database schema in that it describes the general structure and the regularities of the world. The data model is analogous to a database instance in that it describes a particular state of the world. In OWL, the conceptual model consists of concepts and roles, and the data model consists of individuals and relationships among them. To better understand this duality, consider the following example, originally presented in [25]; a similar example can be found in [23]. A natural way to represent kinship between animal species is to organize them in a hierarchy of concepts. For example, we can use the concept *Bird* to represent the set of all birds, and we can make the concept *Eagle* a subconcept of *Bird* to state that all eagles are birds. These statements are of a conceptual nature since they define the general notions of birds and eagles. Knowledge about concrete animals, such as saying

that the individual *Harry* is an instance of the concept *Eagle*, is an example of data knowledge. Combining conceptual and data knowledge together allows us to derive new conclusions, such as the fact that *Harry* is a *Bird*.

In certain applications, one might want to make statements about species themselves, such as "eagles are listed in the IUCN Red List[1] of endangered species." Note an important distinction: we do not want to say that each individual eagle is listed in the Red List, but that the eagle species as a whole is. To represent this statement formally, we can introduce a concept *RedListSpecies* containing all species listed in the Red List. Note, however, that *Eagle* is also a concept, so the proper relationship between *RedListSpecies* and *Eagle* is not clear. Making *Eagle* a subconcept of *RedListSpecies* is incorrect, as this would imply that *Harry* is a *RedListSpecies*—clearly an undesirable conclusion. Rather, *Eagle* is a *type of* (or, equivalently, an instance of) *RedListSpecies*; in other words, *RedListSpecies* is a *metaconcept* for *Eagle*. Modeling with metaconcepts is called *metamodeling*,[2] and it can be used to build concise models if we precisely axiomatize the properties of metaconcepts. For example, by stating that "one is not allowed to hunt the instances of the species listed in the Red List," we formalize the logical properties of the metaconcept *RedListSpecies*, which then allows us to deduce that "one is not allowed to hunt *Harry*."

OWL Full provides a great degree of freedom regarding what can be said and it supports metamodeling. OWL DL differs from OWL Full mainly in that (*i*) it does not allow one to state axioms about the built-in vocabulary (i.e., the symbols, such as *owl:allValuesFrom*, used in the definition of the semantics), (*ii*) it strictly separates the sets of symbols used as concepts, roles, and individuals, and (*iii*) it enforces the well-known restrictions required for decidability, such as allowing only simple roles in number restrictions [12]. Because of the condition (*ii*), OWL DL does not support metamodeling.

[1] `http://www.redlist.org/`

[2] Metamodeling is sometimes understood as studying an object language by means of another metalanguage. In our work, however, metamodeling should be understood in the sense of "higher-order"; for example, a concept is an instance of a higher-order metaconcept. Such usage of the prefix "meta-" has become common in the discussion of Semantic Web ontology languages.

Since it does not enforce (*iii*), OWL Full is trivially undecidable. To practically implement reasoners for expressive logics, advanced optimization techniques are essential, and these are much easier to develop if the logic is decidable [1, Chapter 9]. To obtain a decidable logic that supports metamodeling, it is natural to ask whether imposing the well-known restrictions on OWL Full (or, equivalently, whether extending OWL DL with metamodeling in the style of OWL Full) yields a decidable logic. In Section 2 we answer this question negatively by showing that even fairly inexpressive DLs become undecidable if restrictions (*i*) and (*ii*) are not enforced.

We analyze this undecidability result and show that it is actually due to (*i*)—that is, the fact that we can state axioms about the built-in vocabulary. This makes the semantics of OWL Full nonstandard and controversial [9]. Therefore, in Section 3 we present two alternative semantics: a *contextual* or π-semantics, which is essentially first-order, and a *HiLog* or ν-semantics, which is inspired by HiLog [5]—a logic providing a second-order flavor to first-order logic.

Metamodeling is sometimes criticized on the grounds that it is not clear in what way it affects the logical consequences of a theory. Therefore, in Section 4 we discuss the added expressivity of metamodeling using an example. We identify several cases in which both semantics coincide, and present cases where metamodeling provides new consequences. Our results show that metamodeling, when used with OWL DL alone, does not allow us to derive many new consequences; however, it is very useful when combined with expressive extensions of DLs. For example, using the Semantic Web Rule Language (SWRL) [10], one can precisely define the semantics of metaconcepts, which allows for a very powerful and concise modeling style.

Finally, in Section 5 we show that, under some technical assumptions, both semantics can be combined with the DL $\mathcal{SHOIQ}(\mathbf{D})$ (a slightly more expressive variant of OWL DL), yielding a decidable fragment of OWL Full with the same worst case complexity as OWL DL.

2 Undecidability of Metamodeling in OWL Full

In this section, we show that OWL Full is undecidable even if we impose the well-known restrictions required for the decidability of OWL DL. The definition of OWL Full [20] is quite complex, so we introduce a simplified version of it. Syntactically, OWL Full has been layered on top of the Resource Description Framework (RDF) [14]—a semistructured data model that represents graphs as triples, so our definition follows such an approach as well. We use the standard values for the *rdf:*, *rdfs:*, and *owl:* namespace prefixes.

Definition 1. *Let V be the* vocabulary *set containing exactly these symbols:*

rdf:first, rdf:rest, rdf:nil, owl:Thing, owl:Nothing, rdf:type, rdfs:subClassOf, owl:complementOf, owl:unionOf, owl:intersectionOf, owl:someValuesFrom, owl:allValuesFrom, owl:hasValue, owl:onProperty

Let N be the set of names *such that $V \subseteq N$. An $\mathcal{ALCO}$-Full* knowledge base *KB is a finite set of triples of the form $\langle s, p, o\rangle$, where $s, p, o \in N$. We use $\langle s, p, L[a_1, \ldots, a_n]\rangle$ as a shortcut for the following triples, where $x_1, \ldots, x_n$ are fresh names not occurring anywhere else in KB:*

$$\langle s, p, x_1\rangle,\ \langle x_1, \textit{rdf:first}, a_1\rangle,\ \langle x_1, \textit{rdf:rest}, x_2\rangle, \quad \ldots,$$
$$\langle x_n, \textit{rdf:first}, a_n\rangle,\ \langle x_n, \textit{rdf:rest}, \textit{rdf:nil}\rangle.$$

An interpretation *I is a triple $(\triangle^I, \cdot^I, EXT^I)$, where $\triangle^I$ is a nonempty* domain *set, $\cdot^I : N \to \triangle^I$ is an* interpretation function*, and $EXT^I : \triangle^I \to 2^{\triangle^I \times \triangle^I}$ is an* extension function*. Let $CEXT^I : \triangle^I \to 2^{\triangle^I}$ be the* concept extension function *defined as $CEXT^I(x) = \{y \mid (y, x) \in EXT^I(\textit{rdf:type}^I)\}$. If $y \in CEXT^I(x)$, we say that y is an* instance *of x. An interpretation I is a* model *of KB if it satisfies all conditions from Table 1 on page 7. KB is* satisfiable *if and only if a model of KB exists.*

In contrast to OWL Full, $\mathcal{ALCO}$-Full does not provide for concrete predicates, inverse and transitive roles, role hierarchies, and number restrictions, and it does not include the *axiomatic triples* [7]. These differences do not affect our undecidability proof in any way.

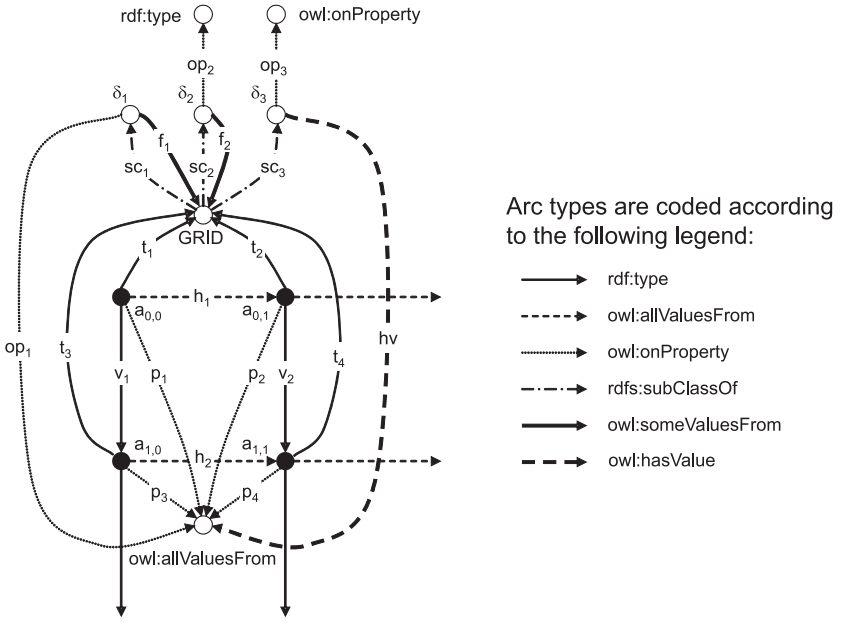

Figure 1: Grid Structure in a Model of $KB_{\mathcal{D}}$

We prove that checking satisfiability of $\mathcal{ALCO}$-Full knowledge bases is undecidable by a reduction from the Domino Tiling problem [4]. A *domino system* is a triple $\mathcal{D} = (D, H, V)$, where $D = \{D_1, \ldots, D_n\}$ is a finite set of *domino types*, $H : D \to 2^D$ is the *horizontal compatibility condition*, and $V : D \to 2^D$ is the *vertical compatibility condition*. A $\mathcal{D}$*-tiling* of an infinite grid is a function $t : \mathbb{N} \times \mathbb{N} \to D$ such that $t(i, j+1) \in H(t(i, j))$ and $t(i+1, j) \in V(t(i, j))$ for all $i, j \in \mathbb{N}$. Checking whether a $\mathcal{D}$-tiling exists is undecidable in general [4].

For a domino system with n tiles, let $KB_{\mathcal{D}}$ be the $\mathcal{ALCO}$-Full knowledge base containing triples (1)–(9) shown on the left-hand side of Table 2 on page 7. As an aid to the reader, we paraphrase the meaning of these triples using the standard DL notation on the right-hand side of the table. We now show that $KB_{\mathcal{D}}$ exactly encodes Domino Tiling for $\mathcal{D}$.

