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 $\mathcal{SHIQ}^D$, WSML-Rule to Logic programming under the Stable Model Semantics (the Well-Founded Semantics can be seen as an approximation), and WSML-Core to $\mathcal{DHL}^D$ (without nominals), a Horn subset of $\mathcal{SHIQ}^D$. 

Copyright © 2007 DERI®, All Rights Reserved. DERI liability, trademark, document use, and software licensing rules apply.
## Contents

1 Introduction .................................................. 4  
2 The Description Logic $SHIQ(D)$ .................................. 7  
3 Frame Logic with Concrete Domains ................................ 9  
4 F-Logic Programs ................................................. 12  
5 FO-AEL with Frames and Concrete Domains ........................ 14  
6 WSML Semantic framework ........................................ 16   
   6.1 WSML Abstract Syntax and Semantics .......................... 16   
   6.2 WSML Language Layering ..................................... 19   
   6.3 Mapping to the WSML Surface Syntax ........................ 21  
7 Correspondence with Target Formalisms .......................... 23  
8 Conclusions ...................................................... 26
1 Introduction

The Web Service Modeling Language WSML\(^1\) [8] is a language for modeling Ontologies and Web Services, based on a number of logical formalisms, which underly the WSML language variants. Figure 1.1(a) shows the different variants and the relationships between them. These variants differ in logical expressiveness and in the underlying language paradigms.

WSML encompasses a framework of variants based on Description Logics [1] and (F-)Logic Programming [10, 12, 20]. Each WSML variant has a target formalism: WSML-Core is based on an intersection of the Description Logic SHIQ\((D)\) and Horn Logic (without equality), called DHL\((D)\) [14]. WSML-DL captures the Description Logic SHIQ\((D)\). 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 [10, 12]. WSML-Rule is based on F-Logic Programming, extended with negation under the Stable Model Semantics [11]\(^2\). WSML-Full extends both WSML-DL and WSML-Rule towards first-order logic with nonmonotonic extensions.

As shown in Figure 1.1(b), WSML has two alternative layerings, namely, WSML-Core ⇒ WSML-DL ⇒ WSML-Full and WSML-Core ⇒ WSML-Flight ⇒ WSML-Rule ⇒ WSML-Full. For both layerings, WSML-Core and WSML-Full mark the least and most expressive layers, respectively. The two layerings are to a certain extent disjoint in the sense that inter-operation in WSML between the Description Logic variant (WSML-DL) on the one hand and the Logic Programming variants (WSML-Flight and WSML-Rule) on the other, is only possible through a common core (WSML-Core) or through a very expressive, undecidable, superset (WSML-Full). The original WSML specification [8] 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.

\(^1\)http://www.wsmo.org/wsml/wsml-syntax

\(^2\)For query answering, the Well-Founded Semantics [10], which was originally considered as a semantic underpinning for WSML-Rule, can be used as an approximation.
In this deliverable, 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) [21, 5], extended with frames [20] and concrete domains [2]. Our approach to concrete domains is a generalization of the approaches typically followed in Description Logics [2] and Datalog [27]. 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. [16]).

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 deliverable we first review the Description Logic \( SHIQ(D) \), in Chapter 2. We then proceed with our definitions of F-Logic with concrete domains, F-Logic Programming, and FF-AEL, in Chapters 3, 4 and 5. We then proceed to describe the abstract syntax for WSML variants, and define strict and loose language layering, in Chapter 6. We demonstrate the correspondence between the variants and the intended target formalisms in Chapter 7. Finally, we conclude the deliverable in Chapter 8.

Note that in this deliverable we only address the semantics of the ontology modeling elements in WSML, and not the semantics of the Web Service and
Goal descriptions. The semantics of the functional and behavioral description of Web Services is a topic of ongoing investigation [18, 19, 26], and beyond the scope of this deliverable.

This deliverable is an extended version of [7].
2 The Description Logic \( \text{SHIQ}(D) \)

The signature \( \Sigma = \langle C, D, R_a, R_c, F_a, F_c \rangle \) of a \( \text{SHIQ}(D) \) language consists of pairwise disjoint sets of concept \( (C) \), datatype \( (D) \), abstract role \( (R_a) \), concrete role \( (R_c) \), individual \( (F_a) \), and data value \( (F_c) \) identifiers. \( \text{SHIQ}(D) \) descriptions are defined as follows, with \( A \) a concept identifier, \( D \) a datatype identifier, \( C, C' \) descriptions, \( S, S' \) abstract role identifiers, \( U, U' \) concrete role identifiers, \( a, b \) individual identifiers, \( o \) a data value identifier, and \( n \) a non-negative integer.

\[
C, C' \rightarrow \bot \mid A \mid C \cap C' \mid \neg C \mid n S.C \mid n U.D \mid \leq n S.C \mid \leq n U.D
\]

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

\[
C \subseteq C' \mid S \subseteq S' \mid U \subseteq U' \mid S \equiv \neg S' \mid (S)^+ \mid C(a) \mid S(a, b) \mid U(a, o) \mid a = b \mid a \neq b
\]

Additionally, we have that in number restrictions \( \geq n S.C \) and \( \leq n S.C \), \( S \) has to be simple, i.e., \( S \) and its sub-roles may not be transitive (with \( (S)^+ \) denoting transitivity).

We present the semantics of \( \text{SHIQ}(D) \) through a translation to first-order logic\(^1\); for a description of the correspondence between the DL semantics and FOL, see e.g. [1]. The function \( \pi \) maps \( \text{SHIQ}(D) \) descriptions to first-order logic formulas with one free variable, \( X \).

