A Denotational Semantics for Standard OIL and Instance OIL

I. Horrocks
Department of Computer Science
University of Manchester, UK
horrocks@cs.man.ac.uk

November 29, 2000

Introduction

In this document we will give a formal specification and semantics for both Standard OIL and Instance OIL languages and associated inferences. This formal specification provides a precise interpreation for all statements in OIL. It serves to make explicit all information that is implicit in OIL statements.

The document provides a model-theoretic specification of the meaning of OIL constructs and details the inference implications of statements. From this document, the reader can determine when an object is a member of a class, when a term is consistent (or inconsistent), and when a term has a subclass (or subrole) relationship to another term.

In this document, we will only consider the **definitions** part of the ontology and we will ignore fields such as **documentation** that have no semantic significance.

OIL semantics

The semantics of OIL rely on a translation into the description logic $\mathcal{SHIQ}$ [4] extended with concrete data types [2]: we will call this logic $\mathcal{SHIQ}(d)$. $\mathcal{SHIQ}(d)$ has a highly expressive concept language that is able to fully capture both the Standard OIL and Instance OIL languages, provided that OIL individuals are treated as "syntactic sugar" for disjoint primitive concepts, and we will define a translation function $\sigma(\cdot)$ that maps OIL ontologies into equivalent $\mathcal{SHIQ}(d)$ terminologies. This has the benefit that reasoning engines for OIL ontologies are (or will soon be) available: an existing $\mathcal{SHIQ}$ reasoner implemented in the FaCT system [3] can be used to reason with OIL ontologies not containing concrete data types, and this will soon be extended to a full $\mathcal{SHIQ}(d)$ reasoner. On the other hand, reasoning with ontologies containing individuals in class definitions seems to problematical, and the design of implementable algorithms is still an open problem.[1]

The translation is quite straightforward and follows directly from the syntax and informal specification of OIL. An OIL ontology $\mathcal{O}$ consists of a list $d_1, \ldots, d_n$, where each d_i is either a class definition, an axiom, a slot definition, an instantiation axiom or a relationship axiom. This list of definitions/axioms is translated into a $\mathcal{SHIQ}(d)$ terminology $\mathcal{T}$ (a set of axioms) as follows:

$$\sigma(d_1, \ldots, d_n) = \bigcup_{i=1}^{n} \sigma(d_i) \cup \mathcal{D}$$

[1] Reasoning with individuals in a so called *Abox* (a set of class and role instantiation axioms) is much more straightforward [5], and work is already underway on an extension of $\mathcal{SHIQ}$ to include Abox reasoning.

where $\mathcal{D}$ is a set of axioms that enforces the disjointness of primitive concepts representing the individuals used in $\mathcal{O}$.

The syntactic correspondence between OIL and $\mathcal{SHIQ}(d)$ is summarised in Figure 1 and described in more detail in the following sections.[2] The translation function $\sigma(\cdot)$ is defined in Figures 2 to 4.

Standard OIL

Class definitions

A class definition is either a pair $\langle \mathsf{CN}, D \rangle$ or a triple $\langle \mathsf{CN}, P, D \rangle$, where CN is a class name, D is a class description and P is either **primitive** or **defined**; $\langle \mathsf{CN}, D \rangle$ is equivalent to $\langle \mathsf{CN}, \textbf{primitive}, D \rangle$. A class definition $\langle \mathsf{CN}, \textbf{primitive}, D \rangle$ is written $\mathsf{CN} \sqsubseteq D$ (it states that CN is a subclass of the class described by D) and a class definition $\langle \mathsf{CN}, \textbf{defined}, D \rangle$ is written $\mathsf{CN} \doteq D$ (it states that CN is equivalent to the class described by D).

A class description D consists of an optional **subclass-of** component, itself a list of one or more **class-expression**s $C_1, \ldots, C_n$, followed by a list of zero or more **slot-constraints** $A_1, \ldots, A_m$. We will write such a class description as

$$[C_1, \ldots, C_n, A_1, \ldots, A_m].$$

Class expressions

A *class-expression* is either a class name CN (some of which have predefined interpretations), an *enumerated-class*, a **slot-constraint**, a conjunction of class expressions, written $C_1 \sqcap \ldots \sqcap C_n$, a disjunction of class expressions, written $C_1 \sqcup \ldots \sqcup C_n$ or a negated class expression, written $\neg C$.

The class names **top**, **thing** and **bottom** have pre-defined interpretations: **top** and **thing** are interpreted as the most general class (written $\top$), while **bottom** is interpreted as the inconsistent class (written $\bot$). Note that **top** and **bottom** can just be considered as abbreviations for the class expressions $(C$ **or** $(\textbf{not}\ C))$ and $(C$ **and** $(\textbf{not}\ C))$ respectively (for some arbitrary class C).

An enumerated-class consists of a list of individual names (introduced by the **one-of** key word) written $i_1, \ldots, i_n$.