Lemma 1. *A $\mathcal{D}$-tiling exists if and only if $KB_{\mathcal{D}}$ is satisfiable.*

Proof. ($\Leftarrow$) Let I be a model of $KB_{\mathcal{D}}$. We show that each I contains a two-dimensional grid, as shown in Figure 1. In the figure, a triple $\langle s, p, o \rangle$ is repre-

Table 1: Semantics of $\mathcal{ALCO}$-Full

1.	$\langle s, p, o\rangle \in KB$ implies $(s^I, o^I) \in \mathrm{EXT}^I(p^I)$
2.	$\mathrm{CEXT}^I(owl{:}Thing^I) = \triangle^I$
3.	$\mathrm{CEXT}^I(owl{:}Nothing^I) = \emptyset$
4.	$(x, y) \in \mathrm{EXT}^I(rdfs{:}subClassOf^I)$ iff $\mathrm{CEXT}^I(x) \subseteq \mathrm{CEXT}^I(y)$
5.	$(x, y) \in \mathrm{EXT}^I(owl{:}complementOf^I)$ iff $\mathrm{CEXT}^I(x) = \triangle^I \setminus \mathrm{CEXT}^I(y)$
6.	$(x, y) \in \mathrm{EXT}^I(owl{:}unionOf^I)$ iff y is a sequence of $y_1, \ldots, y_n$ in I and $\mathrm{CEXT}^I(x) = \mathrm{CEXT}^I(y_1) \cup \ldots \cup \mathrm{CEXT}^I(y_n)$
7.	$(x, y) \in \mathrm{EXT}^I(owl{:}intersectionOf^I)$ iff y is a sequence of $y_1, \ldots, y_n$ in I and $\mathrm{CEXT}^I(x) = \mathrm{CEXT}^I(y_1) \cap \ldots \cap \mathrm{CEXT}^I(y_n)$
8.	$(x, y) \in \mathrm{EXT}^I(owl{:}someValuesFrom^I)$ and $(x, p) \in \mathrm{EXT}^I(owl{:}onProperty^I)$ imply $\mathrm{CEXT}^I(x) = \{w \mid \exists z \in \triangle^I : (w, z) \in \mathrm{EXT}^I(p) \wedge z \in \mathrm{CEXT}^I(y)\}$
9.	$(x, y) \in \mathrm{EXT}^I(owl{:}allValuesFrom^I)$ and $(x, p) \in \mathrm{EXT}^I(owl{:}onProperty^I)$ imply $\mathrm{CEXT}^I(x) = \{w \mid \forall z \in \triangle^I : (w, z) \in \mathrm{EXT}^I(p) \rightarrow z \in \mathrm{CEXT}^I(y)\}$
10.	$(x, y) \in \mathrm{EXT}^I(owl{:}hasValue^I)$ and $(x, p) \in \mathrm{EXT}^I(owl{:}onProperty^I)$ imply $\mathrm{CEXT}^I(x) = \{w \mid (w, y) \in \mathrm{EXT}^I(p)\}$
11.	x_1 is a sequence of $y_1, \ldots, y_n$ in I iff $x_2, \ldots, x_{n+1} \in \triangle^I$ exist such that, for $1 \leq i \leq n$, $(x_i, y_i) \in \mathrm{EXT}^I(rdf{:}first^I)$ and $(x_i, x_{i+1}) \in \mathrm{EXT}^I(rdf{:}rest^I)$, and $x_{n+1} = rdf{:}nil^I$.

Table 2: The Knowledge Base $KB_{\mathcal{D}}$

(1)	$\langle owl{:}Nothing, owl{:}intersectionOf, L[D_i, D_j]\rangle$	$D_i \sqcap D_j \sqsubseteq \bot$
(2)	$\langle GRID, rdfs{:}subClassOf, \alpha\rangle$ $\langle \alpha, owl{:}unionOf, L[D_1, \ldots, D_n]\rangle$	$GRID \sqsubseteq D_1 \sqcup \ldots \sqcup D_n$
(3)	$\langle NotGRID, owl{:}complementOf, GRID\rangle$	$NotGRID \equiv \neg GRID$
(4)	$\langle D_i, rdfs{:}subClassOf, \beta_i\rangle$ $\langle \beta_i, owl{:}onProperty, owl{:}allValuesFrom\rangle$ $\langle \beta_i, owl{:}allValuesFrom, \beta_i'\rangle$ $\langle \beta_i', owl{:}unionOf, L[H(D_i)]\rangle$	$D_i \sqsubseteq \forall owl{:}allValuesFrom. \bigsqcup_{d \in H(D_i)} d$
(5)	$\langle D_i, rdfs{:}subClassOf, \gamma_i\rangle$ $\langle \gamma_i, owl{:}onProperty, rdf{:}type\rangle$ $\langle \gamma_i, owl{:}allValuesFrom, \gamma_i'\rangle$ $\langle \gamma_i', owl{:}unionOf, L[NotGRID, V(D_i)]\rangle$	$D_i \sqsubseteq \forall rdf{:}type. NotGRID \sqcup \bigsqcup_{d \in V(D_i)} d$
(6)	$\langle GRID, rdfs{:}subClassOf, \delta_1\rangle$ $\langle \delta_1, owl{:}onProperty, owl{:}allValuesFrom\rangle$ $\langle \delta_1, owl{:}someValuesFrom, GRID\rangle$	$GRID \sqsubseteq \exists owl{:}allValuesFrom. GRID$
(7)	$\langle GRID, rdfs{:}subClassOf, \delta_2\rangle$ $\langle \delta_2, owl{:}onProperty, rdf{:}type\rangle$ $\langle \delta_2, owl{:}someValuesFrom, GRID\rangle$	$GRID \sqsubseteq \exists owl{:}type. GRID$
(8)	$\langle GRID, rdfs{:}subClassOf, \delta_3\rangle$ $\langle \delta_3, owl{:}onProperty, owl{:}onProperty\rangle$ $\langle \delta_3, owl{:}hasValue, owl{:}allValuesFrom\rangle$	$GRID \sqsubseteq \exists owl{:}onProperty. \{owl{:}allValuesFrom\}$
(9)	$\langle a_{0,0}, rdf{:}type, GRID\rangle$	$GRID(a_{0,0})$

Note: Axioms containing indices i and j should be instantiated for each $1 \leq i < j \leq n$.

sented as an arc pointing from the node s to the node o with p encoded using the line type according to the legend. For easier reference, we assign labels to arcs. For example, the horizontal grid arcs are labeled with h_i, and the vertical grid arcs are labeled with v_i. These labels do not correspond to p in the triple; for example, the arc h_1 represents the triple $\langle a_{0,0}, \mathit{owl{:}allValuesFrom}, a_{0,1}\rangle$. By an abuse of notation, we do not distinguish symbols from their interpretation.

The grid is constructed using triples (6)–(9). Triple (9) implies the existence of node $a_{0,0}$ and arc t_1. Triples (7) imply the existence of arcs sc_2, op_2, and f_2, which, by Item 8 of Table 1, imply that each instance of δ_2 has an *rdf:type* arc to another instance of *GRID*. By sc_2 and Item 4 of Table 1, this also holds for each instance of *GRID*, so I contains node $a_{1,0}$, and arcs v_1 and t_3. Similarly, triples (6) imply the existence of arcs sc_1, op_1, and f_1, which, by Item 8 of Table 1, imply that each instance of δ_1 has an *owl:allValuesFrom* arc to another instance of *GRID*. By sc_1 and Item 4 of Table 1, this also holds for each instance of *GRID*, so I contains nodes $a_{0,1}$ and $a_{1,1}$, and arcs h_1, t_2, h_2, and t_4.

The crux of the construction is to close the grid—that is, to ensure that I contains arc v_2. This is done by applying Item 9 of Table 1 for $x = a_{1,0}$, $y = a_{1,1}$, $p = \mathit{owl{:}allValuesFrom}$, $w = a_{0,0}$, and $z = a_{0,1}$. Arc h_2 ensures that condition $(x, y) \in \mathrm{EXT}^I(\mathit{owl{:}allValuesFrom}^I)$ of Item 9 holds, arc v_1 ensures $w \in \mathrm{CEXT}^I(x)$, and arc h_1 ensures $(w, z) \in \mathrm{EXT}^I(p)$. If we satisfy condition $(x, p) \in \mathrm{EXT}^I(\mathit{owl{:}onProperty}^I)$, then, by Item 9, we have $z \in \mathrm{CEXT}^I(y)$, which produces the required arc v_2. We obtain the missing *owl:onProperty* arc from x to *owl:allValuesFrom* as follows. Triples (8) imply the existence of arcs sc_3, op_3, and hv, which, by Item 10 of Table 1, imply that each instance of δ_3 has an *owl:onProperty* arc to node *owl:allValuesFrom*. By sc_3 and Item 4 of Table 1, this holds for each instance of *GRID* as well, so I contains arc p_3. Thus, $a_{0,0}$, $a_{0,1}$, $a_{1,0}$, and $a_{1,1}$ form a grid, which, by an inductive application of the same argument, extends indefinitely in horizontal and vertical directions.

Note that a node $a_{i,j}$ in I is allowed to have multiple *owl:allValuesFrom* and *rdf:type* successors, so I need not be a two-dimensional grid. A two-

dimensional grid, however, can easily be extracted from I: one can choose any *owl:allValuesFrom* successor $a_{i,j+1}$ and any *rdf:type* successor $a_{i+1,j}$ of $a_{i,j}$, as well as any *owl:allValuesFrom* successor $a_{i+1,j+1}$ of $a_{i+1,j}$. Regardless of the choices, $a_{i,j+1}$ is always connected to $a_{i+1,j+1}$ by *rdf:type*, so $a_{i,j}$, $a_{i,j+1}$, $a_{i+1,j}$, and $a_{i+1,j+1}$ are connected in a grid-like manner.