The correspondence between \( \text{SHIQ}(D) \) axioms and first-order logic formulas is as follows. The mapping function \( \pi \) extends to \( \text{SHIQ}(D) \) in the natural way.

### Mapping concepts to FOL

<table>
<thead>
<tr>
<th>DL axiom</th>
<th>FOL equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C \subseteq C' )</td>
<td>( \forall x \pi(C(x), x) \equiv \pi(C'(x), x) )</td>
</tr>
<tr>
<td>( S \subseteq S' )</td>
<td>( \forall x, y(S(x, y) \supset S'(x, y)) )</td>
</tr>
<tr>
<td>( U \subseteq U' )</td>
<td>( \forall U(x, y) \equiv U'(x, y) )</td>
</tr>
<tr>
<td>( S \equiv \neg S' )</td>
<td>( \forall x, y(S(x, y) \equiv \neg S'(x, y)) )</td>
</tr>
<tr>
<td>( (S)^+ )</td>
<td>( \forall a, x, y(S(x, y) \equiv \exists S'(x, z)) )</td>
</tr>
<tr>
<td>( C(a) )</td>
<td>( n \pi(C, a) )</td>
</tr>
<tr>
<td>( S(a, b) )</td>
<td>( S(a, b) )</td>
</tr>
<tr>
<td>( U(a, o) )</td>
<td>( U(a, o) )</td>
</tr>
<tr>
<td>( a = b )</td>
<td>( a = b )</td>
</tr>
<tr>
<td>( a \neq b )</td>
<td>( (a = b) )</td>
</tr>
</tbody>
</table>

We now describe the syntax of \( \text{DHC}(D) \) ([14]), which is a Datalog subset of \( \text{SHIQ}(D) \). \( \text{DHC}(D) \) descriptions are of the following form. \( C_L, C'_L \) (resp. \( C_R, C'_R \)) are descriptions which are allowed only on the left-(resp. right-)hand side of the inclusion symbol \( \subseteq \).

\[
C, C' \rightarrow A \mid C \cap C' \mid C_R, C'_R \rightarrow C \mid \forall S.C.R \mid \forall U.D
\]

\[
C_L, C'_L \rightarrow C \mid C_L \sqcup C'_L \mid \exists S.C.L \mid \exists U.D \mid \geq 1 S \mid \geq 1 U
\]

\(^1\)It is easy to verify that the semantics we use for concrete domains corresponds to the semantics usually considered in Description Logics when considering \( \text{SHIQ}(D) \) (e.g. [2]).
A $\mathcal{DHL}(\mathbf{D})$ ontology consists of axioms of the following forms.

$$C_L \subseteq C_R \mid S \subseteq S' \mid U \subseteq U' \mid S \equiv S' - \mid (S)^+ \mid \top \subseteq \forall S.C_R \mid \top \subseteq \forall U.D \mid A(a) \mid S(a, b) \mid U(a, o)$$
3 Frame Logic with Concrete Domains

In this section we review F-Logic, following [6], 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 [27].

A language $\mathcal{L}$ has a signature of the form $\Sigma_{\mathcal{L}} = (\mathcal{F}, \mathcal{P}, \mathcal{F}^D, \mathcal{P}^D)$, 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.

Formulas are constructed in the usual way from atoms and molecules using the symbols $\neg, \land, \lor, \Rightarrow, \equiv, \exists, \forall$. Variables quantified using an abstract quantifier ($\exists_a, \forall_a$) may not occur in a concrete term or atom.

An interpretation is a tuple $\mathcal{I} = (\mathcal{U}, \mathcal{U}^D, \prec_U, \in_U, \mathcal{I}_F, \mathcal{I}_P, \mathcal{I}_\bot)$. $\mathcal{U}$ and $\mathcal{U}^D$ are disjoint non-empty countable sets, called the abstract and concrete domains, $\prec_U$ is an irreflexive partial order over $\mathcal{U} \cup \mathcal{U}^D$, and $\in_U$ is a binary relation over $\mathcal{U} \cup \mathcal{U}^D$. We write $a \prec_U b$ when $a \prec_U b$ or $a = b$, for $a, b \in \mathcal{U} \cup \mathcal{U}^D$. Each interpretation holds that if $a \in_U b$ and $b \in_U c$ then $a \in_U c$. Thus, if $b \in_U c$, then $\{k \mid k \in_U b, k \in \mathcal{U} \cup \mathcal{U}^D\} \subseteq \{k \mid k \in_U c, k \in \mathcal{U} \cup \mathcal{U}^D\}$. We call the set $\{k \mid k \in_U b, k \in \mathcal{U} \cup \mathcal{U}^D\}$ the class extension of $b$. Thus, if $b \in_U c$, 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 $\mathcal{U}$: $\mathcal{I}_F(f) : \mathcal{U}^i \rightarrow \mathcal{U}$. An $n$-ary abstract function symbol $f \in \mathcal{F}^D$ is interpreted as a function over the domain $\mathcal{U}^D$: $\mathcal{I}_F(f) : (\mathcal{U}^D)^n \rightarrow \mathcal{U}^D$. An abstract predicate symbol $p \in \mathcal{P}$ is interpreted as a relation over the domain $\mathcal{U} \cup \mathcal{U}^D$: $\mathcal{I}_P(p) \subseteq (\mathcal{U} \cup \mathcal{U}^D)^i$, for every $i \geq 0$. An $n$-ary abstract predicate symbol $p \in \mathcal{P}^D$ is interpreted as a relation over the domain $\mathcal{U}^D$: $\mathcal{I}_P(p) \subseteq (\mathcal{U}^D)^n$. $\mathcal{I}_\bot$ associates a binary relation over $\mathcal{U} \cup \mathcal{U}^D$ with each $u \in \mathcal{U} \cup \mathcal{U}^D$: $\mathcal{I}_\bot(u) \subseteq (\mathcal{U} \cup \mathcal{U}^D) \times (\mathcal{U} \cup \mathcal{U}^D)$.

