A Semantic Framework for Language Layering in WSML*

Jos de Bruijn and Stijn Heymans

DERI, University of Innsbruck, Technikerstraße 21a, 6020, Innsbruck, Austria
{jos.debruijn,stijn.heymans}@deri.org

Abstract. WSML presents a framework encompassing different language variants, rooted in Description Logics and (F-)Logic Programming. So far, the precise relationships between these variants have not been investigated. We take the nonmonotonic *first-order autoepistemic logic*, which generalizes both Description Logics and Logic Programming, and extend it with frames and concrete domains, to capture all features of WSML; we call this novel formalism FF-AEL. We consider two forms of language layering for WSML, namely *loose* and *strict* layering, where the latter enforces additional restrictions on the use of certain language constructs in the rule-based language variants, in order to give additional guarantees about the layering. Finally, we demonstrate that each WSML variant semantically corresponds to its target formalism, i.e. WSML-DL corresponds to $\mathcal{SHIQ}(\mathbf{D})$, WSML-Rule to the Stable Model Semantics for Logic Programs (the Well-Founded Semantics can be seen as an approximation), and WSML-Core to $\mathcal{DHL}(\mathbf{D})$ (without nominals), a Horn subset of $\mathcal{SHIQ}(\mathbf{D})$.

1 Introduction

The Web Service Modeling Language WSML[1] [6] is a language for modeling ontologies and Web services. In this paper we are only concerned with WSML ontologies. Thus, when referring to WSML in the remainder, we mean WSML ontologies. References to the ongoing work on the semantics of the functional and behavioral description of Web services can be found in [5].

WSML encompasses a framework of variants based on Description Logics [1] and (F-)Logic Programming [8, 9, 13]. Each WSML variant has a *target formalism*: *WSML-Core* is based on an intersection of the Description Logic $\mathcal{SHIQ}(\mathbf{D})$ and Horn Logic (without equality), called $\mathcal{DHL}(\mathbf{D})$ [10]. *WSML-DL* captures the Description Logic $\mathcal{SHIQ}(\mathbf{D})$. *WSML-Flight* is based on the Datalog subset of F-Logic, extended with (locally) stratified negation, for which the Well-Founded and Stable Model Semantics correspond [8, 9]. *WSML-Rule* is based on F-Logic Programming, extended with negation under the Well-Founded Semantics [8]. *WSML-Full* extends both WSML-DL and WSML-Rule towards first-order logic with nonmonotonic extensions.

WSML has two alternative *layerings*: Core $\Rightarrow$ DL $\Rightarrow$ Full and Core $\Rightarrow$ Flight $\Rightarrow$ Rule $\Rightarrow$ Full. For both layerings, WSML-Core and WSML-Full mark the least and

* This work was partially supported by the European Commission under the projects Knowledge Web (IST-2004-507482), DIP (FP6-507483), and SUPER (FP6-026850).

[1] `http://www.wsmo.org/wsml/wsml-syntax`

most expressive variants, respectively. The original WSML specification [6] did not demonstrate any semantic properties of this layering, nor did it include a specification of the semantics for WSML-Full; this was considered an open research topic.

In this paper, we specify an abstract syntax for WSML logical expressions, and define the WSML variants as subsets of this syntax. In order to give a semantics to WSML-Full and to investigate the language layering features of WSML, we specify a novel semantic framework for all WSML variants, based on first-order autoepistemic logic (FO-AEL) [14, 3], extended with frames [13] and concrete domains [2]. Our approach to concrete domains is a generalization of the approaches typically followed in Description Logics [2] and Datalog [19]. We call this extended language FF-AEL. We define the semantics of each individual WSML variant through an embedding in FF-AEL. This embedding translates a given WSML description to FF-AEL, and, depending on the language variants, includes a number of sentences which axiomatize the semantics of certain WSML constructs. As an example, we show the difference in the treatment of the subclass (*subConceptOf*) construct in WSML-DL and WSML-Rule.

A subclass statement is of the form $A :: B$, where A, B are terms. In F-Logic, this statement has an *intentional* (only if) semantics: whenever $A :: B$ is true, then every instance of A must be an instance of B. In Description Logics, however, subclass statements (of the form $A \sqsubseteq B$) have an *extensional* (if and only if) semantics: $A \sqsubseteq B$ is true *if and only if* every instance of A is an instance of B. In order to guarantee the correspondence between WSML-DL and Description Logics, this extensional semantics needs to be axiomatized. However, such extensional semantics cannot be axiomatized in a typical rules language such as WSML-Rule, because it would require universal quantification in the body of a rule, which is beyond the expressiveness of a rules language. For example, the following entailment is valid in WSML-DL and WSML-Full ($x : A$ stands for "x is an instance of A"):

$\forall x(x : A \supset x : B) \models A :: B$,
whereas it is not valid in WSML-Rule.

This distinction between intentional and extensional treatment of language constructs leads us to the definition of two approaches to language layering in WSML. When considering *loose* layering, a variant L_2 is layered on a variant L_1 if, considering an arbitrary theory of L_1, every L_1-formula which is a consequence under L_1 semantics, is also a consequence under L_2 semantics. When considering *strict* layering, additionally every L_1-formula which is a consequence under L_2 semantics must be a consequence under L_1 semantics. Considering these notions of language layering in the context of OWL, we observe that OWL Lite and OWL DL are strictly layered, and that OWL DL and OWL Full are not strictly, but loosely layered (cf. [12]).

It turns out that when considering strict language layering in WSML, certain restrictions on the use of ontology modeling constructs (e.g. subclass statements $::$) must be enforced for the rule-based WSML variants.

In the remainder of the paper we first review the Description Logic $\mathcal{SHIQ}(\mathbf{D})$ in Section 2. We proceed with our definitions of F-Logic with concrete domains, F-Logic Programming, and FF-AEL, in Sections 3, 4, and 5. We then proceed to describe the abstract syntax for WSML variants, and define strict and loose language layering, in

Section 6. We demonstrate the correspondence between the variants and the intended target formalisms in Section 7. Finally, we conclude the paper in Section 8.

2 The Description Logic $\mathcal{SHIQ}(\mathrm{D})$

The signature $\Sigma = \langle \mathcal{C}, \mathcal{D}, \mathcal{R}_a, \mathcal{R}_c, \mathcal{F}_a, \mathcal{F}_c \rangle$ of a $\mathcal{SHIQ}(\mathbf{D})$ [1] language consists of pairwise disjoint sets of concept ($\mathcal{C}$), datatype ($\mathcal{D}$), abstract role ($\mathcal{R}_a$), concrete role ($\mathcal{R}_c$), individual ($\mathcal{F}_a$), and data value ($\mathcal{F}_c$) identifiers. $\mathcal{SHIQ}(\mathbf{D})$ descriptions are defined as follows, with A a concept identifier, D a datatype identifier, C, C' descriptions, R, R' role identifiers, S, S' abstract role identifiers, U, U' concrete role identifiers, a, b individual identifiers, o a data value identifier, and n a non-negative integer.

$$C, C' \longrightarrow \bot \mid A \mid C \sqcap C' \mid \neg C \mid \geqslant nS.C \mid \geqslant nU.D \mid \leqslant nS.C \mid \leqslant nU.D$$

A $\mathcal{SHIQ}(\mathbf{D})$ ontology is a set of axioms of the following forms.

$$C \sqsubseteq C' \mid S \sqsubseteq S' \mid U \sqsubseteq U' \mid S \equiv S'^{-} \mid \ (S)^{+} \mid C(a) \mid S(a,b) \mid U(a,o) \mid a = b \mid a \neq b$$

Additionally, we have that in *number restrictions* $\geqslant nS.C$ and $\leqslant nS.C$, S has to be *simple*, i.e., S and its sub-roles may not be transitive (with $(S)^{+}$ denoting transitivity).