Triples (2) ensure that each instance of *GRID* is assigned at least one tile type. Triples (1) axiomatize the extensions of tile types D_i and D_j to be disjoint whenever $i \neq j$, so each instance of *GRID* is assigned exactly one tile type. Finally, (4) axiomatizes the horizontal compatibility condition. By (3) and Item 5 of Table 1, no instance of *GRID* can be an instance of *NotGRID*; hence, (5) axiomatizes the vertical compatibility condition. Thus, I contains an infinite two-dimensional grid in which nodes satisfy the compatibility conditions, so a $\mathcal{D}$-tiling can be extracted easily from I.

($\Rightarrow$) For a $\mathcal{D}$-tiling t, we construct an interpretation I as follows. Let $\triangle^I$ contain V, all names occurring in $KB_{\mathcal{D}}$, and $a_{i,j}$ for each $i, j \in \mathbb{N}$. We interpret each name n by itself—that is, $n^I = n$. We shall define EXT^I in stages. Initially, we make EXT^I contain the arcs shown in Figure 1. To satisfy triples (6)–(9) and Items 1–4, 8, and 10 of Table 1, extend I with appropriate *rdf:type* arcs to *owl:Thing*, δ_1, δ_2, and δ_3.

By Item 9 of Table 1, the concept extension of each $a_{i,j}$ should contain those objects w for which each *owl:allValuesFrom* arc points to an instance of $a_{i,j+1}$. For $i > 0$ and $w = a_{i-1,j}$, the interpretation I already contains an *rdf:type* arc from w to $a_{i,j}$ (see Figure 1). Furthermore, for each $w = a_{k,l}$ such that $w \neq a_{i-1,j}$, node w has an *owl:allValuesFrom* arc to $a_{k,l+1}$ that is not an instance of $a_{i,j+1}$, so we *do not* add an arc from w to $a_{i,j}$. Finally, for w such that $w \notin \mathrm{CEXT}^I(GRID)$, there is no *owl:allValuesFrom* arc from w, so we extend I by adding an *rdf:type* arc from w to $a_{i,j}$. Note that, after these extensions, $\mathrm{CEXT}^I(GRID)$ still contains exactly the objects $a_{i,j}$, and that no additional *rdf:type* arcs between the elements of *GRID* have been introduced.

We now extend I such that $\mathrm{CEXT}^I(NotGRID) = \triangle^I \setminus \mathrm{CEXT}^I(GRID)$ and

$\text{CEXT}^I(D_k) = \{a_{i,j} \mid t(i,j) = D_k\}$, and we add to I the appropriate arcs that satisfy Items 4–7 of Table 1. Clearly, this neither adds new instances to *GRID* nor affects the grid structure. Since t is a tiling, it is clear that (2)–(5) are satisfied. Note that the *NotGRID* disjunct in (5) ensures that vertical compatibility is ensured only along the *rdf:type* arcs that connect instances of *GRID*. Thus, we conclude that I is a model of $KB_{\mathcal{D}}$. □

Undecidability of $\mathcal{ALCO}$-Full is an immediate consequence of Lemma 1 and the known undecidability result for the domino tiling problem [4]:

Theorem 1. *Checking satisfiability of an $\mathcal{ALCO}$-Full knowledge base KB is undecidable.*

Our reduction relies on nominals (the *owl:hasValue* construct) to ensure that each grid element has an *owl:onProperty* arc to the node *owl:allValuesFrom* (the p_i arcs). Furthermore, this reduction holds even if we make the unique name assumption (UNA)—that is, if we require different names to be interpreted as different objects from $\triangle^I$. In [15], we presented an undecidability proof for a simpler logic $\mathcal{ALC}$-Full, which does not provide for nominals, but that allows us to state equivalence between names.

Definition 2. *The logic $\mathcal{ALC}$-Full is defined as in Definition 1, with the difference that the vocabulary V does not contain owl:hasValue, but it contains owl:sameAs, which is interpreted as follows:*

$$(x, y) \in EXT^I(owl{:}sameAs^I) \text{ iff } x = y.$$

Instead of (8), in $\mathcal{ALC}$-Full we can use the following assertions to obtain the necessary arcs op_i from each instance of *GRID* to *owl:allValuesFrom*:

(10) $\langle GRID, rdfs{:}subClassOf, owl{:}allValuesFrom\rangle$

(11) $\langle rdf{:}type, owl{:}sameAs, owl{:}onProperty\rangle$

For a domino system $\mathcal{D}$, let $KB'_{\mathcal{D}}$ be an $\mathcal{ALC}$-Full knowledge base containing triples (1)–(7), (9), (10), and (11). We now prove the following result:

Theorem 2. *Checking satisfiability of an $\mathcal{ALC}$-Full knowledge base KB is undecidable.*

Proof. We show that Lemma 1 holds for $KB'_{\mathcal{D}}$ as well.

($\Leftarrow$) Let I be a model of $KB'_{\mathcal{D}}$. By (11), all *rdf:type* arcs in I can be read as *owl:onProperty* arcs as well. Furthermore, by (10), each instance of *GRID* is also an instance x of *owl:allValuesFrom*, so there is an *rdf:type* arcs from each x to *owl:allValuesFrom*. The latter arc can also be read as *rdf:onProperty*, so I contains the arcs p_i shown in Figure 1. The rest of the grid construction is the same as in Lemma 1.

($\Rightarrow$) For a $\mathcal{D}$-tiling t, we construct an interpretation I as follows. Let $\triangle^I$ contain $V \setminus \{$*owl:onProperty*$\}$, all names from $KB'_{\mathcal{D}}$ apart from *owl:onProperty*, and $a_{i,j}$ for each $i, j \in \mathbb{N}$. Furthermore, for $n \neq$ *owl:onProperty*, we set $n^I = n$, and for $n =$ *owl:onProperty*, we set $n^I =$ *rdf:type*. We construct EXT^I similarly as in Lemma 1. We start with arcs shown in Figure 1 (without node δ_3 and arcs sc_3, op_3, and hv). The main difference to the proof of Lemma 1 is that *rdf:type* arcs can be read as *owl:onProperty* arcs, so we might need to add additional *rdf:type* arcs to satisfy Items 8–10 of Table 1. Clearly, this can be the case only for nodes with either an outgoing *owl:allValuesFrom* arc (i.e., nodes $a_{i,j}$) or an outgoing *owl:someValuesFrom* arc (i.e., nodes δ_1 and δ_2).

By v_1, h_1, and Item 9 of Table 1, $\mathrm{CEXT}^I(a_{0,0})$ must contain all objects that have no $a_{1,0}$ arc to $a_{0,1}$. To satisfy Item 9 without adding *rdf:type* arcs between the grid elements, we extend I by adding an $a_{1,0}$ arc from each instance of *GRID* to, say, *owl:Thing*. More generally, if some $a_{i,j}$ has an *rdf:type* arc to some node λ, then we extend I by adding to all instances of *GRID* a λ arc to *owl:Thing*. To satisfy Item 9 for nodes that are not instances of *GRID*, we simply add an *rdf:type* arc from the relevant nodes to $a_{i,j}$.

Consider now δ_1. Together with f_1, each *rdf:type* arc from δ_1 to some node λ means that each instance of *GRID* must have a λ arc to an instance of *GRID*. For $\lambda =$ *rdf:type* and $\lambda =$ *owl:allValuesFrom*, this condition is already satisfied in I. For the other values of λ, we simply extend I by adding to each instance

of *GRID* a λ arc to, say, $a_{0,0}$. For δ_2, we can satisfy Item 8 in the same way. As in Lemma 1, we can now extend this interpretation to respect the compatibility conditions and thus produce a model of $KB'_{\mathcal{D}}$. □

Unlike Theorem 1, Theorem 2 does not hold if we assume UNA, since there is no way to construct a model in which triple (11) is true.

3 Two Alternatives for Metamodeling

The proof of Lemma 1 suggests the causes for undecidability. Namely, OWL Full not only allows us to treat concepts as individuals, but it allows the built-in vocabulary (the resources such as *owl:allValuesFrom*) to be used as individuals in axioms. We exploited this in assertions (6)–(8) of $KB_{\mathcal{D}}$, in which we stated existential and universal restrictions on *owl:allValuesFrom*, *rdf:type*, and *owl:onProperty*. We do not expect this feature of OWL Full to be particularly useful in practice, so in this section we explore two decidable alternatives.

We proceed as follows. First, in Section 3.1, we devise a DL syntax that provides for metamodeling. We show that metamodeling imposes several rather unexpected requirements on the syntax that should be addressed in the OWL standard. Next, in Section 3.2, we present two different semantics for metamodeling: the *contextual semantics* is a slightly adapted first-order semantics, whereas the *HiLog semantics* is more in the spirit of OWL Full and has been inspired by HiLog [5]. Finally, in Section 3.3, we show how to encode the HiLog semantics into a first-order formula.

3.1 The Syntax of $\mathcal{SHOIQ}(\mathbf{D})$ with Metamodeling

$\mathcal{SHOIQ}(\mathbf{D})$ supports reasoning with concrete datatypes, such as strings or integers. Instead of axiomatizing datatypes in logic, $\mathcal{SHOIQ}(\mathbf{D})$ employs a restricted version of the approach from [2], where the properties of concrete datatypes are encapsulated in *concrete domains*. OWL DL allows only for unary

concrete predicates, so we restrict ourselves to such predicates for simplicity; however, extending our results to n-ary predicates is straightforward.

Definition 3. *A concrete domain* $\mathbf{D}$ *is a pair* $(\triangle_{\mathbf{D}}, \Phi_{\mathbf{D}})$*, where* $\triangle_{\mathbf{D}}$ *is the* domain set *of* $\mathbf{D}$*, and* $\Phi_{\mathbf{D}}$ *is a set of* concrete predicates*. Each* $d \in \Phi_{\mathbf{D}}$ *is associated with an* extension $d^{\mathbf{D}} \subseteq \triangle_{\mathbf{D}}$*. A concrete domain* $\mathbf{D}$ *is* admissible *if (*i*)* $\Phi_{\mathbf{D}}$ *is closed under negation (i.e., for each* $d \in \Phi_{\mathbf{D}}$*, a predicate* $\overline{d} \in \Phi_{\mathbf{D}}$ *exists such that* $\overline{d}^{\mathbf{D}} = \triangle_{\mathbf{D}} \setminus d^{\mathbf{D}}$*), (*ii*)* $\Phi_{\mathbf{D}}$ *contains a unary predicate* $\top_{\mathbf{D}}$ *interpreted as* $\triangle_{\mathbf{D}}$*, and (*iii*)* $\mathbf{D}$*-satisfiability of finite conjunctions of the form* $\bigwedge_{i=1}^{n} d_i(x_i)$ *is decidable in exponential time. (A conjunction of concrete predicates is* $\mathbf{D}$*-satisfiable if and only if an assignment* δ *of variables* x_i *to elements of* $\triangle_{\mathbf{D}}$ *exists such that* $\delta(x_i) \in d_i^{\mathbf{D}}$ *for each* $1 \leq i \leq n$*.)*

We now define the variant of the DL syntax that provides for metamodeling.