A concrete domain scheme $\mathcal{S}$ is a tuple $\mathcal{S} = (\mathcal{U}^\mathcal{S}, \mathcal{F}^\mathcal{S}, \mathcal{P}^\mathcal{S}, \cdot^{\mathcal{S}})$, where $\mathcal{U}^\mathcal{S}$ is a non-empty countable set of concrete values, $\mathcal{F}^\mathcal{S}$ and $\mathcal{P}^\mathcal{S}$ are disjoint sets of concrete function and predicate symbols, each with an associated nonnegative arity $n$, and $\cdot^{\mathcal{S}}$ is an interpretation function which assigns a function $f^{\mathcal{S}} : (\mathcal{U}^\mathcal{S})^n \rightarrow \mathcal{U}^\mathcal{S}$ to every $f \in \mathcal{F}^\mathcal{S}$ and a relation $p^{\mathcal{S}} \subseteq (\mathcal{U}^\mathcal{S})^n$ to every $p \in \mathcal{P}^\mathcal{S}$. A language $\mathcal{L}$ with signature $\Sigma_{\mathcal{L}} = (\mathcal{F}, \mathcal{P}, \mathcal{F}^D, \mathcal{P}^D)$ conforms to a concrete...
domains scheme \( \mathcal{S} = \langle U^\mathcal{S}, \mathcal{F}^\mathcal{S}, \mathcal{P}^\mathcal{S}, \mathcal{E} \rangle \) if \( \mathcal{F}^D = \mathcal{F}^\mathcal{S} \) and \( \mathcal{P}^D = \mathcal{P}^\mathcal{S} \). An interpretation \( \mathbf{I} = (U, U^D, \prec_U, \in_U, \mathbf{I}_P, \mathbf{I}_{\text{ins}}, \mathbf{I}_{\text{bld}}) \) of \( \mathcal{L} \) conforms to \( \mathcal{S} \) if \( U^D = U^\mathcal{S} \), and \( \mathbf{I}_P(f) = f^\mathcal{S}, \mathbf{I}_P(p) = p^\mathcal{S} \) for every \( f \in \mathcal{F}^\mathcal{S}, p \in \mathcal{P}^\mathcal{S} \), 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 \( \mathcal{S} = \langle U^\mathcal{S}, \mathcal{F}^\mathcal{S}, \mathcal{P}^\mathcal{S}, \mathcal{E} \rangle \) as follows: \( U^\mathcal{S} \) is the union of the sets of integer numbers and finite-length sequences of Unicode characters. \( \mathcal{F}^\mathcal{S} \) 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. \( \mathcal{P}^\mathcal{S} \) consists of unary predicate symbols \( \text{integer} \) and \( \text{string} \), and the binary predicate \( \text{symbol numeric-equals} \). The interpretation function \( \cdot^\mathcal{S} \) interprets (signed) sequences of decimal digits and "-delimited sequences of characters as integers and strings, respectively, in the natural way; \( \cdot^\mathcal{S} \) interprets the unary predicate symbols \( \text{integer} \) and \( \text{string} \) as the sets of integers and strings, respectively; finally, \( \cdot^\mathcal{S} \) interprets numeric-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 [24], and Datalog [27] (where they are called built-ins). In DLs, all predicate symbols are sorted (using the sorts \( \text{abstract} \) and \( \text{concrete} \); binary predicates with the sort \( \text{abstract} \times \text{concrete} \) are usually called \( \text{features} \)) 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 \( \mathbf{I}^B_t \), is defined as: \( \mathbf{I}^B_t = t^B \) if \( t \in \mathcal{V} \), and \( \mathbf{I}^B_t = \mathbf{I}_P(f)(\mathbf{I}^{t_1^B} t_{1}, \ldots, \mathbf{I}^{t_n^B} t_{n}) \) if \( t \) is of the form \( f(t_1, \ldots, t_n) \). A variable substitution \( \beta \), usually written in postfix notation, is a partial mapping from variable symbols to ground terms. A variable substitution \( \beta \) is associated with (cf. [5]) a variable assignment \( B \) if for every variable symbol \( x \) such that \( x^B = k \) and there exists a ground term \( t \) such that \( \mathbf{I}^B_t = k \), then \( x^{\beta} = t' \) for some ground term \( t' \) such that \( t'^{B \beta} = k \); if no such \( t \) exists, \( x^{\beta} \) is not defined.

Satisfaction of atomic formulas and molecules \( \phi \) in \( \mathbf{I} \), given the variable assignment \( B \), denoted \( (\mathbf{I}, B) \models \phi \), is defined as:

- \((\mathbf{I}, B) \models \top\),
- \((\mathbf{I}, B) \not\models \bot\),
- \((\mathbf{I}, B) \models p(t_1, \ldots, t_n) \text{ if } \mathbf{I}^{t_1^B} \cdots \mathbf{I}^{t_n^B} \in \mathbf{I}_P(p)\),
- \((\mathbf{I}, B) \models t_1 : t_2 \text{ if } \mathbf{I}^{t_1^B} \in_U \mathbf{I}^{t_2^B}\).
• \( (I, B) \models t_1 :: t_2 \text{ iff } t_1^I B \subseteq_U t_2^I B \),

• \( (I, B) \models t_1[t_2 \mapsto t_3] \text{ iff } \langle t_1^I B, t_3^I B \rangle \in \mathbf{I}_\to(t_2^I B), \) and

• \( (I, B) \models t_1 = t_2 \text{ iff } t_1^I B = t_2^I B \).