For reasons of space, we do not present the $\mathcal{SHIQ}(\mathbf{D})$ semantics here, but refer to [1]. Given a $\mathcal{SHIQ}(\mathbf{D})$ axiom ϕ (resp., ontology Φ), we denote the FOL-equivalent of π (resp., Φ) with $\pi(\phi)$ (resp., $\pi(\Phi)$); by [1] we know that such equivalents exist.

$\mathcal{DHL}(\mathbf{D})$ ([10]) is a Horn subset of $\mathcal{SHIQ}(\mathbf{D})$, which means that every $\mathcal{DHL}(\mathbf{D})$ ontology is equivalent to a Horn theory. For the complete definition of $\mathcal{DHL}(\mathbf{D})$, see [5].

3 Frame Logic with Concrete Domains

In this section we review F-Logic, following [4], and define a novel extension with concrete domains, which is similar to, but more general than, the concrete domains extensions usually considered in Description Logics [2] and Datalog [19].

A language $\mathcal{L}$ has a signature of the form $\Sigma_{\mathcal{L}} = \langle \mathcal{F}, \mathcal{P}, \mathcal{F}^D, \mathcal{P}^D \rangle$, with $\mathcal{F}$ and $\mathcal{P}$ sets of function- and predicate-symbols, and $\mathcal{F}^D$ and $\mathcal{P}^D$ sets of concrete function and predicate symbols, each with an associated arity n, which is a nonnegative integer; $\mathcal{F}$ and $\mathcal{F}^D$ (resp., $\mathcal{P}$ and $\mathcal{P}^D$) are pairwise disjoint. Notice that the symbols in $\mathcal{F}$ and $\mathcal{P}$ do not have associated arities.

Let $\mathcal{V}$ be a set of variable symbols, disjoint from all sets of symbols in $\Sigma_{\mathcal{L}}$. Abstract terms are constructed using symbols from $\mathcal{F}$ and $\mathcal{V}$ as usual. Concrete terms are constructed using symbols from $\mathcal{F}^D$ and $\mathcal{V}$. Terms are either abstract or concrete terms. Abstract atomic formulas (atoms) are $\top, \bot$ or are constructed from terms and symbols in $\mathcal{P}$ in the usual way. Concrete atoms are constructed from concrete terms and symbols in $\mathcal{P}^D$. Atoms are either abstract or concrete atoms. Molecules are *isa* molecules of the form $t_1 : t_2$, *subclass* molecules of the form $t_1 :: t_2$, or *attribute value* molecules of the form $t_1[t_2 \twoheadrightarrow t_3]$, with t_1, t_2, t_3 terms.

Formulas are constructed in the usual way from atoms and molecules using the symbols $\neg, \wedge, \vee, \supset, \equiv, \forall, \exists,), ($, with the difference that abstract quantifiers are indexed with ${}_a$ $(\exists_a, \forall_a)$ and concrete quantifiers are indexed with ${}_c$ $(\exists_c, \forall_c)$. Finally, variables quantified using an abstract quantifier $(\exists_a, \forall_a)$ may not occur in a concrete term or atom.

An interpretation is a tuple $\mathbf{I} = \langle U, U^D, \prec_U, \in_U, \mathbf{I}_F, \mathbf{I}_P, \mathbf{I}_{\twoheadrightarrow} \rangle$. U and U^D are disjoint non-empty countable sets, called the *abstract* and *concrete* domains, $\prec_U$ is an irreflexive partial order over $U \cup U^D$, and $\in_U$ is a binary relation over $U \cup U^D$. We write $a \preceq_U b$ when $a \prec_U b$ or $a = b$, for $a, b \in U \cup U^D$. For each interpretation holds that if $a \in_U b$ and $b \preceq_U c$ then $a \in_U c$. Thus, if $b \preceq_U c$, then $\{k \mid k \in_U b, k \in U \cup U^D\} \subseteq \{k \mid k \in_U c, k \in U \cup U^D\}$. We call the set $\{k \mid k \in_U b, k \in U \cup U^D\}$ the *class extension* of b. Thus, if $b \preceq_U c$, then the class extension of b is a subset of the class extension of c. However, the converse of this statement is not universally true.

An abstract function symbol $f \in \mathcal{F}$ is interpreted as a function over the domain U: $\mathbf{I}_F(f) : U^i \to U$, for every $i \geq 0$. An n-ary concrete function symbol $f \in \mathcal{F}^D$ is interpreted as a function over the domain U^D: $\mathbf{I}_F(f) : (U^D)^n \to U^D$. An abstract predicate symbol $p \in P$ is interpreted as a relation over the domain $U \cup U^D$: $\mathbf{I}_P(p) \subseteq (U \cup U^D)^i$, for every $i \geq 0$. An n-ary concrete predicate symbol $p \in P^D$ is interpreted as a relation over the domain U^D: $\mathbf{I}_P(p) \subseteq (U^D)^n$. $\mathbf{I}_{\twoheadrightarrow}$ associates a binary relation over $U \cup U^D$ with each $u \in U \cup U^D$: $\mathbf{I}_{\twoheadrightarrow}(u) \subseteq (U \cup U^D) \times (U \cup U^D)$.

A *concrete domain scheme* $\mathfrak{S}$ is a tuple $\mathfrak{S} = \langle U^{\mathfrak{S}}, \mathcal{F}^{\mathfrak{S}}, \mathcal{P}^{\mathfrak{S}}, \cdot^{\mathfrak{S}} \rangle$, where $U^{\mathfrak{S}}$ is a non-empty countable set of concrete values, $\mathcal{F}^{\mathfrak{S}}$ and $\mathcal{P}^{\mathfrak{S}}$ are disjoint sets of concrete function and predicate symbols, each with an associated nonnegative arity n, and $\cdot^{\mathfrak{S}}$ is an interpretation function which assigns a function $f^{\mathfrak{S}} : (U^{\mathfrak{S}})^n \to U^{\mathfrak{S}}$ to every $f \in \mathcal{F}^{\mathfrak{S}}$ and a relation $p^{\mathfrak{S}} \subseteq (U^{\mathfrak{S}})^n$ to every $p \in \mathcal{P}^{\mathfrak{S}}$. A language $\mathcal{L}$ with signature $\Sigma_{\mathcal{L}} = \langle \mathcal{F}, \mathcal{P}, \mathcal{F}^D, \mathcal{P}^D \rangle$ *conforms to* a concrete domains scheme $\mathfrak{S} = \langle U^{\mathfrak{S}}, \mathcal{F}^{\mathfrak{S}}, \mathcal{P}^{\mathfrak{S}}, \cdot^{\mathfrak{S}} \rangle$ if $\mathcal{F}^D = \mathcal{F}^{\mathfrak{S}}$ and $\mathcal{P}^D = \mathcal{P}^{\mathfrak{S}}$. An interpretation $\mathbf{I} = \langle U, U^D, \prec_U, \in_U, \mathbf{I}_F, \mathbf{I}_P, \mathbf{I}_{\twoheadrightarrow} \rangle$ of $\mathcal{L}$ *conforms to* $\mathfrak{S}$ if $U^D = U^{\mathfrak{S}}$, and $\mathbf{I}_F(f) = f^{\mathfrak{S}}, \mathbf{I}_P(p) = p^{\mathfrak{S}}$ for every $f \in \mathcal{F}^{\mathfrak{S}}, p \in \mathcal{P}^{\mathfrak{S}}$, respectively. In the remainder we assume that every language conforms to the concrete domain scheme under consideration. We illustrate the concept through the definition of a concrete domain scheme for integers and strings.