Concrete type expressions

Slot constraints can take concrete type expressions as well as class expressions. A concrete type expression can be either:

- One of the predicates **min** d (written $\geqslant_d$), **max** d (written $\leqslant_d$), **greater-than** d (written $>_d$), **less-than** d (written $<_d$), **range** d_1 d_2 (written $(\geqslant_{d_1} \sqcap \leqslant_{d_2})$) and **equal** d (written $(\geqslant_d \sqcap \leqslant_d)$).
- A conjunction of concrete type expressions, written $C_1 \sqcap \ldots \sqcap C_n$, a disjunction of concrete type expressions, written $C_1 \sqcup \ldots \sqcup C_n$ or a negated concrete type expression, written $\neg C$.
- **integer** and **string** are also provided as abbreviations for the expressions ((**min** 0) **or** (**max** 0)) and ((**min** "A") **or** (**max** "A")) respectively.

Individual data values (e.g., the integer 123 or the string "xyz") can also be used as *slot fillers*.

[2] Familiarity with OIL syntax is assumed (see `http://www.ontoknowledge.org/oil/syntax/` for full details).

class-def (**primitive** | **defined**) CN	CN ($\sqsubseteq$ | $\doteq$) $\top$
subclass-of $C_1 \ldots C_n$	$\sqcap \sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n)$
slot-constraint$_1$	$\sqcap \sigma(\text{slot-constraint}_1)$
$\vdots$	$\vdots$
slot-constraint$_m$	$\sqcap \sigma(\text{slot-constraint}_m)$
top | **thing** | **bottom**	$C \sqcup \neg C \mid C \sqcup \neg C \mid C \sqcap \neg C$
(**min** d) | (**max** d)	$\geqslant_d \mid \leqslant_d$
(**greater-than** d) | (**less-than** d)	$>_d \mid <_d$
(**equal** d) | (**range** d_1 d_2)	$(\geqslant_d \sqcap \leqslant_d) \mid (\geqslant_{d_1} \sqcap \leqslant_{d_2})$
(C_1 **and** $\ldots$ **and** C_n)	$(\sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n))$
(C_1 **or** $\ldots$ **or** C_n)	$(\sigma(C_1) \sqcup \ldots \sqcup \sigma(C_n))$
(**not** C)	$(\neg\sigma(C))$
(**one-of** $i_1 \ldots i_n$)	$(P_{i_1} \sqcup \ldots \sqcup P_{i_n})$
slot-constraint *SN*	$\top$
has-value $C_1 \ldots C_n$	$\sqcap \exists SN.\sigma(C_1) \sqcap \ldots \sqcap \exists SN.\sigma(C_n)$
value-type $C_1 \ldots C_n$	$\sqcap \forall SN.\sigma(C_1) \sqcap \ldots \sqcap \forall SN.\sigma(C_n)$
max-cardinality n C	$\sqcap \leqslant n SN.\sigma(C)$
min-cardinality n C	$\sqcap \geqslant n SN.\sigma(C)$
cardinality n C	$\sqcap \geqslant n SN.\sigma(C) \sqcap \leqslant n SN.\sigma(C)$
has-filler d	$\sqcap \exists SN.\sigma(d)$
slot-def *SN*	
subslot-of $SN_1 \ldots SN_n$	$(SN \sqsubseteq SN_1) \ldots (SN \sqsubseteq SN_n)$
domain $C_1 \ldots C_n$	$\exists SN.\top \sqsubseteq \sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n)$
range $C_1 \ldots C_n$	$\top \sqsubseteq \forall SN.\sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n)$
inverse *RN*	$(SN^- \sqsubseteq RN)\ (RN^- \sqsubseteq SN)$
properties transitive	$SN \in \mathbf{S}_+$
properties symmetric	$(SN \sqsubseteq SN^-)\ (SN^- \sqsubseteq SN)$
properties functional	$\top \sqsubseteq \leqslant 1 SN$
disjoint $C_1\ C_2 \ldots C_n$	$(\sigma(C_1) \sqsubseteq \neg\sigma(C_2)) \ldots (\sigma(C_{n-1}) \sqsubseteq \neg\sigma(C_n))$
covered C **by** $C_1 \ldots C_n$	$\sigma(C) \sqsubseteq \sigma(C_1) \sqcup \ldots \sqcup \sigma(C_n)$
disjoint-covered C **by** $C_1 \ldots C_n$	$(\sigma(C_1) \sqsubseteq \neg\sigma(C_2)) \ldots (\sigma(C_{n-1}) \sqsubseteq \neg\sigma(C_n))$ $(\sigma(C) \sqsubseteq \sigma(C_1) \sqcup \ldots \sqcup \sigma(C_n))$
equivalent $C\ C_1 \ldots C_n$	$(\sigma(C) \doteq \sigma(C_1)) \ldots (\sigma(C_{n-1}) \doteq \sigma(C_n))$
instance-of $i\ C_1 \ldots C_n$	$P_i \sqsubseteq \sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n)$
related *SN* $i\ j$	$P_i \sqsubseteq \exists SN.P_j$

Figure 1: Syntactic correspondence between OIL and $\mathcal{SHIQ}(d)$

Slot constraints

A **slot-constraint** consists of a slot name *SN* followed by one or more constraints that apply to the slot, written $SN[a_1, \ldots, a_n]$. Each constraint can be either:

- A **value** constraint with either a list of one or more class-expressions or a list of one or more

concrete type expressions, written $\exists C_1, \ldots, C_n$.