This extends to arbitrary formulas as follows:

• \( (I, B) \models \phi_1 \land \phi_2 \text{ iff } (I, B) \models \phi_1 \) and \( (I, B) \models \phi_2 \),

• \( (I, B) \models \phi_1 \lor \phi_2 \text{ iff } (I, B) \models \phi_1 \) or \( (I, B) \models \phi_2 \),

• \( (I, B) \models \neg \phi_1 \text{ iff } (I, B) \notin \phi_1 \),

• \( (I, B) \models \forall_a x(\phi_1) \text{ iff for every } B'_a \text{ which is an abstract } x\text{-variant of } B, (I, B'_a) \models \phi_1 \),

• \( (I, B) \models \exists_a x(\phi_1) \text{ iff for some } B'_a \text{ which is an abstract } x\text{-variant of } B, (I, B'_a) \models \phi_1 \),

• \( (I, B) \models \forall_c x(\phi_1) \text{ iff for every } B'_c \text{ which is a concrete } x\text{-variant of } B, (I, B'_c) \models \phi_1 \),

• \( (I, B) \models \exists_c x(\phi_1) \text{ iff for some } B'_c \text{ which is a concrete } x\text{-variant of } B, (I, B'_c) \models \phi_1 \).

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 \( \mathcal{S} \), an interpretation \( I \) is a model of a formula \( \phi \) if \( I \) conforms to \( \mathcal{S} \) and for every variable assignment \( B, (I, B) \models \phi \). A formula \( \phi \) is satisfiable if it has a model; \( \phi \) is valid if every interpretation which conforms to \( \mathcal{S} \) is a model of \( \phi \). 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 \( \Phi \) is also a model of \( \phi \).

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 \( \models \).

Given a classical FOL formula (resp., theory) \( \phi \) (resp., \( \Phi \)), then \( \delta(\phi) \) (resp., \( \delta(\Phi) \)) is the F-Logic formula (resp., theory) obtained from \( \phi \) (resp., \( \Phi \)) 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 \( t[R \rightarrow t_2] \), respectively. The following correspondence between \( SHIQ(D) \) and F-Logic ontologies is a straightforward extension of a result in [6].

**Proposition 1.** Given a concrete domain scheme \( \mathcal{S} \), let \( \Phi, \phi \) be a \( SHIQ(D) \) theory and formula, respectively. Then, \( \Phi \models \phi \text{ iff } \delta(\pi(\Phi)) \models \delta(\pi(\phi)) \).
4 F-Logic Programs

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

\[
h \leftarrow b_1, \ldots, b_m, \text{not } c_1, \ldots, \text{not } c_n,
\]

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: \((\ast)\) \( x::z \leftarrow x::y, y::z \), \((\ast\ast)\) \( x::z \leftarrow x::y, y::z \), and \((\ast\ast\ast)\) \( x::x \), where \((\ast)\) axiomatizes transitivity of the subclass relation; \((\ast\ast)\) axiomatizes inheritance of class membership; and \((\ast\ast\ast)\) axiomatizes the fact that every class is a subclass of itself\(^4\). A normal F-Logic program \( P \) is a set of rules of the form (4.1) which includes the rules \((\ast, \ast\ast, \ast\ast\ast)\). 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 \( \mathcal{S} \), \( \top \in M \), \( \bot \notin M \), 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 [12], 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) 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 \( \Phi \) is obtained by replacing the arrow \( \leftarrow \) and comma (,) in every rule with the symbols \( \lor \) and \( \land \) 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 \( \mathcal{S} \), let \( P \) be a positive logic program and \( \Phi \) be the corresponding Horn F-Logic theory, then

- \( P \) has a stable model iff \( \Phi \) is satisfiable, and
- if \( P \) has a stable model \( M \), it is unique, and for every ground atom or molecule \( \alpha \), \( \alpha \in M \) iff \( \Phi \models \alpha \).

\(^4\)Note that the rule (4) is not safe. However, \((\ast\ast\ast)\) is not necessary in case subclass statements \((::)\) do not occur in rule bodies and are not considered when determining consequences.
Proof. Let $\Phi$ be an F-Logic theory and let $\Phi'$ be the FOL theory obtained from $\Phi$ by replacing every molecule of the form $t_1 : t_2$ with an atomic formula of the form $isa(t_1, t_2)$, molecules of the form $t_1 :: t_2$ with atomic formulas of the form $subclass(t_1, t_2)$, and molecules of the form $t_1[t_2 \rightarrow t_3]$ with atomic formulas of the form $att(t_1, t_2, t_3)$, with $t_1, t_2, t_3$ terms and $isa, subclass, att$ predicate symbols which do not occur in $\Phi$, and adding the following formulas:

$$\forall a \ x, y, z \ (subclass(x, y) \land subclass(y, z) \supset subclass(x, z)),$$
$$\forall a \ x, y, z \ (isa(x, y) \land subclass(y, z) \supset isa(x, z)),$$ and
$$\forall a \ x \ (subclass(x, x)).$$

It is easy to verify that the F-Logic models of $\Phi$ and the FOL models of $\Phi'$ are isomorphic. The Theorem follows immediately from the correspondence between minimal Herbrand models and first-order entailment, and the correspondence between minimal Herbrand models and stable models for positive programs. □
First-Order Autoepistemic Logic with Frames and Concrete Domains

First-Order Autoepistemic Logic (FO-AEL) [21, 5] 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 \( L \) is defined relative to a language \( L \):

- any atomic formula or molecule in \( L \) is a formula in \( L \),
- if \( \phi \) is a formula in \( L \), then \( L\phi \), called a modal atom, is a formula in \( L \), 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 \( \langle I, \Gamma \rangle \), where \( I = \langle U, U^D, \prec_U, \in_U, I_F, I_P, \ldots \rangle \) is an interpretation, and \( \Gamma \subseteq \mathcal{L}_k \) is a set of sentences, called the belief set. Satisfaction of objective atomic formulas in \( \langle I, \Gamma \rangle \) corresponds to satisfaction in \( I \).