Example 1. We define the concrete domain scheme $\mathfrak{S} = \langle U^{\mathfrak{S}}, \mathcal{F}^{\mathfrak{S}}, \mathcal{P}^{\mathfrak{S}}, \cdot^{\mathfrak{S}} \rangle$ as follows: $U^{\mathfrak{S}}$ is the union of the sets of integer numbers and finite-length sequences of Unicode characters. $\mathcal{F}^{\mathfrak{S}}$ is the union of the set of finite-length sequences of decimal digits, optionally with a leading minus (-), and the set of finite-length sequences of Unicode characters, delimited with `"` (for simplicity, we assume that the character `"` does not occur in such strings), all with arity 0. $\mathcal{P}^{\mathfrak{S}}$ consists of unary predicate symbols $integer$ and $string$, and the binary predicate symbol $numeric\text{-}equals$. The interpretation function $\cdot^{\mathfrak{S}}$ interprets (signed) sequences of decimal digits and `"`-delimited sequences of characters as integers and strings, respectively, in the natural way; $\cdot^{\mathfrak{S}}$ interprets $integer$ and $string$ as the set of integers and strings; finally, $\cdot^{\mathfrak{S}}$ interprets $numeric\text{-}equals$ as identity over the set of integers.

Our approach to integrating concrete domains is a generalization of the usual approaches to integrating concrete domains in Description Logics [2], as well as extensions such as [17], and Datalog [19] (where they are called *built-ins*). In DLs, all predicate symbols are sorted (using the sorts *abstract* and *concrete*; binary predicates with the sort $abstract \times concrete$ are usually called *features*) and certain restrictions apply on the concrete domain schemes in order to guarantee decidability of reasoning and the existence of effective algorithms. In Datalog concrete predicates are only allowed to occur in rule bodies, and variables must occur in abstract atoms in the body of the rule; this guarantees the existence of effective terminating reasoning methods.

A variable assignment B assigns each variable $x \in \mathcal{V}$ to an individual $x^B \in U \cup U^D$. A variable assignment B' is an abstract (resp., concrete) x-variant of B if $x^{B'} \in U \cup U^D$ (resp., $x^{B'} \in U^D$) and $y^{B'} = y^B$ for $y \neq x$. The interpretation of a term t in some $\mathbf{I}$ with respect to some variable assignment B, written $t^{\mathbf{I},B}$, is defined as: $t^{\mathbf{I},B} = t^B$ if $t \in \mathcal{V}$, and $t^{\mathbf{I},B} = \mathbf{I}_F(f)(t_1^{\mathbf{I},B}, \ldots, t_n^{\mathbf{I},B})$ if t is of the form $f(t_1, \ldots, t_n)$. A variable substitution β, usually written in postfix notation, is a partial mapping from variable symbols to ground terms. A variable substitution β is *associated with* (cf. [3]) a variable assignment B if for every variable symbol x such that $x^B = k$ and there exists a ground term t such that $t^{\mathbf{I},B} = k$, then $x\beta = t'$ for some ground term t' such that $t'^{\mathbf{I},B} = k$; otherwise $x\beta$ is not defined.

Satisfaction of atomic formulas and molecules ϕ in $\mathbf{I}$, given the variable assignment B, denoted $(\mathbf{I}, B) \models_{\mathsf{f}} \phi$, is defined as: $(\mathbf{I}, B) \models_{\mathsf{f}} \top$, $(\mathbf{I}, B) \not\models_{\mathsf{f}} \bot$, $(\mathbf{I}, B) \models_{\mathsf{f}} p(t_1, \ldots, t_n)$ iff $(t_1^{\mathbf{I},B}, \ldots, t_n^{\mathbf{I},B}) \in \mathbf{I}_P(p)$, $(\mathbf{I}, B) \models_{\mathsf{f}} t_1 : t_2$ iff $t_1^{\mathbf{I},B} \in_U t_2^{\mathbf{I},B}$, $(\mathbf{I}, B) \models_{\mathsf{f}} t_1 :: t_2$ iff $t_1^{\mathbf{I},B} \preceq_U t_2^{\mathbf{I},B}$, $(\mathbf{I}, B) \models_{\mathsf{f}} t_1[t_2 \twoheadrightarrow t_3]$ iff $\langle t_1^{\mathbf{I},B}, t_3^{\mathbf{I},B} \rangle \in \mathbf{I}_{\twoheadrightarrow}(t_2^{\mathbf{I},B})$, and $(\mathbf{I}, B) \models_{\mathsf{f}} t_1 = t_2$ iff $t_1^{\mathbf{I},B} = t_2^{\mathbf{I},B}$.

This extends to arbitrary formulas as follows: $(\mathbf{I}, B) \models_{\mathsf{f}} \phi_1 \wedge \phi_2$ (resp. $(\mathbf{I}, B) \models_{\mathsf{f}} \phi_1 \vee \phi_2$, $(\mathbf{I}, B) \models_{\mathsf{f}} \neg\phi_1$) iff $(\mathbf{I}, B) \models_{\mathsf{f}} \phi_1$ and $(\mathbf{I}, B) \models_{\mathsf{f}} \phi_2$ (resp. $(\mathbf{I}, B) \models_{\mathsf{f}} \phi_1$ or $(\mathbf{I}, B) \models_{\mathsf{f}} \phi_2$, $(\mathbf{I}, B) \not\models \phi_1$); $(\mathbf{I}, B) \models_{\mathsf{f}} \forall_a x(\phi_1)$ (resp. $(\mathbf{I}, B) \models_{\mathsf{f}} \exists_a x(\phi_1)$) iff for every (resp. for some) B'_a which is an abstract x-variant of B, $(\mathbf{I}, B'_a) \models_{\mathsf{f}} \phi_1$; $(\mathbf{I}, B) \models_{\mathsf{f}} \forall_c x(\phi_1)$ (resp. $(\mathbf{I}, B) \models_{\mathsf{f}} \exists_c x\ (\phi_1)$) iff for every (resp. for some) B'_c which is a concrete x-variant of B, $(\mathbf{I}, B'_c) \models_{\mathsf{f}} \phi_1$. If a variable x is quantified using a concrete quantifier ($\forall_c, \exists_c$), x is a *concrete variable*; otherwise, x is an *abstract variable*.

Given a concrete domain scheme $\mathfrak{S}$, an interpretation $\mathbf{I}$ is a *model* of a formula ϕ if $\mathbf{I}$ conforms to $\mathfrak{S}$ and for every variable assignment B, $(\mathbf{I}, B) \models_{\mathsf{f}} \phi$. A formula ϕ is *satisfiable* if it has a model; ϕ is *valid* if every interpretation which conforms to $\mathfrak{S}$ is a model of ϕ. These notions extend to theories $\Phi \subseteq \mathcal{L}$ in the natural way. A theory $\Phi \subseteq \mathcal{L}$ *entails* a formula $\phi \in \mathcal{L}$ if every model of Φ is also a model of ϕ.

Contextual FOL is F-Logic without molecules. *Classical FOL* is contextual first-order logic in which each function symbol and predicate symbol has one associated arity n, which is a nonnegative integer. We denote satisfaction and entailment in classical FOL with the symbol $\models$.