Definition 4 (Syntax)**.** *Let* $\mathbf{D}$ *be an admissible concrete domain,* N *a set of* names*, and* N_{I_c} *a set of* concrete individuals*, such that* $\Phi_{\mathbf{D}}$*,* N*, and* N_{I_c} *are all pairwise disjoint. The set of* roles *is defined as* $N \cup \{n^- \mid n \in N\}$*, and the roles* n^- *are called* inverse roles*. For each* $n \in N$*, let* $\mathsf{Inv}(n) = n^-$ *and* $\mathsf{Inv}(n^-) = n$*.*

A $\mathcal{SHOIQ}(\mathbf{D})$ *RBox* $KB_{\mathcal{R}}$ *is a finite set of* transitivity axioms $\mathsf{Trans}(S)$*,* abstract role inclusions $S \sqsubseteq_a T$*, and* concrete role inclusions $U \sqsubseteq_c V$*, where* S *and* T *are roles, and* U *and* V *are names. Let* $\sqsubseteq_a^*$ *be the reflexive-transitive closure of* $\{S \sqsubseteq_a T, \mathsf{Inv}(S) \sqsubseteq_a \mathsf{Inv}(T) \mid S \sqsubseteq_a T \in KB_{\mathcal{R}}\}$*. A role* S *is* transitive *in* $KB_{\mathcal{R}}$ *if a role* T *exists such that* $T \sqsubseteq_a^* S$*,* $S \sqsubseteq_a^* T$*, and either* $\mathsf{Trans}(T) \in KB_{\mathcal{R}}$ *or* $\mathsf{Trans}(\mathsf{Inv}(T)) \in KB_{\mathcal{R}}$*; furthermore,* S *is* simple *in* $KB_{\mathcal{R}}$ *if no transitive role* T *exists such that* $T \sqsubseteq_a^* S$*.*

The set of $\mathcal{SHOIQ}(\mathbf{D})$ concepts *over* $KB_{\mathcal{R}}$ *is defined by the following grammar, where* A*,* U *and* a *are names,* S *is a role,* T *is a simple role,* d *is a concrete predicate, and* m *is a nonnegative integer:*

$$\begin{aligned} D \quad \rightarrow \quad & \top \mid \bot \mid A \mid \{a\} \mid \neg D \mid D_1 \sqcap D_2 \mid D_1 \sqcup D_2 \mid \exists S.D \mid \forall S.D \mid \\ & \geq m\, T.D \mid \leq m\, T.D \mid \exists U.d \mid \forall U.d \mid \geq m\, U.d \mid \leq m\, U.d \end{aligned}$$

A $\mathcal{SHOIQ}(\mathbf{D})$ TBox $KB_{\mathcal{T}}$ is a finite set of concept inclusions *$D_1 \sqsubseteq_t D_2$, where D_1 and D_2 are $\mathcal{SHOIQ}(\mathbf{D})$ concepts. A $\mathcal{SHOIQ}(\mathbf{D})$ ABox $KB_{\mathcal{A}}$ is a finite set of* assertions *$D(a)$, $S(a, b)$, $U(a, b^{\mathsf{c}})$, or (in)equalities $a \approx b$, $a \not\approx b$, $a^{\mathsf{c}} \approx b^{\mathsf{c}}$, and $a^{\mathsf{c}} \not\approx b^{\mathsf{c}}$, where D is a $\mathcal{SHOIQ}(\mathbf{D})$ concept, S is a role, U, a, and b are names, and a^{c} and b^{c} are concrete individuals. A $\mathcal{SHOIQ}(\mathbf{D})$ knowledge base KB is a triple $(KB_{\mathcal{R}}, KB_{\mathcal{T}}, KB_{\mathcal{A}})$. With N_{KB} we denote the subset of those names that occur in KB.*

The logic $\mathcal{ALCHOIQ}(\mathbf{D})$ is obtained from $\mathcal{SHOIQ}(\mathbf{D})$ by disallowing transitivity axioms.

Definition 4 differs from the usual definitions of $\mathcal{SHOIQ}(\mathbf{D})$ (such as [1, 11]) in that it merges the sets of atomic concepts, atomic roles, and individuals into one set of names. This has important consequences, as we discuss next.

Metamodeling and Concrete Predicates. Unlike OWL Full, Definition 4 does not provide for metamodeling on concrete predicates—that is, the set of names is assumed to be disjoint from the set of concrete predicates or concrete individuals. This is mainly a technical assumption that makes defining the semantics easier. We do not believe that this is an important limitation.

Syntactic Ambiguities due to Metamodeling. In the usual definitions of the syntax of $\mathcal{SHOIQ}(\mathbf{D})$, the sets of concept, role, and individual names are mutually disjoint. Thus, axioms of the syntactic form $X \sqsubseteq Y$ can denote both concept and role inclusions, and the intended meaning is disambiguated by looking at the types of X and Y. With metamodeling this is not possible: X and Y can be names that are used as both concepts and roles, so, from its syntactic form, we cannot determine if the axiom represents a concept or a role inclusion. To remedy this, we introduce three different inclusion symbols: $\sqsubseteq_a$ represents inclusion for *abstract* roles (i.e., roles that connect two nonconcrete individuals), $\sqsubseteq_c$ represents inclusion for *concrete* roles (i.e., roles that connect a nonconcrete with a concrete individual), and $\sqsubseteq_t$ represents inclusion between

concepts. Note that syntactic ambiguities do not arise for concepts of the form $\exists S.D$ and $\exists U.d$ because the sets of names and concrete predicates are disjoint. (Similar comments apply to concept constructors $\forall$, $\geq$, and $\leq$.) Finally, assertions of the form $S(a, b)$ can be distinguished from assertions of the form $U(a, b^{\mathsf{c}})$ because the set of concrete individuals is assumed to be disjoint with the set of names.

Practical Problems with OWL Syntax. Three syntaxes for OWL have been defined so far: OWL RDF [3] encodes an OWL ontology using a set of RDF triples, OWL XML [8] encodes an OWL ontology as an XML document, and OWL Abstract Syntax [20] is a human-readable textual syntax. Without metamodeling, neither of these syntaxes suffers from the syntactic ambiguities, because the names used as concepts, individuals, abstract roles, and concrete roles are required to be disjoint. In contrast, only the OWL XML variant is compatible with metamodeling: for example, in OWL RDF, an inclusion between roles S and T is represented as the triple $\langle S, \mathit{owl{:}subPropertyOf}, T\rangle$ regardless of whether the inclusion is on abstract or on concrete roles. We hope that these details will be resolved in OWL 1.1[3]—a revision of OWL that is currently in the standardization process.

3.2 Two Semantics for $\mathcal{SHOIQ}(\mathbf{D})$ with Metamodeling

We now define two semantics for $\mathcal{SHOIQ}(\mathbf{D})$ that take metamodeling into account. We first present the *contextual* semantics.

Definition 5 (Contextual Semantics)**.** *For a $\mathcal{SHOIQ}(\mathbf{D})$ knowledge base KB, a π-interpretation I is a 5-tuple $(\triangle^I, \cdot^I, C_t^I, R_a^I, R_c^I)$ where*

- *$\triangle^I$ is a nonempty* domain set *disjoint with $\triangle_{\mathbf{D}}$,*
- *$\cdot^I$ is a* name interpretation function *that maps each element of N into an element of $\triangle^I$ and each element of N_{I_c} into an element of $\triangle_{\mathbf{D}}$,*

[3] `http://www.webont.org/owl/1.1/`

- $C_t^I : N \to 2^{\triangle^I}$ *is an* atomic concept extension function*,*
- $R_a^I : N \to 2^{\triangle^I \times \triangle^I}$ *is an* abstract role extension function*, and*
- $R_c^I : N \to 2^{\triangle^I \times \triangle_{\mathbf{D}}}$ *is a* concrete role extension function*.*

The functions R_a^I, R_c^I, and C_t^I are extended to roles and concepts as specified in Table 3, upper left section, by interpreting names contextually*—that is, depending on their syntactic position. A π-interpretation I is a π-model of KB if it satisfies all conditions from Table 3, lower left section. The notions of π-satisfiability, π-unsatisfiability, and π-entailment (written $\models_\pi$) are defined as usual.*

It is widely accepted that the separation of the concrete from the abstract domain (i.e., the requirement that $\triangle_{\mathbf{D}} \cap \triangle^I = \emptyset$) is semantically and practically justified. Therefore, we separate the abstract and the concrete role interpretation of a name—that is, for each name, there are two distinct role interpretations, which are referred to depending on the syntactic context in which the name is used. The set $\triangle^I$ is often called the *abstract domain*. Since the abstract and the concrete domains are disjoint, our logic is actually two-sorted. In the rest of this paper, with a we denote the abstract and with c the concrete sort.

The two-sorted nature of our semantics has certain consequences that might seem surprising. For example, the axioms $S(a, b)$ and $S \sqsubseteq_c T$ do not imply $T(a, b)$: the name S in $S(a, b)$ is interpreted as an abstract role, whereas the axiom $S \sqsubseteq_c T$ states that the interpretation of S as a concrete role is included into the interpretation of T as a concrete role. Also, note that $\mathsf{Trans}(S)$ makes only the abstract role interpretation of S transitive. This is because the separation of abstract and concrete role interpretations makes the definition of transitive concrete roles quite difficult.

The main difference between the π-semantics and the contextual semantics from [5] is that we consider only unary and binary predicates and that we use a multisorted logic to separate the abstract from the concrete roles. Despite

these minor technical differences, we call our semantics contextual because it was inspired by the contextual semantics from [5].