Satisfaction of a formula \( L\phi (\phi \in \mathcal{L}_k) \) in an interpretation \( \langle I, \Gamma \rangle \) with respect to a variable assignment \( B \) under the any-name semantics\(^1\), denoted \( \langle I, B \rangle \models_\Gamma L\phi \), is defined as follows:

\[
\langle I, B \rangle \models_\Gamma L\phi \text{ iff, for some variable substitution(s) } \beta, \text{ associated with } B, \phi\beta \text{ has no free variables and } \phi\beta \in \Gamma.
\]

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

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

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 \( T \subseteq \mathcal{L}_k \) is a stable expansion of a base set \( A \subseteq \mathcal{L}_k \) iff \( T = \{ \phi \mid A \models_\Gamma \phi \} \).

A formula \( \phi \) is an autoepistemic consequence of \( A \) if \( \phi \) 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 [21].

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

**Embedding Logic Programs** Following [5], 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

\(^1\)[21] presents also the all-names semantics, but we follow [21, 5] in their choice for the any-name semantics.
In FF-AEL, it is necessary to axiomatize default uniqueness of names. With $UNA_\Sigma$ we denote the set of axioms
\[ \neg L(t_1 = t_2) \supset t_1 \neq t_2, \]
for all pairs of distinct ground terms $t_1, t_2$.

Let $r$ be a normal rule of the form (4.1). Then,
\[ \tau_{HP}(r) = (\forall) \bigwedge_{1 \leq i \leq m} b_i \land \bigwedge_{1 \leq j \leq n} \neg L c_j \supset h, \]
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:
\[ \tau_{HP}(P) = \{ \tau_{HP}(r) \mid r \in P \} \cup UNA_\Sigma. \]

Note that [5] considered three different embeddings of normal logic programs. We have chosen to use the embedding $\tau_{HP}$ for WSML because it allows for a very tight integration between the rules in the logic program and the Description Logic axioms, and it generalizes several current approaches to combining DL and LP such as SWRL [15] and $DL + log$ [25] when considering only positive programs.

Recall the three rules ($\ast$), ($\ast\ast$) and ($\ast\ast\ast$), which are part of every F-Logic program. These rules translate to FF-AEL as follows: ($\ast$) $\forall a x, y, z (x :: y \land y :: z \supset x :: z)$, ($\ast\ast$) $\forall a x, y, z (x :: y \land y :: z \supset x :: z)$ and ($\ast\ast\ast$) $\forall a x (x :: x)$. 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). Therefore, one may disregard these rules in the translation. Faithfulness of the embedding is established in the following proposition, which generalizes a result in [5].

**Proposition 4.** Given a concrete domain scheme $\mathcal{S}$, a Herbrand interpretation $M$ of a normal F-Logic program $P$ is a stable model of $P$ if and 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 $T$.

**Proof.** Follows immediately from [5, Theorem 1] and Theorem 2. \qed
6 WSML Semantic framework

In this chapter we specify an abstract syntax and semantics for WSML, based on FF-AEL, in Section 6.1. We discuss language layering in WSML in Section 6.2. Finally, we present the mapping from the abstract syntax to the concrete surface syntax, which was used in the original WSML specification, in Section 6.3.

6.1 WSML Abstract Syntax and Semantics

We present an abstract syntax for WSML logical expressions, and define their semantics through an embedding in FF-AEL. 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, shown in Section 6.3.

Given a concrete domain scheme $S$, the signature of a WSML language $L$ is of the form $\Sigma = \langle F, P, F^S, P^S \rangle$, 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 $t_1[t_2 \times t_3]$, with $x \in \{\text{ot}, \text{it}, \text{hv}\}$, are molecules. The symbol $\text{ot}$ stands for the WSML construct ofType; a statement $t_1[t_2 \text{ ot} t_3]$ 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 $t_1[t_2 \text{ it} t_3]$ implies that all values for the attribute $t_2$ are a member of the class $t_3$; $\text{hv}$ stands for the WSML construct hasValue; a molecule $t_1[t_2 \text{ hv} t_3]$ says that the individual $t_1$ has an attribute $t_2$ with value $t_3$.

WSML formulas are inductively defined as follows, with $\phi, \psi \in L$:
- atoms and molecules are formulas;
- $\sim \phi$, with $\sim \in \{\neg, \text{not}\}$, is a formula;
- $\phi \ast \psi$, with $\ast \in \{\land, \lor, \supset, \equiv\}$, is a formula; and
- $Qx(\phi)$, with $Q \in \{\forall a, \exists a, \forall c, \exists c\}$ and $x \in V$, 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 $\_\text{it}, \_\text{ot}$ 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 $\Phi$ 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 $\phi$ by replacing each occurrence of $\text{not}$ with $\neg L$, replacing $\text{hv}$ with $\rightarrow$, and replacing molecules of the forms $t_1[t_2 \_\text{ot} t_3]$ and $t_1[t_2 \_\text{it} t_3]$, with $t_1, t_2$ and $t_3$ terms, with atoms of the forms $\_\text{ot}(t_1, t_2, t_3)$ and $\_\text{it}(t_1, t_2, t_3)$, respectively. Finally, the following formulas are