The following correspondence between $\mathcal{SHIQ}(\mathbf{D})$ and F-Logic ontologies is a straightforward extension of a result in [4]. Given an FOL formula (resp., theory) ϕ (resp., Φ), then $\delta(\phi)$ (resp., $\delta(\Phi)$) is the F-Logic formula (resp., theory) obtained from ϕ (resp., Φ) by replacing all atoms of the forms $A(t)$ and $R(t_1, t_2)$, where t, t_1, t_2 are terms, with molecules of the forms $t : A$ and $t_1[R \twoheadrightarrow t_2]$, respectively.

Proposition 1. *Given a concrete domain scheme $\mathfrak{S}$, let Φ, ϕ be a $\mathcal{SHIQ}(\mathbf{D})$ theory and formula, respectively. Then, $\Phi \models \phi$ iff $\delta(\pi(\Phi)) \models_{\mathsf{f}} \delta(\pi(\phi))$.*

4 F-Logic Programs

Given a concrete domain scheme $\mathfrak{S}$ and a language $\mathcal{L}$ with at least one 0-ary function symbol, a rule is of the form

$$h \leftarrow b_1, \ldots, b_m, \mathit{not}\, c_1, \ldots, \mathit{not}\, c_n, \tag{1}$$

where $h, b_1, \ldots, b_m, c_1, \ldots, c_n$ are (equality-free) atoms or molecules, and h is not a concrete atom. h is the *head atom* of r, $B^+(r) = \{b_1, \ldots, b_m\}$ is the *positive body* of r, and $B^-(r) = \{c_1, \ldots, c_n\}$ is the *negative body* of r. If $B^-(r) = \emptyset$, then r is *positive*. If every variable in r occurs in an abstract atom in $B^+(r)$, then r is *safe*. If a variable occurs in a concrete atom, it is a concrete variable; otherwise, it is an abstract variable. The following rules axiomatize the semantics of subclass molecules: $(*)\ x :: z \ \leftarrow\ x :: y, y :: z$, $(**)\ x : z \ \leftarrow\ x : y, y :: z$, and $(* * *)\ x :: x$, where (*) axiomatizes transitivity of the subclass relation; $(**)$ axiomatizes inheritance of class membership; and $(* * *)$ axiomatizes the fact that every class is a subclass of itself[2]. A *normal F-Logic program* P is a set of rules of the form (1) which includes the rules $(*, **, * * *)$. If every rule $r \in P$ is positive (resp., safe), then P is positive (resp., safe).

The *Herbrand base* of $\mathcal{L}$ is the set of ground atomic formulas and molecules of $\mathcal{L}$. Subsets of the Herbrand base are called *Herbrand interpretations*.

The *grounding* of a logic program P, denoted $gr(P)$, is the union of all possible ground instantiations of P, obtained by replacing each abstract (resp., concrete) variable in a rule r with a ground (resp., ground concrete) term of $\mathcal{L}$, for each rule $r \in P$.

Let P be a positive program. A Herbrand interpretation M of P is a *model* of P if M conforms to $\mathfrak{S}$, $\top \in M, \bot \notin M$, and, for every rule $r \in gr(P)$, $B^+(r) \subseteq M$ implies $H(r) \cap M \neq \emptyset$. A Herbrand model M is *minimal* iff for every model M' such that $M' \subseteq M$, $M' = M$.

Following [9], the *reduct* of a logic program P with respect to an interpretation M, denoted P^M, is obtained from $gr(P)$ by deleting (i) each rule r with $B^-(r) \cap M \neq \emptyset$, and (ii) $\mathit{not}\ c$ from the body of every remaining rule r with $c \in B^-(r)$. If M is a minimal Herbrand model of P^M, then M is a *stable model* of P.

If P is a positive logic program, then the corresponding Horn F-Logic theory Φ is obtained by replacing the arrow $\leftarrow$ and comma (,) in every rule with the symbols $\supset$ and $\wedge$ in the usual way, and prefixing the formula with a concrete (resp., abstract) universal quantifier ($\forall_c$ or $\forall_a$, resp.) for every concrete (resp., abstract) variable x. The following proposition follows straightforwardly from the definition, and the classical results by Herbrand.

Proposition 2. *Given a concrete domain scheme $\mathfrak{S}$, let P be a positive logic program and Φ be the corresponding Horn F-Logic theory, then*
– P has a stable model iff Φ is satisfiable, and
– if P has a stable model M, it is unique, and for every ground atom or molecule α, $\alpha \in M$ iff $\Phi \models_{\mathsf{f}} \alpha$.

[2] Note that the rule (4) is not safe. However, $(* * *)$ is not necessary in case subclass statements (::) do not occur in rule bodies and are not considered when determining consequences.

5 First-Order Autoepistemic Logic with Frames and Concrete Domains

First-Order Autoepistemic Logic (FO-AEL) [14, 3] is an extension of first-order logic with a modal belief operator L, which is interpreted nonmonotonically. We specify an extension of FO-AEL, based on F-Logic with concrete domains, called FF-AEL.

An FF-AEL language $\mathcal{L}_{\mathsf{L}}$ is defined relative to a language $\mathcal{L}$:
– any atomic formula or molecule in $\mathcal{L}$ is a formula in $\mathcal{L}_{\mathsf{L}}$,
– if ϕ is a *formula* in $\mathcal{L}_{\mathsf{L}}$, then $\mathsf{L}\phi$, called a *modal atom*, is a formula in $\mathcal{L}_{\mathsf{L}}$, and
– *complex formulas* are constructed as in F-Logic with concrete domains.
A formula without modal atoms is an *objective* formula.

An *autoepistemic interpretation* is a pair $\langle \mathbf{I}, \Gamma \rangle$, where $\mathbf{I} = \langle U, U^D, \prec_U, \in_U, \mathbf{I}_F, \mathbf{I}_P, \mathbf{I}_{\twoheadrightarrow} \rangle$ is an interpretation, and $\Gamma \subseteq \mathcal{L}_{\mathsf{L}}$ is a set of sentences, called the *belief set*. Satisfaction of objective atomic formulas in $\langle \mathbf{I}, \Gamma \rangle$ corresponds to satisfaction in $\mathbf{I}$.

Satisfaction of a formula $\mathsf{L}\phi$ ($\phi \in \mathcal{L}_{\mathsf{L}}$) in an interpretation $\langle \mathbf{I}, \Gamma \rangle$ with respect to a variable assignment B under the *any-name semantics*[3], denoted $(\mathbf{I}, B) \models_\Gamma \mathsf{L}\phi$, is defined as follows:

> $(\mathbf{I}, B) \models_\Gamma \mathsf{L}\phi$ iff, for some variable substitution(s) β, associated with B, $\phi\beta$ has no free variables and $\phi\beta \in \Gamma$.

This extends to arbitrary formulas in the usual way (see also Section 3).

$\langle \mathbf{I}, \Gamma \rangle$ is a *model* of ϕ, denoted $\mathbf{I} \models_\Gamma \phi$, if $(\mathbf{I}, B) \models_\Gamma \phi$ for every variable assignment B. This extends to sets of formulas in the usual way. A set of formulas $A \subseteq \mathcal{L}_{\mathsf{L}}$ *entails* a sentence ϕ with respect to a belief set Γ, denoted $A \models_\Gamma \phi$, if for every interpretation $\mathbf{I}$ such that $\mathbf{I} \models_\Gamma A$, $\mathbf{I} \models_\Gamma \phi$.