The contextual semantics can be understood as "punning" with names: we can eliminate metamodeling by replacing each name n with a distinct name $n_{concept}$, n_{ind}, $n_{a\text{-}role}$, or $n_{c\text{-}role}$, depending on the syntactic occurrence of n. In fact, n used as a concept and as an individuals are two different things that have nothing to do with each other semantically, even though they share the same name. Thus, the π-semantics is actually a variant of the first-order semantics, and we use it mainly as a baseline for a comparison with the HiLog semantics, which is more in the spirit of OWL Full.

Definition 6 (HiLog Semantics)**.** *For a $\mathcal{SHOIQ}(\mathbf{D})$ knowledge base KB, a ν-interpretation I is a 5-tuple $(\triangle^I, \cdot^I, C_t^I, R_a^I, R_c^I)$ where*

- $\triangle^I$ *is a nonempty* domain set *disjoint with $\triangle_{\mathbf{D}}$,*
- $\cdot^I$ *is a* name interpretation function *that maps each element of N into an element of $\triangle^I$ and each element of N_{I_c} into an element of $\triangle_{\mathbf{D}}$,*
- $C_t^I : \triangle^I \to 2^{\triangle^I}$ *is an* atomic concept extension function*,*
- $R_a^I : \triangle^I \to 2^{\triangle^I \times \triangle^I}$ *is an* abstract role extension function*, and*
- $R_c^I : \triangle^I \to 2^{\triangle^I \times \triangle_{\mathbf{D}}}$ *is a* concrete role extension function.

Table 3, right section, shows how to interpret roles, concepts, and axioms. The notions of ν-satisfiability, ν-unsatisfiability, and ν-entailment (written $\models_\nu$) are defined as usual.

Again, the ν-semantics differs from the HiLog semantics from [5] in that we consider only unary and binary predicates in a multisorted setting. Despite these technical differences, we call our semantics the HiLog semantics because it, just like HiLog, associates extensions with the elements of $\triangle^I$ rather than with the elements of N.

Table 3: Two Semantics for $\mathcal{SHOIQ}(\mathbf{D})$ with Metamodeling

π-Semantics		ν-Semantics
Interpretation of Abstract Roles R_a^I		
A	$R_a^I(A) \subseteq \triangle^I \times \triangle^I$	
A^-	$R_a^I(A)^-$	
Interpretation of Concepts C_t^I		
$\top$	$\triangle^I$	
$\bot$	$\emptyset$	
A	$C_t^I(A) \subseteq \triangle^I$	
$\{a\}$	$\{a^I\}$	
$\neg D$	$\triangle^I \setminus C_t^I(D)$	
$D_1 \sqcap D_2$	$C_t^I(D_1) \cap C_t^I(D_2)$	Interpretations of roles, concepts, and axioms are obtained from the ones for the π-semantics by performing the following changes:
$D_1 \sqcup D_2$	$C_t^I(D_1) \cup C_t^I(D_2)$	
$\exists S.D$	$\{x \mid \exists y : (x,y) \in R_a^I(S) \wedge y \in C_t^I(D)\}$	
$\forall S.D$	$\{x \mid \forall y : (x,y) \in R_a^I(S) \rightarrow y \in C_t^I(D)\}$	
$\leq m\, S.D$	$\{x \mid \sharp\{y \mid (x,y) \in R_a^I(S) \wedge y \in C_t^I(D)\} \leq m\}$	
$\geq m\, S.D$	$\{x \mid \sharp\{y \mid (x,y) \in R_a^I(S) \wedge y \in C_t^I(D)\} \geq m\}$	
$\exists U.d$	$\{x \mid \exists y : (x,y) \in R_c^I(U) \wedge y \in d^{\mathbf{D}}\}$	$C_t^I(A) \rightsquigarrow C_t^I(A^I)$
$\forall U.d$	$\{x \mid \forall y : (x,y) \in R_c^I(U) \rightarrow y \in d^{\mathbf{D}}\}$	$R_a^I(A) \rightsquigarrow R_a^I(A^I)$
$\leq m\, U.d$	$\{x \mid \sharp\{y \mid (x,y) \in R_c^I(U) \wedge y \in d^{\mathbf{D}}\} \leq m\}$	$R_c^I(U) \rightsquigarrow R_c^I(U^I)$
$\geq m\, U.d$	$\{x \mid \sharp\{y \mid (x,y) \in R_c^I(U) \wedge y \in d^{\mathbf{D}}\} \geq m\}$	$R_c^I(V) \rightsquigarrow R_c^I(V^I)$
Interpretation of Axioms		
$S \sqsubseteq_a T$	$R_a^I(S) \subseteq R_a^I(T)$	
$U \sqsubseteq_c V$	$R_c^I(U) \subseteq R_c^I(V)$	
$D_1 \sqsubseteq_t D_2$	$C_t^I(D_1) \subseteq C_t^I(D_2)$	
$\mathsf{Trans}(S)$	$R_a^I(S)^+ \subseteq R_a^I(S)$	
$D(a)$	$a^I \in C_t^I(D)$	
$S(a,b)$	$(a^I, b^I) \in R_a^I(S)$	
$U(a, b^{\mathsf{c}})$	$(a^I, (b^{\mathsf{c}})^I) \in R_c^I(U)$	
$a^{(\mathsf{c})} \approx b^{(\mathsf{c})}$	$(a^{(\mathsf{c})})^I = (b^{(\mathsf{c})})^I$	
$a^{(\mathsf{c})} \not\approx b^{(\mathsf{c})}$	$(a^{(\mathsf{c})})^I \neq (b^{(\mathsf{c})})^I$	

Note: $\sharp N$ is the number of elements in N; R^+ is the transitive closure of R; R^- is the inverse relation of R; A, U, and V are names; and S and T are roles.

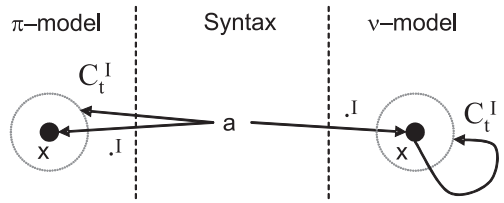

Figure 2: π- and ν-Models of the Axiom $a(a)$

To understand the essential difference between the π- and the ν-semantics, consider the knowledge base KB containing only the axiom $a(a)$, where the name a is used both as an individual and as a concept. A π-model of KB is depicted on the left-hand side of Figure 2: both the individual interpretation $\cdot^I$ and the concept interpretation C^I are assigned directly to the name a. A ν-model of KB is depicted on the right-hand side of Figure 2: the individual interpretation $\cdot^I$ assigns the domain individual x to the name a; however, the concept interpretation is not assigned to a, but to x. We discuss in Section 4 the consequences that such a definition has on entailment.

Neither the π- nor the ν-semantics requires different names to be interpreted as different domain objects. If this is required, the unique name assumption (UNA) should be axiomatized explicitly, by introducing an axiom $n_i \not\approx n_j$ for each $n_i, n_j \in N$ such that $n_i \neq n_j$.

3.3 Relationship with First-Order Logic

In this section, we show how to encode a $\mathcal{SHOIQ}(\mathbf{D})$ knowledge base KB into a first-order formula $\nu(KB)$ that is equivalent to KB under the ν-semantics. This not only provides additional insight into the nature of the ν-semantics, but it also allows one to apply well-known methods for first-order reasoning, such as the resolution calculus.

The idea behind our translation can be traced back to [22], and it is similar to the encoding of HiLog formulae into first-order logic. We translate names into constants of sort a, and we represent the functions C_t^I, R_a^I, and R_c^I using

predicates isa, arole, and crole, respectively. The predicate isa is of sort a, the predicate arole is of sort $\mathsf{a} \times \mathsf{a}$, and the predicate crole is of sort $\mathsf{a} \times \mathsf{c}$. To stipulate that the logic is two-sorted, we use an equality predicate $\approx$ for the abstract domain and a distinct equality predicate $\approx_{\mathbf{D}}$ for the concrete domain. To denote that a variable is of concrete sort, we write it as y^{c}. Our translation uses counting quantifiers $\exists^{\leq n}$ and $\exists^{\geq n}$, which can be encoded in first-order logic as follows, for $\mathbf{y}$ a vector of variables:

$$\exists^{\geq n} x : \varphi(x, \mathbf{y}) = \exists x_1, \ldots, x_n : \left[\bigwedge_{i=1}^{n} \varphi(x_i, \mathbf{y}) \wedge \bigwedge_{1 \leq i < j \leq n} x_i \not\approx x_j \right] \tag{12}$$

$$\exists^{\leq n} x : \varphi(x, \mathbf{y}) = \forall x_1, \ldots, x_{n+1} : \left[\bigwedge_{i=1}^{n+1} \varphi(x_i, \mathbf{y}) \rightarrow \bigvee_{1 \leq i < j \leq n+1} x_i \approx x_j \right] \tag{13}$$

The encoding is analogous if x is a concrete variable, with the difference that one should use the concrete equality $\approx_{\mathbf{D}}$ instead of the abstract equality $\approx$.

Definition 7. *For a $\mathcal{SHOIQ}(\mathbf{D})$ knowledge base KB, let $\nu(KB)$ be the translation of KB into first-order formulae as specified in Table 4.*

Theorem 3. *A $\mathcal{SHOIQ}(\mathbf{D})$ knowledge base KB is ν-satisfiable if and only if a first-order model of $\nu(KB)$ exists.*

Proof. For the ($\Rightarrow$) direction, let I_ν be a ν-model of KB. We construct a first-order interpretation I for $\nu(KB)$ by setting $\triangle^I = \triangle^{I_\nu}$, $a^I = a^{I_\nu}$, $(x, y) \in \mathsf{isa}^I$ iff $y \in C_t^{I_\nu}(x)$, $(x, y, z) \in \mathsf{arole}^I$ iff $(y, z) \in R_a^{I_\nu}(x)$, and $(x, y, z) \in \mathsf{crole}^I$ iff $(y, z) \in R_c^{I_\nu}(x)$. By induction on the structure of formulae in $\nu(KB)$, one can easily show that I is a model of $\nu(KB)$. The ($\Leftarrow$) direction is similar. □

4 Expressivity of Metamodeling

We now discuss the benefits of metamodeling in terms of additional consequences that can be drawn. These results are similar to the ones for HiLog from [5].