\footnote{It is assumed in WSML that such a concrete domain scheme incorporates at least the XML Schema datatypes string, integer, and decimal [8, Appendix C].}
used to axiomatize the intentional (only if) semantics of the \( \text{ot} \) and it molecules:

\[
\forall a \ x,y,z,v,w \ \ (\text{ot}(x,y,z) \land v:x \land v[y \rightarrow w] \land \neg L:z \supset \bot); \quad (6.1)
\]

\[
\forall a \ x,y,z,v,w \ \ (\text{it}(x,y,z) \land v:x \land v[y \rightarrow w] \supset w:z), \quad (6.2)
\]

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

\[
\forall a \ x,y,z \ \ (\forall a v,w (v:x \land v[y \rightarrow w] \supset w:z)) \supset \text{it}(x,y,z), \quad (6.3)
\]

\[
\forall a \ x,y \ \ (\forall a v (v:x \supset v:y)) \supset x::y. \quad (6.4)
\]

**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 \( \Phi \) is given through the embedding \( tr_{\text{Full}}(\Phi) = tr(\Phi) \cup \{ (6.1), (6.2), (6.3), (6.4) \} \).

A WSML-FOL sentence is a WSML-Full sentence which neither contains \( \text{ot} \)-molecules, nor occurrences of the default negation operator \( \text{not} \). A WSML-FOL theory is a set of WSML-FOL sentences. The semantics of a WSML-FOL theory \( \Phi \) is given through the embedding \( tr_{\text{FOL}}(\Phi) = tr(\Phi) \cup \{ (6.2), (6.3), (6.4) \} \).

**WSML-Rule** WSML-Rule formulas are of the form

\[
(\forall)b_1 \land \ldots \land b_l \land \text{not} \ c_1 \land \ldots \land \text{not} \ c_m \supset h \quad (6.5)
\]

where \( b_1, \ldots, b_l, c_1, \ldots, c_m \) are atoms or \( hv \), \( \text{ot} \), or \( \text{isa} (:) \) molecules, with \( l, m \) nonnegative 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 \( \Phi \) is given through the embedding \( tr_{\text{Rule}}(\Phi) = tr(\Phi) \cup \{ (6.1), (6.2) \} \).

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 (6.5) and rules in a logic program of the form (4.1). Thus, WSML-Rule formulas are essentially rules with a head and a body. Notice that, whereas the embedding \( tr_{\text{Full}}(\Phi) \) includes the sentences (6.3) and (6.4), the embedding \( tr_{\text{Rule}}(\Phi) \) does not, because there is no natural correspondence to rules due to the universal quantification in the antecedent of the formulas (6.3) and (6.4). 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 \( \Phi \) consisting of the formulas

\[
\forall a x(x:A \supset x:B) \text{ and } \ A::B \supset q.
\]

The theory \( tr_{\text{Rule}}(\Phi) \) neither has \( A::B \) nor \( q \) among its consequences. In contrast, it is easy to verify that, by (6.4), \( tr_{\text{Full}}(\Phi) \) has both \( A::B \) and \( q \) among its consequences.
Notice that a theory $\Phi = \{ \forall_a x (A \supset x : B) \}$ does not have $A :: B$ among its consequences under WSML-Rule semantics, but it does under WSML-Full semantics. Therefore, certain restrictions on the consequences we consider are required in order to achieve the desired language layering. These restrictions will be precisely defined in the language layering theorem at the end of this section.

**WSML-Flight** A *WSML-Flight theory* is a WSML-Rule theory for which holds that, for every formula of the form (6.5), every variable occurs in a positive abstract body atom $b_i$, no function symbol in (6.5) is used with an arity higher than 0, and the theory is locally stratified. The semantics of a WSML-Flight theory $\Phi$ is given through the embedding $tr_{Flight}(\Phi) = tr_{Rule}(\Phi)$.

**WSML-DL** Given an FOL formula $\phi$, $\delta'(\phi)$ is obtained from $\phi$ 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 $t_1 : A, t_1 \circ [R hv t_2]$. Given a SHIQ($D$) signature $\Sigma = \langle C, D, R_a, R_c, F_a, F_c \rangle$, the corresponding WSML signature is $\langle C \cup D \cup R_a \cup R_c \cup F_a, \emptyset, F_c', D' \rangle$, where $F_c'$ is the set of 0-ary functions symbols obtained from $F_c$ and $D'$ is the set of 1-ary predicate symbols obtained from $D$. A WSML-DL formula is a WSML formula of the form

- $\delta'(\phi)$, where $\phi$ is the FOL equivalent of a SHIQ($D$) axiom of the signature $\Sigma$,
- $a :: b$, with $a, b \in C$,
- $a [s \circ b]$, with $s \in R_a$ and $a, b \in C$, or
- $a [u \circ d]$, with $u \in R_c$, $a \in C$ and $d \in D$.

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

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

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

---

2 Each atom or molecule in $gr(\Phi)$ is assigned a stratum, which is an integer. We say that $gr(\Phi)$ 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 $gr(\Phi)$ is stratified, then $\Phi$ is locally stratified.

3 These conditions correspond to the usual safety condition which must hold for Datalog programs, and the usual local stratification for logic programs.
6.2 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 $\Phi$, $\text{tr}_{\text{Core}}(\Phi) = \text{tr}_{\text{Flight}}(\Phi) = \text{tr}_{\text{Rule}}(\Phi)$); however, there are also certain differences between embeddings (e.g., given a WSML-Core theory $\Phi$, $\text{tr}_{\text{Core}}(\Phi) \neq \text{tr}_{\text{DL}}(\Phi) \neq \text{tr}_{\text{Full}}(\Phi)$). 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 theory.