A central notion in FF-AEL is the *stable expansion*, which is the set of beliefs of an ideally introspective agent, given some base set. A belief set $T \subseteq \mathcal{L}_{\mathsf{L}}$ is a *stable expansion* of a base set $A \subseteq \mathcal{L}_{\mathsf{L}}$ iff $T = \{\phi \mid A \models_T \phi\}$.

A formula ϕ is an *autoepistemic consequence* of A if ϕ is included in every stable expansion of A. In the remainder, when referring to consequences of a theory A we mean *objective autoepistemic consequences*, unless specified otherwise. The following proposition is a straightforward generalization of a result in [14].

Proposition 3. *Given a concrete domain scheme $\mathfrak{S}$, let $\Phi \subseteq \mathcal{L}$ be a satisfiable F-Logic theory. Then, Φ has one consistent stable expansion T, and $T \cap \mathcal{L} = \{\phi \mid \Phi \models_{\mathsf{f}} \phi\}$.*

Embedding Logic Programs Following [3], we define an embedding as a function which takes a normal F-Logic program P as its argument and returns a set of FF-AEL sentences. Since the unique-names assumption does not hold in FF-AEL, it is necessary to axiomatize default uniqueness of names. With UNA_Σ we denote the set of axioms

$$\neg\mathsf{L}(t_1 = t_2) \supset t_1 \neq t_2, \quad \text{for all pairs of distinct ground terms } t_1, t_2.$$

[3] [14] presents also the all-names semantics, but we follow [14, 3] in their choice for the any-name semantics.

Let r be a normal rule of the form (1). Then,

$$\tau_{HP}(r) = (\forall) \bigwedge\nolimits_{1 \le i \le m} b_i \wedge \bigwedge\nolimits_{1 \le j \le n} \neg \mathsf{L} c_j \supset h,$$

such that each concrete variable is quantified using $\forall_c$, and all other variables are quantified using $\forall_a$. For a normal F-Logic program P, we define:

$$\tau_{HP}(P) = \{\tau_{HP}(r) \mid r \in P\} \cup UNA_{\Sigma_P}.$$

Recall the three rules $(*)$, $(**)$ and $(* * *)$, which are part of every F-Logic program. These rules translate to FF-AEL as follows: $(*)$ $\forall_a x, y, z\ (x :: y \wedge y :: z \supset x :: z)$, $(**)$ $\forall_a x, y, z\ (x : y \wedge y :: z \supset x : z)$ and $(* * *)$ $\forall_a x\ (x :: x)$. It can be easily verified, using the definition of interpretations and satisfaction in F-Logic, that the embeddings of these formulas are all valid in F-Logic and thus in FF-AEL (i.e. they are included in every stable expansion). Faithfulness of the embedding is established in the following proposition, which generalizes a result in [3].

Proposition 4. *Given a concrete domain scheme $\mathfrak{S}$, a Herbrand interpretation M of a normal F-Logic program P is a stable model of P iff there is a consistent stable expansion T of $\tau_{HP}(P)$ such that M coincides with the set of objective ground atoms and molecules in T.*

6 WSML Logical Expressions

In this section we present an abstract syntax for WSML logical expressions, and define their semantics through an embedding in FF-AEL. We use this abstract syntax to discuss two forms of language layering between the WSML variants. Note that this abstract syntax for WSML formulas differs from the more verbose (logical expression) surface syntax in the original specification. There is, however, a straightforward mapping from the syntax we use here to the surface syntax; see [5, Section 4.3].

Given a concrete domain scheme $\mathfrak{S}$[4], the signature of a WSML language $\mathcal{L}$ is of the form $\Sigma = \langle \mathcal{F}, \mathcal{P}, \mathcal{F}^{\mathfrak{S}}, \mathcal{P}^{\mathfrak{S}} \rangle$, as in F-Logic with concrete domains (cf. Section 3).

Terms and atoms are defined as in Section 3. Molecules are defined analogously to F-Logic: if t_1, t_2, t_3 are terms, then $t_1 : t_2$, $t_1 :: t_2$ and $t_1[t_2 \,\mathsf{x}\, t_3]$, with $\mathsf{x} \in \{\mathsf{ot}, \mathsf{it}, \mathsf{hv}\}$, are molecules. The symbol ot stands for the WSML construct *ofType*; a statement $t_1[t_2 \,\mathsf{ot}\, t_3]$ requires all values for the attribute t_2 to be *known* to be of a member of the type t_3; it stands for the WSML construct *impliesType*; a statement $t_1[t_2 \,\mathsf{it}\, t_3]$ implies that all values for the attribute t_2 are a member of the class t_3; hv stands for the WSML construct *hasValue*; a molecule $t_1[t_2 \,\mathsf{hv}\, t_3]$ says that the individual t_1 has an attribute t_2 with value t_3.

WSML formulas are inductively defined as follows, with $\phi, \psi \in \mathcal{L}$:
– atoms and molecules are formulas;
– $\sim \phi$, with $\sim \in \{\neg, \mathit{not}\}$, is a formula;
– $\phi \star \psi$, with $\star \in \{\wedge, \vee, \supset, \equiv\}$, is a formula; and

[4] It is assumed in WSML that such a concrete domain scheme incorporates at least the XML Schema datatypes *string*, *integer*, and *decimal* [6, Appendix C].

– $Q\ x(\phi)$, with $Q \in \{\forall_a, \exists_a, \forall_c, \exists_c\}$ and $x \in \mathcal{V}$, is a formula.
Additionally, no variable quantified using an abstract quantifier ($\forall_a, \exists_a$) may be used in a concrete atom. We assume that the predicate symbols $\_it$, $\_ot$ are not used in any WSML formula. As usual, WSML sentences are WSML formulas with no free variables.

The semantics of WSML formulas is defined through a translation to FF-AEL: let Φ be a set of WSML formulas, then $tr(\Phi)$ is the FF-AEL theory obtained as follows: for each $\phi \in \Phi$, $tr(\phi)$ is obtained from ϕ by replacing each occurrence of not with $\neg\mathsf{L}$, replacing hv with $\twoheadrightarrow$, and replacing molecules of the forms $t_1[t_2\ \mathsf{ot}\ t_3]$ and $t_1[t_2\ \mathsf{it}\ t_3]$, with t_1, t_2 and t_3 terms, with atoms of the forms $\_ot(t_1, t_2, t_3)$ and $\_it(t_1, t_2, t_3)$, respectively. Finally, the following formulas are used to axiomatize the *intentional* (only if) semantics of the ot and it molecules:

$$\forall_a\ x, y, z, v, w\ (\_ot(x, y, z) \wedge v : x \wedge v[y \twoheadrightarrow w] \wedge \neg\mathsf{L}w : z \supset \bot); \tag{2}$$
$$\forall_a\ x, y, z, v, w\ (\_it(x, y, z) \wedge v : x \wedge v[y \twoheadrightarrow w] \supset w : z), \tag{3}$$

and the following formulas are used to axiomatize the *extensional* (if and only if) semantics of the it and :: molecules, necessary for DL-like languages:

$$\forall_a\ x, y, z\ (\forall_a v, w(v : x \wedge v[y \twoheadrightarrow w] \supset w : z)) \supset \_it(x, y, z), \text{ and} \tag{4}$$
$$\forall_a\ \ x, y\ \ (\forall_a v(v : x \supset v : y)) \supset x :: y. \tag{5}$$

WSML-Full and -FOL Any WSML sentence is a WSML-Full sentence. A *WSML-Full theory* is a set of WSML-Full sentences. The semantics of a WSML-Full theory Φ is given through the embedding $tr_{Full}(\Phi) = tr(\Phi) \cup \{(2), (3), (4), (5)\}$.