Table 4: Mapping ν-Semantics into First-order Logic

Mapping Roles to FOL	
$\nu^a_{xy}(A) = \mathsf{arole}(A, x, y)$	$\nu^a_{yx}(A) = \mathsf{arole}(A, y, x)$
$\nu^a_{xy}(A^-) = \mathsf{arole}(A, y, x)$	$\nu^a_{yx}(A^-) = \mathsf{arole}(A, x, y)$
$\nu^c_{xy}(U) = \mathsf{crole}(U, x, y^{\mathsf{c}})$	$\nu^c_{yx}(U) = \mathsf{crole}(U, y, x^{\mathsf{c}})$
Mapping Concepts to FOL	
$\nu_x(\top) = \mathsf{true}$	$\nu_y(\top) = \mathsf{true}$
$\nu_x(\bot) = \mathsf{false}$	$\nu_y(\bot) = \mathsf{false}$
$\nu_x(A) = \mathsf{isa}(A, x)$	$\nu_y(A) = \mathsf{isa}(A, y)$
$\nu_x(\{a\}) = x \approx a$	$\nu_y(\{a\}) = y \approx a$
$\nu_x(\neg D) = \neg\nu_x(D)$	$\nu_y(\neg D) = \neg\nu_y(D)$
$\nu_x(D_1 \sqcap D_2) = \nu_x(D_1) \wedge \nu_x(D_2)$	$\nu_y(D_1 \sqcap D_2) = \nu_y(D_1) \wedge \nu_y(D_2)$
$\nu_x(D_1 \sqcup D_2) = \nu_x(D_1) \vee \nu_x(D_2)$	$\nu_y(D_1 \sqcup D_2) = \nu_y(D_1) \vee \nu_y(D_2)$
$\nu_x(\forall S.D) = \forall y : \nu^a_{xy}(S) \rightarrow \nu_y(D)$	$\nu_y(\forall S.D) = \forall x : \nu^a_{yx}(S) \rightarrow \nu_x(D)$
$\nu_x(\exists S.D) = \exists y : \nu^a_{xy}(S) \wedge \nu_y(D)$	$\nu_y(\exists S.D) = \exists x : \nu^a_{yx}(S) \wedge \nu_x(D)$
$\nu_x(\leq m\, S.D) = \exists^{\leq m} y : [\nu^a_{xy}(S) \wedge \nu_y(D)]$	$\nu_y(\leq m\, S.D) = \exists^{\leq m} x : [\nu^a_{yx}(S) \wedge \nu_x(D)]$
$\nu_x(\geq m\, S.D) = \exists^{\geq m} y : [\nu^a_{xy}(S) \wedge \nu_y(D)]$	$\nu_y(\geq m\, S.D) = \exists^{\geq m} x : [\nu^a_{yx}(S) \wedge \nu_x(D)]$
$\nu_x(\forall S.d) = \forall y^{\mathsf{c}} : \nu^c_{xy}(S) \rightarrow d(y^{\mathsf{c}})$	$\nu_y(\forall S.d) = \forall x^{\mathsf{c}} : \nu^c_{yx}(S) \rightarrow d(x^{\mathsf{c}})$
$\nu_x(\exists S.d) = \exists y^{\mathsf{c}} : \nu^c_{xy}(S) \wedge d(y^{\mathsf{c}})$	$\nu_y(\exists S.d) = \exists x^{\mathsf{c}} : \nu^c_{yx}(S) \wedge d(x^{\mathsf{c}})$
$\nu_x(\leq m\, S.d) = \exists^{\leq m} y^{\mathsf{c}} : [\nu^c_{xy}(S) \wedge d(y^{\mathsf{c}})]$	$\nu_y(\leq m\, S.d) = \exists^{\leq m} x^{\mathsf{c}} : [\nu^c_{yx}(S) \wedge d(x^{\mathsf{c}})]$
$\nu_x(\geq m\, S.d) = \exists^{\geq m} y^{\mathsf{c}} : [\nu^c_{xy}(S) \wedge d(y^{\mathsf{c}})]$	$\nu_y(\geq m\, S.d) = \exists^{\geq m} x^{\mathsf{c}} : [\nu^c_{yx}(S) \wedge d(x^{\mathsf{c}})]$
Mapping Axioms to FOL	
$\nu(\mathsf{Trans}(A)) = \forall x, y, z : \mathsf{arole}(A, x, y) \wedge \mathsf{arole}(A, y, z) \rightarrow \mathsf{arole}(A, x, z)$	
$\nu(\mathsf{Trans}(A^-)) = \forall x, y, z : \mathsf{arole}(A, y, x) \wedge \mathsf{arole}(A, z, y) \rightarrow \mathsf{arole}(A, z, x)$	
$\nu(S \sqsubseteq_a T) = \forall x, y : \nu^a_{xy}(S) \rightarrow \nu^a_{xy}(T)$	$\nu(U \sqsubseteq_c V) = \forall x, y^{\mathsf{c}} : \nu^c_{xy}(U) \rightarrow \nu^c_{xy}(V)$
$\nu(D_1 \sqsubseteq_t D_2) = \forall x : \nu_x(D_1) \rightarrow \nu_x(D_2)$	$\nu(D(a)) = \nu_x(D)\{x \mapsto a\}$
$\nu(S(a, b)) = \nu^a_{xy}(S)\{x \mapsto a, y \mapsto b\}$	$\nu(U(a, b^{\mathsf{c}})) = \nu^c_{xy}(U)\{x \mapsto a, y^{\mathsf{c}} \mapsto b^{\mathsf{c}}\}$
$\nu(a \approx b) = a \approx b$	$\nu(a \not\approx b) = a \not\approx b$
$\nu(a^{\mathsf{c}} \approx b^{\mathsf{c}}) = a^{\mathsf{c}} \approx_{\mathbf{D}} b^{\mathsf{c}}$	$\nu(a^{\mathsf{c}} \not\approx b^{\mathsf{c}}) = a^{\mathsf{c}} \not\approx_{\mathbf{D}} b^{\mathsf{c}}$
Mapping KB to FOL	
$\nu(KB_{\mathcal{R}}) = \bigwedge_{\alpha \in KB_{\mathcal{R}}} \nu(\alpha)$	
$\nu(KB_{\mathcal{T}}) = \bigwedge_{\alpha \in KB_{\mathcal{T}}} \nu(\alpha)$	
$\nu(KB_{\mathcal{A}}) = \bigwedge_{\alpha \in KB_{\mathcal{A}}} \nu(\alpha)$	
$\nu(KB) = \nu(KB_{\mathcal{R}}) \wedge \nu(KB_{\mathcal{T}}) \wedge \nu(KB_{\mathcal{A}})$	

Note: A, U, V, a, and b are names, $D_{(i)}$ are concepts, S and T are roles, m is a nonnegative integer, b^{c} is a concrete individual, and $\{x \mapsto a, y^{(\mathsf{c})} \mapsto b^{(\mathsf{c})}\}$ is a mapping of the variables x and $y^{(\mathsf{c})}$ to a and $b^{(\mathsf{c})}$, respectively.

It is easy to see that ν-satisfiability is a strictly stronger notion than π-satisfiability. Consider the following knowledge base KB:[4]

$$Eagle(Harry) \tag{14}$$

$$\neg Aquila(Harry) \tag{15}$$

$$Eagle \approx Aquila \tag{16}$$

Under the contextual semantics, the interpretations of the names *Eagle* and *Aquila* as concepts and as individuals are independent, so KB is π-satisfiable. KB, however, is ν-unsatisfiable: in each ν-interpretation $Eagle^I = Aquila^I = \alpha$, so it cannot be that $Harry^I \in C^I(Eagle^I)$ and $Harry^I \notin C^I(Aquila^I)$. For the other direction, we have the following theorem:

Theorem 4. *Each ν-satisfiable $\mathcal{SHOIQ}(\mathbf{D})$ knowledge base KB is also π-satisfiable.*

Proof. Let I_ν be a ν-model of KB. We construct from I_ν a π-interpretation I_π as follows: $\triangle^{I_\pi} = \triangle^{I_\nu}$, $n^{I_\pi} = n^{I_\nu}$, $C_t^{I_\pi}(n) = C_t^{I_\nu}(n^{I_\nu})$, $R_a^{I_\pi}(n) = R_a^{I_\nu}(n^{I_\nu})$, and $R_c^{I_\pi}(n) = R_c^{I_\nu}(n^{I_\nu})$ for each $n \in N$. By a simple induction on the concept structure, it can be shown that $C_t^{I_\pi}(X) = C_t^{I_\nu}(X)$ for each concept X, so I_π is a π-model of KB. □

Furthermore, for a knowledge base with unique name assumption or without equality (either explicit or implicit—for example, introduced through number restrictions or nominals), π- and ν-satisfiability coincide:

Theorem 5. *Let KB be an $\mathcal{SHOIQ}(\mathbf{D})$ knowledge base that either employs the unique name assumption or contains no explicit equality statements, number restrictions, or nominals. Then, KB is π-satisfiable if and only if it is ν-satisfiable.*

Proof. The ($\Leftarrow$) direction follows from Theorem 4. For the ($\Rightarrow$) direction, let KB be π-satisfiable in some model I_π. Due to the theorem's assumptions, we

[4] Aquila is the Latin name for eagle.

can assume without loss of generality that $n_i \neq n_j$ implies ${n_i}^{I_\pi} \neq {n_j}^{I_\pi}$ for all for $n_i, n_j \in N$. (This holds trivially if KB employs the unique name assumption. If KB does not employ equality, then we can interpret KB in a Herbrand model, for which the required property clearly holds as well.)