The admissible consequences under strict language layering for WSML are as follows: every $\textit{-}$ and $\text{::}$-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 $L_{|\text{as}}$ (resp., $L_{|\text{al}}$).

**Definition 1.** Let $L_1, L_2$ be two WSML variants with associated embeddings (semantics) $\text{tr}_1, \text{tr}_2$. 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 $\phi \in L_1_{|\text{as}}$, $\phi$ is a consequence of $\text{tr}_1(\Phi)$ if and only if $\phi$ is a consequence of $\text{tr}_2(\Phi)$, 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 $\phi \in L_1_{|\text{al}}$, $\phi$ is a consequence of $\text{tr}_2(\Phi)$ whenever $\phi$ is a consequence of $\text{tr}_1(\Phi)$.

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 $\textit{-}$ and $\text{::}$-molecules in the body, i.e. for formulas of the form (6.5) 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).

- $\text{WSML-Core} \Rightarrow_s \text{WSML-Flight} \Rightarrow_s \text{WSML-Rule} \Rightarrow_s \text{WSML-Full}$.
- $\text{WSML-Core} \Rightarrow_s \text{WSML-DL} \Rightarrow_s \text{WSML-FOL} \Rightarrow_s \text{WSML-Full}$.
- $\text{Gen. WSML-Core} \Rightarrow_{\text{al}} \text{Gen. WSML-Flight} \Rightarrow_{\text{al}} \text{Gen. WSML-Rule} \Rightarrow_{\text{al}} \text{WSML-Full}$.
- $\text{Gen. WSML-Core} \Rightarrow_{l} \text{WSML-DL} \Rightarrow_{l} \text{WSML-FOL} \Rightarrow_{l} \text{WSML-Full}$.

**Sketch.** For strict layering, the results follow from the following results, which can be easily verified.
The semantics (i.e. the embedding functions $tr_{Core}$, $tr_{Flight}$, $tr_{Rule}$) of WSML-Core, Flight, and Rule correspond.

The semantics (i.e. the embedding functions $tr_{DL}$, $tr_{FOL}$) of WSML-DL and FOL correspond.

Let $\Phi$ be a WSML-Core (resp., Rule, FOL) theory and let $\phi$ be an admissible WSML-Core (resp., Rule, FOL) consequence under strict layering. Then, $\phi$ is a consequence of $tr_{Core}(\Phi)$ (resp., $tr_{Rule}(\Phi)$, $tr_{FOL}(\Phi)$) iff $\phi$ is a consequence of $tr_{DL}(\Phi)$ (resp., $tr_{Full}(\Phi)$, $tr_{Full}(\Phi)$).

For loose layering the results follow from strict layering, combined with the following result.

Let $\Phi$ be a generalized WSML-Core (resp., Rule) theory and let $\phi$ be an admissible WSML-Core (resp., Rule) consequence under loose layering. Then, $\phi$ is a consequence of $tr_{DL}(\Phi)$ (resp., $tr_{Full}(\Phi)$) whenever $\phi$ is a consequence of $tr_{Core}(\Phi)$ (resp., $tr_{Rule}(\Phi)$).

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 $\Phi = \{Person[hasChild it Person], Astronaut :: Person, \gamma_a x[\forall x: Person \supset x: Animal]\}$ 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 $\phi_1 = Astronaut[hasChild it Person]$ and $\phi_2 = Person :: Animal$; $\phi_1$ and $\phi_2$ 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 $\phi_1$ and $\phi_2$ 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-
6.3 Mapping to the WSML Surface Syntax

Table 6.1 shows the mapping from the WSML abstract syntax to the surface syntax specified in the language reference [8]. In the table, $t_1, \ldots, t_n$ are terms, $p$ is a predicate symbol, $x$ is a variable, $b_1, \ldots, b_l, c_1, \ldots, c_m, h$ are atoms or molecules, and $\phi, \psi$ are formulas.

Notice that the original WSML specification allows complex formulas in both the bodies and the heads of the rules of WSML-Flight and WSML-Rule ontologies. These can be eliminated through the usual Lloyd-Topor transformations described in the specification. Compound molecules can be eliminated by replacing them with conjunctions of molecules. Notice also that the WSML surface syntax allows anonymous identifiers; these simply correspond to ground terms in the abstract syntax in the usual way.

The WSML language reference [8] also specifies a conceptual syntax. The reference document describes the translation from the conceptual syntax to the logical expression syntax, which is used to obtain the semantics. Future versions
of WSML will consider using an abstract syntax similar to the syntax specified in this document, with extensions for capturing the conceptual elements in ontology, Web service, goal, and mediator descriptions.
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 DHL(D), SHIQ(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. [23]). 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 nonmonotonic modal operator $\mathcal{L}$ 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 $\mathcal{S}$, a WSML language $L$, a WSML-FOL theory $\Phi \in L$ and a formula $\phi \in L$, then $\text{tr}_{\text{FOL}}(\Phi) \models_{\text{f}} \text{tr}(\phi)$ iff $\text{tr}(\phi)$ is a consequence of $\text{tr}_{\text{FOL}}(\Phi)$.

**Proof.** Follows from the definition of WSML-FOL, the definition of $\models_{\text{f}}$, and the results in [21].

**WSML-DL and -Core** The usual reasoning tasks for the Description Logic SHIQ(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 $\mathcal{S}$, if $\Phi$ is a WSML-DL theory and $\phi$ is a WSML-DL axiom, then there are a corresponding SHIQ(D) theory $\Phi'$ and SHIQ(D) axiom $\phi'$ (and vice versa) such that $\Phi' \models_{\text{f}} \phi'$ iff $\text{tr}(\phi)$ is a consequence of $\text{tr}_{\text{DL}}(\Phi)$.