A WSML-FOL sentence is a WSML-Full sentence which neither contains ot-molecules, nor occurrences of the default negation operator not . A *WSML-FOL theory* is a set of WSML-FOL sentences. The semantics of a WSML-FOL theory Φ is given through the embedding $tr_{FOL}(\Phi) = tr(\Phi) \cup \{(3), (4), (5)\}$.

WSML-Rule WSML-Rule formulas are of the form

$$(\forall) b_1 \wedge \ldots \wedge b_l \wedge not\ c_1 \wedge \ldots \wedge not\ c_m \supset h \tag{6}$$

where $b_1, \ldots, b_l, c_1, \ldots, c_m$ are atoms or hv, ot, or *isa* (:) molecules, with l, m non-negative integers, and h an abstract equality-free atom or molecule; if $h = \bot$, then we call the rule an *integrity constraint*. Additionally, each quantifier is either abstract ($\forall_a$) or concrete ($\forall_c$). A *WSML-Rule theory* is a set of WSML-Rule sentences. The semantics of a WSML-Rule theory Φ is given through the embedding $tr_{Rule}(\Phi) = tr(\Phi) \cup \{(2), (3)\}$.

Concrete atoms in WSML-Rule correspond to the common built-in atoms in Logic Programming.

Notice that there is a natural correspondence between WSML-Rule formulas of the form (6) and rules in a logic program of the form (1). Thus, WSML-Rule formulas are essentially rules with a head and a body. Notice that, whereas the embedding $tr_{Full}(\Phi)$ includes the sentences (4) and (5), the embedding $tr_{Rule}(\Phi)$ does not, because there is no natural correspondence to rules due to the universal quantification in the antecedent of the formulas (4) and (5). The it- and $::$-molecules may not be used in the body of a rule in order to maintain a strict correspondence between the WSML-Rule semantics and the WSML-Full semantics, as illustrated in the following example.

Example 2. Consider the theory $\varPhi$ consisting of the formulas

$$\forall_a x(x : A \supset x : B) \text{ and}$$
$$A :: B \supset q.$$

The theory $tr_{Rule}(\varPhi)$ neither has $A :: B$ nor q among its consequences. In contrast, it is easy to verify that, by (5), $tr_{Full}(\varPhi)$ has both $A :: B$ and q among its consequences.

WSML-Flight A *WSML-Flight theory* is a WSML-Rule theory for which holds that, for every formula of the form (6), every variable occurs in a positive abstract body atom b_i, no function symbol in (6) is used with an arity higher than 0, and the theory is *locally stratified*[5].[6] The semantics of a WSML-Flight theory $\varPhi$ is given through the embedding $tr_{Flight}(\varPhi) = tr_{Rule}(\varPhi)$.

WSML-DL Given an FOL formula ϕ, $\delta'(\phi)$ is obtained from ϕ by replacing atoms of the forms $A(t_1), R(t_1, t_2)$, with t_1, t_2 terms, with molecules of the forms $t_1 : A, t_1[R \,\mathsf{hv}\, t_2]$.

Given a $\mathcal{SHIQ}(\mathbf{D})$ signature $\varSigma = \langle \mathcal{C}, \mathcal{D}, \mathcal{R}_a, \mathcal{R}_c, \mathcal{F}_a, \mathcal{F}_c \rangle$, the corresponding WSML signature is $\langle \mathcal{C} \cup \mathcal{D} \cup \mathcal{R}_a \cup \mathcal{R}_c \cup \mathcal{F}_a, \emptyset, \mathcal{F}'_c, \mathcal{D}' \rangle$, where $\mathcal{F}'_c$ is the set of 0-ary functions symbols obtained from $\mathcal{F}_c$ and $\mathcal{D}'$ is the set of 1-ary predicate symbols obtained from $\mathcal{D}$. A WSML-DL formula is a WSML formula of the form

- $\delta'(\phi)$, where ϕ is the FOL equivalent of a $\mathcal{SHIQ}(\mathbf{D})$ axiom of the signature $\varSigma$,
- $a :: b$, with $a, b \in \mathcal{C}$,
- $a[s \,\mathsf{it}\, b]$, with $s \in \mathcal{R}_a$ and $a, b \in \mathcal{C}$, or
- $a[u \,\mathsf{it}\, d]$, with $u \in \mathcal{R}_c$, $a \in \mathcal{C}$ and $d \in \mathcal{D}$.

Given a $\mathcal{SHIQ}(\mathbf{D})$ signature $\varSigma$, a *WSML-DL theory* is a set of WSML-DL sentences. The semantics of a WSML-DL theory $\varPhi$ is given through the embedding $tr_{DL}(\varPhi) = tr_{FOL}(\varPhi)$.

WSML-Core A WSML-DL formula which is also a Flight formula is a WSML-Core formula. *WSML-Core theory* is a set of WSML-Core sentences. The semantics of a WSML-Core theory $\varPhi$ is given through the embedding $tr_{Core}(\varPhi) = tr_{Flight}(\varPhi) = tr_{Rule}(\varPhi)$.

Let $\mathfrak{S}$ be a concrete domain scheme and $x \in \{Core, Flight, Rule, DL, FOL, Full\}$ a WSML variant. We say that a WSML-x theory $\varPhi$ is *consistent* if $tr_x(\varPhi)$ has a consistent stable expansion an WSML-x formula ϕ, and is a WSML-x consequence of $\varPhi$ if $\phi \in T$ for every stable expansion T of $tr_x(\varPhi)$.

[5] Each atom or molecule in $gr(\varPhi)$ is assigned a stratum, which is an integer. We say that $gr(\varPhi)$ is stratified if there is an assignment of atoms and molecules to strata such that: if an atom or molecule p occurs positively in a rule with an atom or molecule q as its head, then p has the same or a lower stratum, and if p occurs negatively in a rule with q as its head, then p has a lower stratum than q. If $gr(\varPhi)$ is stratified, then $\varPhi$ is locally stratified.

[6] These conditions correspond to the usual safety condition which must hold for Datalog programs, and the usual local stratification for logic programs.

WSML Language Layering We now turn to the relationships between the language variants. Certain relationships are straightforward, because of equivalence of the embeddings in FF-AEL (e.g. given a WSML-Core theory Φ, $tr_{Core}(\Phi) = tr_{Flight}(\Phi) = tr_{Rule}(\Phi)$); however, there are also certain differences between embeddings (e.g. given a WSML-Core theory Φ, $tr_{Core}(\Phi) \neq tr_{DL}(\Phi) \neq tr_{Full}(\Phi)$). We consider two forms of language layering: *strict* and *loose* language layering.

Admissible consequences under strict/loose language layering are subsets of all formulas of a given WSML variant. Intuitively, admissible consequences are the formulas allowed to be considered when checking consequences of a given a theory.

The admissible consequences under strict language layering for WSML are as follows: every it- and $::$-free WSML-(Core/Flight/Rule) sentence is an admissible consequence of WSML-(Core/Flight/Rule), and every WSML-(DL/FOL/Full) sentence is an admissible consequence of WSML-(DL/FOL/Full) under strict language layering. Under loose layering, additionally every WSML-(Core/Flight/Rule) sentence is an admissible consequence of WSML-(Core/Flight/Rule). We denote the set of admissible consequences under strict (resp., loose) language layering of a given WSML variant L with $L|_{as}$ (resp., $L|_{al}$).