For such a π-model I_π, we build a ν-interpretation I_ν as follows: $\triangle^{I_\nu} = \triangle^{I_\pi}$, $n^{I_\nu} = n^{I_\pi}$, $C_t^{I_\nu}(n^{I_\nu}) = C_t^{I_\pi}(n)$, $R_a^{I_\nu}(n^{I_\nu}) = R_a^{I_\pi}(n)$, and $R_c^{I_\nu}(n^{I_\nu}) = R_c^{I_\pi}(n)$ for $n \in N$. Furthermore, for all $x \in \triangle^{I_\nu}$ such that there is no $n \in N$ with $x = a^{I_\nu}$, let $C_t^{I_\nu}(x) = R_a^{I_\nu}(x) = R_c^{I_\nu}(x) = \emptyset$. Since we can assume that different names from N are interpreted as different elements from $\triangle^{I_\nu}$, such a construction defines exactly one value for $C_t^{I_\nu}(x)$, $R_a^{I_\nu}(x)$, and $R_c^{I_\nu}(x)$ for each $x \in \triangle^{I_\nu}$, so I_ν is correctly defined. By a simple induction on the concept structure, it can be shown that $C_t^{I_\nu}(X) = C_t^{I_\pi}(X)$ for each concept X, so I_ν is a ν-model of KB. □

Note that, for Theorem 5 to hold, KB must employ the unique name assumption on all names, even if they are used only as concepts or roles. For example, the axioms $\top \sqsubseteq \{a\}$ and $C(a)$ entail $a(a)$ under the ν-semantics, but not under the π-semantics. Since a is the only individual, the choice of UNA seems immaterial, so this example seems to invalidate Theorem 5. To satisfy the conditions of Theorem 5, however, we must additionally assert $C \not\approx a$, which makes the knowledge base unsatisfiable under both semantics.

To summarize, for a knowledge base KB, all consequences of the π-semantics are derivable by the ν-semantics as well; furthermore, the latter derives additional consequences only if two names can be derived to be equal. For example, from (14) and (16) we can derive $Aquila(Harry)$. Furthermore, if the unique name assumption is employed, the ν-semantics does not yield any additional consequences over the π-semantics. This seems to suggest that the benefits of the ν-semantics do not outweigh the fact that this semantics is nonstandard, and that the π-semantics might be sufficient for many practical applications.

The ν-semantics, however, becomes useful if combined with expressive extensions of OWL. For example, by combining the ν-semantics with SWRL, one can explicitly axiomatize the semantics of metaconcepts. Consider the example

from Section 1. By (17) we state that *Eagle* is a *RedListSpecies*, and by the SWRL rule (18) we state that instances of species listed in the Red List are not allowed to be hunted. Notice that, in the atom $S(I)$, we use the variable S in the position of a predicate. Under the ν-semantics, such an atom is equivalent to $\mathsf{isa}(S, I)$; however, to give meaning to such an atom under π-semantics, we would need to use a higher-order logic. From (14), (17), and (18), we can now infer *CannotHunt*(*Harry*), so *RedListSpecies* truly acts as a metaconcept for the concept *Eagle*.

$$RedListSpecies(Eagle) \tag{17}$$

$$RedListSpecies(S) \wedge S(I) \rightarrow CannotHunt(I) \tag{18}$$

To summarize, we believe that the ν-semantics provides a sound basis for metamodeling, which, when combined with expressive logical formalisms such as SWRL, allows us to precisely axiomatize the semantic interaction between concepts and metaconcepts. Therefore, we consider this semantics very relevant for future extensions of OWL.

5 Reasoning with the ν-Semantics

Since the contextual semantics is essentially first-order, it can be decided using known algorithms, such as [11, 13]. In this section we present an algorithm for deciding ν-satisfiability. We first consider $\mathcal{ALCHOIQ}(\mathbf{D})$ knowledge bases in Section 5.1, and then discuss the problems introduced by transitivity axioms in Section 5.2. Finally, in Section 5.3 we discuss how the embedding of the ν-semantics into first-order logic from Section 3.3 can be used to extend existing resolution decision procedures to obtain practical reasoning algorithms.

5.1 Deciding ν-Satisfiability of $\mathcal{ALCHOIQ}(\mathbf{D})$

Before presenting a decision procedure, we introduce some notation. Let $\mathcal{E}$ be an equivalence relation over a set of names N_{KB}. For each equivalence class of $\mathcal{E}$, we arbitrarily select one *representative name*. For a name n, let $n/_{\mathcal{E}}$ denote the representative name chosen for the equivalence class of n; similarly, for an $\mathcal{ALCHOIQ}(\mathbf{D})$ concept (axiom) α, let $\alpha/_{\mathcal{E}}$ denote the concept (axiom) obtained from α by replacing each name n with $n/_{\mathcal{E}}$. Finally, for an $\mathcal{ALCHOIQ}(\mathbf{D})$ knowledge base KB, with $KB/_{\mathcal{E}}$ we denote the knowledge base obtained from KB by (i) replacing each axiom α with $\alpha/_{\mathcal{E}}$, and by (ii) appending an axiom $n_i/_{\mathcal{E}} \not\approx n_j/_{\mathcal{E}}$ for each pair of names $n_i, n_j \in N_{KB}$ such that $n_i/_{\mathcal{E}} \neq n_j/_{\mathcal{E}}$.

An algorithm for checking ν-satisfiability of KB is relatively simple: it nondeterministically guesses an equivalence relation $\mathcal{E}$ over N_{KB} and then checks π-satisfiability of $KB/_{\mathcal{E}}$ (under UNA). Soundness and completeness of the algorithm follow immediately from the following lemma:

Lemma 2. *An $\mathcal{ALCHOIQ}(\mathbf{D})$ knowledge base KB is ν-satisfiable if and only if an equivalence relation $\mathcal{E}$ over N_{KB} exists such that $KB/_{\mathcal{E}}$ is π-satisfiable.*

Proof. ($\Leftarrow$) Since the knowledge base $KB/_{\mathcal{E}}$ employs the unique name assumption, it is π-satisfiable if and only if it is ν-satisfiable by Theorem 5. Let I'_ν be a ν-model of $KB/_{\mathcal{E}}$, and let I_ν be a model obtained from I'_ν by setting $n^{I_\nu} = (n/_{\mathcal{E}})^{I'_\nu}$ for each $n \in N_{KB}$. Clearly, I_ν is a ν-model of KB.

($\Rightarrow$) For a ν-model I_ν of KB, let $\mathcal{E}$ be the equivalence relation on names defined as $\mathcal{E} = \{(n_i, n_j) \mid {n_i}^{I_\nu} = {n_j}^{I_\nu} \text{ for } n_i, n_j \in N_{KB}\}$. Clearly, I_ν is a ν-model of $KB/_{\mathcal{E}}$. Furthermore, $KB/_{\mathcal{E}}$ employs the unique name assumption, so it is π-satisfiable by Theorem 5. □

Please note that Lemma 2 holds even if KB is a $\mathcal{SHOIQ}(\mathbf{D})$ knowledge base. As we discuss in Section 5.2, however, the following result holds only for $\mathcal{ALCHOIQ}(\mathbf{D})$:

Theorem 6. *Checking ν-satisfiability of an $\mathcal{ALCHOIQ}(\mathbf{D})$ knowledge base KB can be performed in nondeterministic exponential time.*

Proof. Observe that the number of elements in N_{KB} is linear in the size of KB, and that each equivalence relation $\mathcal{E}$ is a subset of $N_{KB} \times N_{KB}$. Hence, the number of possible equivalence relations is exponential in the size of KB. A decision procedure for ν-satisfiability can systematically examine all equivalence relations $\mathcal{E}$ and, for each one, perform a π-satisfiability check of $KB/_{\mathcal{E}}$. The last step can be performed in NExpTime: the logic $\mathcal{ALCHOIQ}$, obtained from $\mathcal{ALCHOIQ}(\mathbf{D})$ by disallowing concrete domains, is a fragment of $\mathcal{C}^2$ (the two-variable first-order logic with counting), which is decidable in NExpTime [21] even for binary coding of numbers; furthermore, extending ExpTime-hard logics with a restricted form of a concrete domain (i.e., a concrete domain without feature chains) does not increase the complexity of reasoning (please refer to Appendix A for more information). Thus, our algorithm makes an exponential number of calls to an algorithm in NExpTime, which gives a total complexity of NExpTime as well. □

We briefly compare the results of Theorem 6 with that of Theorems 1 and 2. The main feature of the ν-semantics is the reification of concept and role names. The ν-semantics, however, is more like the π-semantics and less like the OWL Full semantics in the way it handles the modeling primitives. In the ν- and the π-semantics, the modeling primitives are expressed as formulae, and are not accessible as individuals in the knowledge base. In contrast, the modeling primitives in OWL Full are reified, so they can be used in axioms.

5.2 HiLog Semantics and Transitivity

The algorithm for deciding ν-satisfiability is essentially based on an algorithm for deciding π-satisfiability, and the latter one can easily be extended to support transitive roles. Therefore, one might intuitively expect that the algorithm from Section 5.1 can easily be extended to handle transitive roles as well. Consider, however, the following knowledge base KB:

$$\top \sqsubseteq \geq 3\, S \tag{19}$$

(20) $$S \approx T$$

(21) $$\mathsf{Trans}(T)$$

Note that the role S is simple (it is neither transitive nor does it have transitive subroles, so it passes the syntactic criterion from Definition 4); hence, KB is a well-formed $\mathcal{SHOIQ}(\mathbf{D})$ knowledge base. In any ν-interpretation I, however, the axiom (20) ensures that $S^I = T^I = \alpha$; furthermore, $R_a^I(\alpha)$ is transitive because of (21). Effectively, a transitive role is used in a number restriction in the axiom (19), even though S is syntactically a simple role.

Equality of role names might be nontrivially entailed by KB, so identifying simple roles would itself require theorem proving. Thus, defining simple roles is difficult, if not impossible under the ν-semantics. Allowing transitive roles in number restrictions is known to lead to undecidability [12], so we get the following result:

Proposition 1. *Checking ν-satisfiability of a $\mathcal{SHOIQ}(\mathbf{D})$ knowledge base KB is undecidable.*

Intuitively, Theorem 6 does not apply to $\mathcal{SHOIQ}(\mathbf{D})$ because, even if KB contain only simple roles in number restrictions, the equivalence relation $\mathcal{E}$ can be such that $KB/_{\mathcal{E}}$ contains nonsimple roles in number restrictions. Hence, we cannot decide satisfiability of $KB/_{\mathcal{E}}$ for every choice of $\mathcal{E}$.

Decidability can be regained if we ensure that the names used as roles are interpreted as distinct domain individuals.