**Sketch.** By definition of WSML-DL we have that for each SHIQ(D) theory $\Psi$ there is an equivalent WSML-DL theory $\Phi$.

Let $\Phi$ be a WSML-DL theory and $\phi$ be a WSML-DL axiom, and let $\phi'$ be the (FOL equivalent of a) SHIQ(D) axiom obtained from $\phi$ by replacing each molecule of the form $t : f$ with an atom of the form $f(t)$, each molecule of the form $t_1 \mathcal{R} t_2$ with an atom of the form $r(t_1, t_2)$, each formula of the form $\forall x(t_1(x) \Rightarrow t_2(x))$, each formula of the form $t_1[t_2 \mathcal{R} t_3]$ with a formula of the form $\forall x,y(t_1(x) \land t_2(x,y) \Rightarrow t_3(y))$, and let $\Phi'$ be obtained from $\Phi$ in the same way, discarding the formulas (6.2,6.3,6.4). It is easy to verify that $\Phi'$ and $\phi'$ are FOL equivalents of a SHIQ(D) theory and axiom. Using Proposition 1 it is can be verified that $\text{tr}_{\text{DL}}(\Phi) \models_{\text{f}} \text{tr}(\phi)$ iff $\Phi' \models_{\text{f}} \phi'$.
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 $\mathcal{S}$, if $\Phi$ is a WSML-Core theory and $\phi$ is a :: - and it - free WSML-Core axiom, then there are a corresponding $\mathcal{DHL}(\mathcal{D})$ theory $\Phi'$ and $\mathcal{DHL}(\mathcal{D})$ axiom $\phi'$ (and vice versa) such that $\Phi' \models \phi'$ iff $\text{tr}(\phi)$ is a consequence of $\text{tr}_{\text{Core}}(\Phi)$.

**WSML-Rule and -Flight** A common reasoning task for the Stable Model Semantics is inference of ground atoms, i.e. inclusion of ground atoms in all stable models. Additionally, as WSML-Flight and WSML-Rule have integrity constraints, consistency checking is also an important reasoning task.

Reasoning in the Well-Founded Semantics [10] can be seen as an approximation to reasoning in the Stable Model Semantics. In fact, given a logic program $P$, if a ground atom $\alpha$ is true in the well-founded model of $P$, then $\alpha$ is included in every stable model of $P$, and thus is entailed under cautious inferencing.

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 $\mathcal{S}$, if $\Phi$ 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 $\text{tr}_{\text{Rule}}(\Phi)$ such that $M$ coincides with the set of objective ground atoms and molecules in $T$.

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

**Proof.** It is easy to verify that for each WSML-Rule theory $\Phi$, there is an F-Logic program $P$ such that $\tau_{HP}(P) = \text{tr}_{\text{Rule}}(\Phi)$, and vice versa. The correspondence follows straightforwardly from Proposition 4.

The results about WSML-Flight theories follow immediately from the results about locally stratified programs [10].

These results on the relationship between WSML and the target formalisms, combined with the existing results about complexity of reasoning in these formalisms [1, 4, 3, 17, 13], leads to the following characterization of the complexity of reasoning in WSML, not considering concrete domains. For the task of query answering we usually consider data complexity, which is the complexity measured in the size of the data (ground statements); the size of the formulas (ontology) is usually considered fixed.

**Theorem 6 (Complexity of WSML without concrete domains).**
• Consistency checking and ground entailment (and hence query answering) in WSML-Core and WSML-Flight are complete for $P$ with respect to the size of the data.

• Consistency checking and ground entailment (and hence query answering) in WSML-Rule are complete for $\Pi_1^1$, and thus undecidable.

• Consistency checking and entailment in WSML-Full are $\Pi_1^1$-hard.

• Instance checking and conjunctive query answering in WSML-DL are complete for $\text{coNP}$ with respect to the size of the data.

• Consistency checking (resp., entailment) in WSML-DL are complete for $\text{ExpTime}$ with respect to the size of the knowledge base (resp., the knowledge bases and the query).
8 Conclusions

In this paper we have presented a novel semantic framework for WSML based on FF-AEL, which is first-order autoepistemic logic [21, 5] extended with Frames [20] 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 \textit{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 [27], and might be used to extend the support for concrete domains in WSML-DL towards customized data types [24].

Two alternative embeddings for logic programs in FO-AEL have been considered in [5], besides the one we used in this paper (\(\tau_{HP}\)). The distinguishing feature between the embedding we have considered in this paper, and these two alternative embeddings, is that, using the embedding \(\tau_{HP}\), 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 [22]. 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 [22] is quite different from the embedding \(\tau_{HP}\) which we considered in this paper, but it is very close in spirit to the embedding \(\tau_{EH}\), which is one of the alternative embeddings considered in [5]. Investigating the relationship between FF-AEL and MKNF, as well as other formalisms which combine Description Logics and Logic Programming (e.g., [9, 25, 15]) is future work. Since positive rules are interpreted as Horn formulas, we conjecture that our semantics corresponds to that of SWRL [15], provided only positive programs are considered, and certain restrictions apply to the allowed concrete domain schemes.

Acknowledgements

The work is funded by the European Commission under the projects ASG, EASAIER, enRaF, Knowledge Web, Musing, Salero, Seemp, SemanticGOV, Super, SWING and TripCom; by Science Foundation Ireland under the DERI-Lion Grant No.SFI/02/CE1/I13; by the FIT-IT (Forschung, Innovation, Tech-
nologie - Informationstechnologie) under the projects Grisino, RW², SemBiz, SeNSE and TSC.

The authors would like to thank all the members of the WSML working group for their advice and input into this document.
Bibliography