Definition 1. *Let L_1, L_2 be two WSML variants with associated embeddings (semantics) tr_1, tr_2. Then,*

- *L_2 is* strictly layered *on top of L_1, denoted $L_1 \Rightarrow_s L_2$, if for every theory $\Phi \subseteq L_1$ and every formula $\phi \in L_1|_{as}$, ϕ is a consequence of $tr_1(\Phi)$ if and only if ϕ is a consequence of $tr_2(\Phi)$, and*
- *L_2 is* loosely layered *on top of L_1, denoted $L_1 \Rightarrow_l L_2$, if for every theory $\Phi \subseteq L_1$ and every formula $\phi \in L_1|_{al}$, ϕ is a consequence of $tr_2(\Phi)$ whenever ϕ is a consequence of $tr_1(\Phi)$.*

It turns out that when considering loose language layering, we can consider *generalized* WSML-(Core/Flight/Rule) formulas, which are WSML-(Core/Flight/Rule) formulas which additionally allow it- and $::$-molecules in the body, i.e. for formulas of the form (6) holds that b_i, c_i may be atoms or *arbitrary* molecules. This notion naturally extends to WSML-(Core/Flight/Rule) theories. We thus obtain the *generalized* WSML-(Core/Flight/Rule) language variants.

Theorem 1 (WSML Language Layering).

- *WSML-Core $\Rightarrow_s$ WSML-Flight $\Rightarrow_s$ WSML-Rule $\Rightarrow_s$ WSML-Full.*
- *WSML-Core $\Rightarrow_s$ WSML-DL $\Rightarrow_s$ WSML-FOL $\Rightarrow_s$ WSML-Full.*
- *Gen. WSML-Core $\Rightarrow_l$ gen. WSML-Flight $\Rightarrow_l$ gen. WSML-Rule $\Rightarrow_l$ WSML-Full.*
- *Gen. WSML-Core $\Rightarrow_l$ WSML-DL $\Rightarrow_l$ WSML-FOL $\Rightarrow_l$ WSML-Full.*

An important distinction between the strict and the loose language layering, is that in the strict language layering setting certain schema-level formulas (i.e. those involving it- and $::$- molecules) are not among the admissible consequences. Therefore, it is not possible in WSML-Flight and WSML-Rule to reason about subclass and certain typing relationships, when adhering to strict layering. We illustrate the differences between the two forms of layering with an example.

Example 3. Consider the WSML-Core theory $\Phi = \{Person[hasChild\,\mathsf{it}\,Person], Astronaut :: Person, \forall_a\ x(x : Person \supset x : Animal)\}$ which says that, for every instance of the class $Person$, each value of the attribute $hasChild$ is an instance of $Person$, $Astronaut$ is a subclass of $Person$, and every instance of $Person$ is also an instance of $Animal$. Now consider the formulas $\phi_1 = Astronaut[hasChild\ \mathsf{it}\ Person]$ and $\phi_2 = Person :: Animal$; ϕ_1 and ϕ_2 are both consequences of $tr_{DL}(\Phi)$, but neither is a consequence of $tr_{Core}(\Phi)$ (or indeed $tr_{Flight}(\Phi)$ or $tr_{Rule}(\Phi)$). One can verify that, in fact, the set of consequences of $tr_{Core}(\Phi)$ is a subset of the set of consequences of $tr_{DL}(\Phi)$. Observe also that ϕ_1 and ϕ_2 are not admissible WSML-Core consequences under strict language layering. In fact, the sets of consequences of $tr_{Core}(\Phi)$ and $tr_{DL}(\Phi)$ coincide with respect to admissible WSML-Core consequences under strict language layering, as was demonstrated with Theorem 1.

Comparing strict and loose language layering, we observe that if strict language layering is considered, the definitions of WSML-Flight and WSML-Rule formulas are more restrictive, and there are certain (some may argue, unintuitive) restrictions on the kinds of consequences which are admissible. In fact, under strict language layering, the Core, Flight, and Rule variants are significantly less expressive than the corresponding generalized variants under loose layering, because inferences of it- and $::$ statements may not be considered. Therefore, the use of loose language layering seems more attractive. Indeed, the use of loose language layering is common in Semantic Web standards; for example, RDFS is loosely layered on top of RDF, OWL Full is loosely layered on top of RDFS, and OWL Full is loosely layered on top of OWL DL. However, one could imagine scenarios in which strict language layering is more attractive. For example, when directly using a WSML-DL reasoner for reasoning with WSML-Core theories, one needs to be sure that the semantics correspond; otherwise, certain inferences might be incorrect with respect to the WSML-Core semantics.

7 Correspondence with Target Formalisms

In this section we show the correspondences between the WSML language variants and the logical language formalisms which have originally motivated the definition of these variants, with respect to the reasoning tasks relevant in the formalism. The *target formalisms* for WSML-Core, WSML-DL, WSML-Flight, WSML-Rule and WSML-FOL are $\mathcal{DHL}(\mathbf{D})$, $\mathcal{SHIQ}(\mathbf{D})$, the Well-Founded Semantics for stratified and general logic programs, and (F-Logic-extended) classical first-order logic, respectively.

WSML-Full and WSML-FOL The usual reasoning tasks for autoepistemic logic are existence of stable expansions, inclusion of a formula in some stable expansion, and inclusion of a formula in all stable expansions (autoepistemic consequence) (cf. [16]). We expect these reasoning tasks to be relevant for WSML-Full as well.

From the definition we can see that WSML-FOL does not make use of the non-monotonic modal operator L and thus basically corresponds to F-Logic with concrete domains. The following theorem follows straightforwardly from Proposition 3.

Theorem 2 (WSML-FOL correspondence). *Given a concrete domain scheme $\mathfrak{S}$, a WSML language $\mathcal{L}$, a WSML-FOL theory $\Phi \in \mathcal{L}$ and a formula $\phi \in \mathcal{L}$, then $tr_{FOL}(\Phi) \models_{\mathsf{f}} tr(\phi)$ iff $tr(\phi)$ is a consequence of $tr_{FOL}(\Phi)$.*

WSML-DL and -Core The usual reasoning tasks for the Description Logic $\mathcal{SHIQ}(\mathbf{D})$ are concept satisfiability, knowledge base satisfiability and logical entailment (usually restricted to formulas of a specific shape, such as ground atoms and subsumption axioms). Since these problems can all be reduced to each other [1], we only need to consider the entailment problem.

Theorem 3 (WSML-DL correspondence). *Given a concrete domain scheme $\mathfrak{S}$, if Φ is a WSML-DL theory and ϕ is a WSML-DL axiom, then there are a corresponding $\mathcal{SHIQ}(\mathbf{D})$ theory Φ' and $\mathcal{SHIQ}(\mathbf{D})$ axiom ϕ' (and vice versa) such that $\Phi' \models \phi'$ iff $tr(\phi)$ is a consequence of $tr_{DL}(\Phi)$.*

Proof (Sketch). By definition of WSML-DL we have that for each $\mathcal{SHIQ}(\mathbf{D})$ theory Ψ there is an equivalent WSML-DL theory Φ.