Definition 8 (Unique Role Assumption)**.** *A $\mathcal{SHOIQ}(\mathbf{D})$ knowledge base KB employs the* unique role assumption *(URA) if it contains an axiom $S \not\approx T$ for each two distinct names S and T used as roles in KB.*

If KB does not contain explicit equality statements, number restrictions, or nominals, then we can always assume URA without affecting satisfiability of KB. Furthermore, if KB employs URA, then the interpretations of different role names are independent, so we can check for simple roles as usual. Thus,

we can check the satisfiability of KB using the algorithm from Section 5.1, with a minor modification that we do not need to consider the equivalences $\mathcal{E}$ that are not compatible with URA (for such $\mathcal{E}$, the knowledge base $KB/_{\mathcal{E}}$ is trivially unsatisfiable). Due to URA, if KB does not contain nonsimple roles in number restrictions, then $KB/_{\mathcal{E}}$ does not contain them either. Therefore, Theorem 6 carries over to $\mathcal{SHOIQ}(\mathbf{D})$ with URA without problems.

5.3 Practical Reasoning with the ν-Semantics

The reasoning procedure from Section 5.1 is worst-case optimal, but it is unlikely to be effective in practice, since it systematically examines exponentially many equivalence relations. Goal-oriented decision procedures for the ν-semantics can be obtained by slightly modifying several existing procedures for the π-semantics based on the embedding of the ν-semantics into first-order logic from Section 3.3. For example, a resolution-based decision procedure for the DL $\mathcal{SHIQ}(\mathbf{D})$ under the π-semantics was developed in [16], and an extension to the logic $\mathcal{SHOIQ}$ was presented in [13]. These procedures translate a DL knowledge base KB into a first-order formula $\pi(KB)$, transform it into a set of clauses, and then apply resolution to the clauses to determine satisfiability.

These procedures can easily be adapted to the ν-semantics by transforming KB into $\nu(KB)$ instead of $\pi(KB)$. In [16] we have shown that, with a couple of technical assumptions, this does not change the nature of the original decision procedure for the π-semantics. Since the resolution procedure for the π-semantics is relatively complex, we do not include further details into this paper; for further information, please refer to [16]. We believe that the same approach is applicable to the procedure from [13] as well.

6 Related Work

The definition of ν-satisfiability given in Section 3 is inspired by HiLog [5], a logic in which general terms are allowed to occur in place of function and predicate

symbols in formulae. The authors show that HiLog can be considered "syntactic sugar," since each HiLog formula can be encoded into an equisatisfiable first-order formula. The definition of the ν operator from Table 3 closely resembles this encoding. Finally, the authors show that a satisfiable first-order formula without equality is also satisfiable under the HiLog semantics.

In [18], the RDFS Model Theory was criticized for allowing an unlimited number of meta-layers in a model. The authors argue that such a semantics is inappropriate for the Semantic Web because (*i*) it does not provide adequate support for inferencing, (*ii*) it allows the definition of classes that contain themselves, and (*iii*) by adding classes, one necessarily introduces objects in the interpretation universe. The authors propose RDFS-FA, a stratified four-level approach, consisting of the meta-language layer, the language layer, the ontology layer, and the instance layer. In [9] similar arguments were used to criticize the semantics of OWL Full. Our approach follows the principles of RDFS-FA by strictly separating the modeling primitives from the ontology and the instance layers. To allow for metamodeling, however, our definition of the ν-semantics merges the ontology and the instance layers into one layer. Furthermore, we believe that features (*ii*) and (*iii*) are to be expected of a logic that provides for metamodeling.

In [25], the authors argue that metamodeling is useful in many applications. There, the modeling layers are connected using *spanning instances*; however, no logical framework capturing their semantics was presented.

7 Conclusion

In this paper we have analyzed the metamodeling features of OWL Full, the most expressive of the OWL family of Semantic Web ontology languages. We have shown that the style of metamodeling adopted in OWL Full leads to the undecidability of basic reasoning problems, because it allows us to state axioms about the built-in vocabulary. To obtain a decidable and expressive language

that supports metamodeling, we have proposed two alternative semantics: the contextual one, which is essentially first-order, and the HiLog one, which is more in the spirit of OWL Full. We have shown that, under certain technical assumptions, both semantics are decidable when combined with the description logic $\mathcal{SHOIQ}(\mathbf{D})$ that underpins OWL DL.

We have analyzed the added expressivity of metamodeling and have shown that the HiLog semantics allows one to derive new conclusions only via equality reasoning. This approach, however, becomes useful if it is combined with expressive extensions, such as SWRL, since it allows us to axiomatize the logical interaction between concepts and their metaconcepts.

Acknowledgment

We thank Evgeny Zolin for numerous invaluable comments about this paper.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, January 2003.

[2] F. Baader and P. Hanschke. A Scheme for Integrating Concrete Domains into Concept Languages. In *Proc. IJCAI '91*, pages 452–457, Sydney, Australia, 1991.

[3] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference, W3C Recommendation, February 10 2004. `http://www.w3.org/TR/owl-ref/`.

[4] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer, 1996.

[5] W. Chen, M. Kifer, and D. S. Warren. A Foundation for Higher-Order Logic Programming. *Journal of Logic Programming*, 15(3):187–230, 1993.

[6] V. Haarslev and R. Möller. RACER System Description. In *Proc. IJCAR 2001*, pages 701–706, Siena, Italy, 2001.

[7] P. Hayes. RDF Semantics, February 10 2004. `http://www.w3.org/TR/rdf-mt/`.

[8] M. Hori, J. Euzenat, and P. F. Patel-Schneider. OWL Web Ontology Language: XML Presentation Syntax, W3C Note, June 11 2003. `http://www.w3.org/TR/owl-xmlsyntax/`.

[9] I. Horrocks and P. F. Patel-Schneider. Three Theses of Representation in the Semantic Web. In *Proc. WWW 2003*, pages 39–47, Budapest, Hungary, May 20–24 2003.

[10] I. Horrocks and P. F. Patel-Schneider. A Proposal for an OWL Rules Language. In *Proc. WWW 2004*, pages 723–731, New York, NY, USA, May 17–22 2004.

[11] I. Horrocks and U. Sattler. A Tableaux Decision Procedure for $\mathcal{SHOIQ}$. In *Proc. IJCAI 2005*, pages 448–453, Edinburgh, UK, 2005.

[12] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–263, 2000.

[13] Y. Kazakov and B. Motik. A Resolution-Based Decision Procedure for $\mathcal{SHOIQ}$. In *Proc. IJCAR 2006*, Seattle, WA, USA, 2006. To appear.

[14] G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax, February 10 2004. http://www.w3.org/TR/rdf-concepts/.

[15] B. Motik. On the Properties of Metamodeling in OWL. In *Proc. ISWC 2005*, pages 548–562, Galway, Ireland, 2005.

[16] B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, University of Karlsruhe, Germany, 2006.

[17] B. Motik and U. Sattler. A Comparison of Techniques for Querying Large Description Logic ABoxes. In *Proc. LPAR 2006*, Phnom Penh, Cambodia, 2006. To appear.

[18] J. Z. Pan and I. Horrocks. RDFS(FA) and RDF MT: Two Semantics for RDFS. In *Proc. ISWC 2003*, pages 30–46, Sanibel Island, FL, USA, 2003.

[19] B. Parsia and E. Sirin. Pellet: An OWL-DL Reasoner. Poster, In Proc. ISWC 2004, Hiroshima, Japan, 2004.

[20] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language: Semantics and Abstract Syntax, W3C Recommendation, February 10 2004. `http://www.w3.org/TR/owl-semantics/`.

[21] I. Pratt-Hartmann. Complexity of the Two-Variable Fragment with Counting Quantifiers. *Journal of Logic, Language and Information*, 14(3):369–395, 2005.

[22] W. V. O. Quine. *Elementary Logic: Revised Edition*. Harvard University Press, 2005.

[23] G. Schreiber. The Web is not well-formed. *IEEE Intelligent Systems*, 17(2):79–80, 2002. Contribution to the section "Trends & Controversies: Ontologies KISSES in Standardization", edited by S. Staab.

[24] D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. IJCAR 2006*, Seattle, WA, USA, 2006. To appear.

[25] C. Welty and D. Ferrucci. What's in an instance? Technical Report 94–18, RPI Computer Science, 1994.

A The Complexity of $\mathcal{ALCHIQ}(\mathbf{D})$

The description logic $\mathcal{ALCHIQ}$ is decidable in NExpTime because it can be viewed as a fragment of $\mathcal{C}^2$ (first-order logic with counting quantifiers), and this logic is decidable in NExpTime even for binary coding of numbers [21]. It is widely believed that $\mathcal{ALCHIQ}(\mathbf{D})$ is also decidable in NExpTime; unfortunately, no result proving this has been published. Therefore, in this appendix we outline how to adapt the decision procedure from [21] to handle the restricted form of a concrete domain available in $\mathcal{ALCHIQ}(\mathbf{D})$.

Given a $\mathcal{C}^2$ formula φ obtained by translating an $\mathcal{ALCHIQ}$ knowledge base into first-order logic, the procedure from [21] represents (possibly infinite) first-order models using *1-types* and *2-types*—maximally consistent sets of unary and binary predicates, respectively (often called Hintikka sets). The concrete predicates in our case are all unary, so we can extend the notion of 1-types to include maximally consistent sets of positive concrete domain predicates. (We can consider only positive predicates because the set of the concrete predicates is closed under complement.) In fact, since $\mathcal{ALCHIQ}(\mathbf{D})$ is a two-sorted logic, we shall have two kinds of 1-types: one containing only ordinary unary predicates and one containing only unary concrete predicates. Since the number of concrete predicates occurring in φ is linear in the size of φ, we can clearly have at most exponentially many concrete 1-types. We eliminate all inconsistent sets by checking the consistency of the conjunction of all the member predicates. Since each such conjunction is linear in the size of φ, by the assumption (*iii*) of Definition 3, each check can be performed in exponential time. Hence, we can generate all 1-types of the concrete sort using an exponential preprocessing step, after which we can apply the algorithm from [21] as usual. Since the latter algorithm runs in NExpTime, our entire algorithm runs in NExpTime as well.

In some versions of $\mathcal{ALCHIQ}(\mathbf{D})$, the concrete domain can contain concrete predicates of arbitrary arity. It is currently unclear whether such a logic is decidable in NExpTime.