Let Φ be a WSML-DL theory and ϕ be a WSML-DL axiom, and let ϕ' be the (FOL equivalent of a) $\mathcal{SHIQ}(\mathbf{D})$ axiom obtained from ϕ by replacing each molecule of the form $t\!:\!f$ with an atom of the form $f(t)$, each molecule of the form $t_1[r \,\mathsf{hv}\, t_2]$ with an atom of the form $r(t_1, t_2)$, each formula of the form $t_1 :: t_2$ with a formula of the form $\forall x(t_1(x) \supset t_2(x))$, each formula of the form $t_1[t_2 \,\mathsf{it}\, t_3]$ with a formula of the form $\forall x, y(t_1(x) \wedge t_2(x,y) \supset t_3(y))$, and let Φ' be obtained from Φ in the same way, discarding the formulas (3,4,5). It is easy to verify that Φ' and ϕ' are FOL equivalents of a $\mathcal{SHIQ}(\mathbf{D})$ theory and axiom. Using Proposition 1 it is can be verified that $tr_{DL}(\Phi) \models_{\mathsf{f}} tr(\phi)$ iff $\Phi' \models \phi'$ (under standard FOL semantics). The theorem then follows immediately from Theorem 2. $\square$

The following Theorem follows straightforwardly from the proof of Theorem 3 and the definition of WSML-Core.

Theorem 4 (WSML-Core correspondence). *Given a concrete domain scheme $\mathfrak{S}$, if Φ is a WSML-Core theory and ϕ is a $::$- and it-free WSML-Core axiom, then there are a corresponding $\mathcal{DHL}(\mathbf{D})$ theory Φ' and $\mathcal{DHL}(\mathbf{D})$ axiom ϕ' (and vice versa) such that $\Phi' \models \phi'$ iff $tr(\phi)$ is a consequence of $tr_{Core}(\Phi)$.*

WSML-Rule and -Flight The usual reasoning task for the Well-Founded Semantics is ground entailment, i.e. inclusion in the well-founded model. Additionally, as WSML-Flight and WSML-Rule have integrity constraints, consistency checking is also an important reasoning task.

Reasoning in the Well-Founded Semantics can be seen as an approximation to reasoning in the Stable Model Semantics. In fact, given a logic program P, if a ground atom α is true in the well-founded model of P, then α is included in every stable model of P, and thus is entailed under cautious inferencing. In the remainder we consider the Stable Model Semantics because of its close relation to autoepistemic logic.

In the following theorem we establish a correspondence between the stable expansions of a WSML-Rule theory and the stable models of the corresponding logic program. Correspondence with respect to all relevant reasoning tasks follows immediately.

For example, cautious reasoning corresponds to autoepistemic consequence, and consistency checking corresponds to existence of a consistent stable expansion. The theorem follows straightforwardly from Proposition 4.

Theorem 5 (WSML-Rule and WSML-Flight correspondence). *Given a concrete domain scheme $\mathfrak{S}$, if Φ is a WSML-Rule theory, then there is a corresponding normal F-Logic Program P (and vice versa) such that a Herbrand interpretation M of P is a stable model of P iff there is a consistent stable expansion T of $tr_{Rule}(\Phi)$ such that M coincides with the set of objective ground atoms and molecules in T.*

If, additionally, Φ is a WSML-Flight theory, then P has at most one stable model, and Φ is consistent iff P has exactly one model.

8 Conclusions

In this paper we have presented a novel semantic framework for WSML based on FF-AEL, which is first-order autoepistemic logic [14, 3] extended with Frames [13] and concrete domains [2]. Using this framework we have defined a semantics for WSML-Full, and have proposed two paradigms for language layering in WSML. *Strict* language layering requires additional restrictions on the syntax of the WSML variants, but gives more guarantees on the preservation of consequences than *loose* language layering. The WSML group is considering adopting loose language layering for future versions of the language; the main motivation is that it is considered unintuitive to disallow certain inferences (i.e. those involving it- and $::$-molecules).

The approach for defining concrete domains in FF-AEL is very general, and might be applied in the area of Logic Programming, to extend current approaches to built-ins such as the one in Datalog [19], and might be used to extend the support for concrete domains in WSML-DL towards customized data types [17].

Two alternative embeddings for logic programs in FO-AEL have been considered in [3], besides the one we used in this paper (τ_{HP}). The distinguishing feature between the embedding we have considered in this paper, and these two alternative embeddings, is that, using the embedding τ_{HP}, positive rules are translated to Horn formulas, which means that there is a very tight integration between the axioms originating from a DL knowledge base and the rules originating from the logic program, corresponding to our intuition behind WSML-Full as a unifying integrating language.

An alternative formalism which has been used for combining rules and ontologies in a unifying semantics is MKNF [15]. This approach is very similar to ours (however, frames are not considered in MKNF), although the precise relationship between FF-AEL and MKNF remains to be investigated. The embedding of logic programs used in [15] is quite different from the embedding τ_{HP} which we considered in this paper, but it is very close in spirit to the embedding τ_{EH}, which is one of the alternative embeddings considered in [3]. Investigating the relationship between FF-AEL and MKNF, as well as other formalisms which combine Description Logics and Logic Programming (e.g. [7, 18, 11]) is future work. Since positive rules are interpreted as Horn formulas, we conjecture that our semantics corresponds to that of SWRL [11], provided only positive programs are considered, and certain restrictions apply to the allowed concrete domain schemes.

Acknowledgements

We would like to thank the members of the WSML working group, especially Axel Polleres, for valuable discussions about WSML language layering, and we would like to thank two anonymous reviewers for valuable comments on an earlier version of this paper.

References

1. Franz Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds. *The Description Logic Handbook*. Cambridge Univ. Press, 2003.
2. Franz Baader and Philipp Hanschke. A scheme for integrating concrete domains into concept languages. In *IJCAI 1991*.
3. Jos de Bruijn, T. Eiter, A. Polleres, and H. Tompits. Embedding non-ground logic programs into autoepistemic logic for knowledge-base combination. In *IJCAI 2007*.
4. Jos de Bruijn and Stijn Heymans. Translating ontologies from predicate-based to frame-based languages. In *RuleML 2006*.
5. Jos de Bruijn and Stijn Heymans. WSML ontology semantics. WSML Final Draft d28.3, 2007.
6. Jos de Bruijn et al. The web service modeling language WSML. WSML Final Draft D16.1v0.21, 2005.
7. Thomas Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the semantic web. In *KR2004*.
8. Allen Van Gelder, K. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *JACM*, 38(3):620–650, 1991.
9. Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.
10. Benjamin N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *WWW 2003*.
11. Ian Horrocks, P. F. Patel-Schneider, S. Bechhofer, and D. Tsarkov. OWL rules: A proposal and prototype implementation. *J. Web Sem.*, 3(1):23–40, 2005.
12. Ian Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *J. Web Sem.*, 1(1):7–26, 2003.
13. Michael Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *JACM*, 42(4):741–843, 1995.
14. Kurt Konolige. Quantification in autoepistemic logic. *Fundamenta Informaticae*, 15(3–4):275–300, 1991.
15. Boris Motik and Riccardo Rosati. A faithful integration of description logics with logic programming. In *IJCAI 2007*.
16. Ilkka Niemelä. On the decidability and complexity of autoepistemic reasoning. *Fundamenta Informaticae*, 17(1,2):117–155, 1992.
17. Jeff Z. Pan and Ian Horrocks. OWL-Eu: Adding customised datatypes into OWL. *J. Web Sem.*, 4(1):29–39, 2006.
18. Riccardo Rosati. $\mathcal{DL}$+*log*: Tight integration of description logics and disjunctive datalog. In *KR2006*.
19. Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I*. Computer Science Press, 1988.