- A **value-type** constraint with either a list of one or more class-expressions or a list of one or more concrete type expressions, written $\forall C_1, \ldots, C_n$.
- A **has-filler** constraint with either a list of one or more individual names or a list of one or more data values, written $\exists C_1, \ldots, C_n$.
- A **max-cardinality** constraint with a non-negative integer n followed (optionally) by either a class expression C or a concrete-type-expression, written $\leqslant n.C$ ($\leqslant n.\top$ if the expression is omitted).
- A **min-cardinality** constraint with a non-negative integer n followed (optionally) by either a class expression or a concrete type expression, written $\geqslant n.C$ ($\geqslant n.\top$ if the expression is omitted).
- A **cardinality** constraint with a non-negative integer n followed (optionally) by either a class expression C or a concrete type expression, written $=n.C$ ($=n.\top$ if the class expression is omitted).

In order to maintain the decidability of the language, cardinality constraints can only be applied to *simple* slots. A simple slot is one that is neither transitive nor has any transitive subslots. However, as the transitivity of a slot can be inferred (e.g., from the fact that the inverse of the slot is a transitive slot), simple slot is defined in terms of the translation into $\mathcal{SHIQ}(d)$: a slot SN in an ontology $\mathcal{O}$ is a simple slot iff $\sigma(\mathsf{SN})$ is a simple role in the $\mathcal{SHIQ}(d)$ terminology $\sigma(\mathcal{O})$.

Axioms

Standard OIL includes four kinds of axiom:

disjoint a list of two or more class expressions that are pairwise disjoint, written $\|C_1, \ldots, C_n\|$.

covered a single class expression that is covered by the succeeding list of class expressions, written $C \sqsubseteq (C_1, \ldots, C_n)$.

disjoint-covered a single class expression that is covered by the succeeding list of disjoint class expressions, written $C \sqsubseteq \|C_1, \ldots, C_n\|$.

equivalent a list of two or more class expressions that are equivalent, written $\doteq(C_1, \ldots, C_n)$.

The mapping function $\sigma(\cdot)$

We can now define how the function $\sigma(\cdot)$ maps OIL axioms and class definitions into (sets of) $\mathcal{SHIQ}(d)$ axioms. The definition is given in Figures 2 and 3, where CN is a class name (or a $\mathcal{SHIQ}(d)$ concept name), SN is a slot name (or $\mathcal{SHIQ}(d)$ role name), C (possibly subscripted) is a class expression, D (possibly subscripted) is a class or concrete type expression, E is a class expression or a class description (super-classes plus slot constraints), A (possibly subscripted) is a slot constraint, a_i is a constraint (on a slot), i is an OIL individual, P_i is the $\mathcal{SHIQ}(d)$ primitive concept used to represent the OIL individual i, d is a concrete data value (an integer or a string), n is a non-negative integer and p_d is a unary predicate (i.e., $p \in \{\geqslant, \leqslant, >, <\}$ and d is a concrete data value).

$$
\begin{array}{rcl}
\sigma(C \sqsubseteq E) & = & \{\sigma(C) \sqsubseteq \sigma(E)\} \\
\sigma(C \doteq E) & = & \{\sigma(C) \sqsubseteq \sigma(E), \sigma(E) \sqsubseteq \sigma(C)\} \\
\sigma(\|C_1, \ldots, C_n\|) & = & \bigcup_{i=1}^{n-1} \bigcup_{j=i+1}^{n} \{\sigma(C_i) \sqsubseteq \neg\sigma(C_j)\} \\
\sigma(C \sqsubseteq (C_1, \ldots, C_n)) & = & \{\sigma(C) \sqsubseteq \sigma(C_1) \sqcup \ldots \sqcup \sigma(C_n)\} \\
\sigma(C \sqsubseteq \|C_1, \ldots, C_n\|) & = & \sigma(C \sqsubseteq (C_1, \ldots, C_n)) \cup \sigma(\|C_1, \ldots, C_n\|) \\
\sigma(\doteq(C_1, \ldots, C_n)) & = & \bigcup_{i=1}^{n-1} \sigma(C_i \doteq C_{i+1})
\end{array}
$$

Figure 2: Translation of OIL axioms into $\mathcal{SHIQ}(d)$

$$
\begin{array}{rcl}
\sigma([C_1, \ldots, C_n, A_1, \ldots, A_m]) & = & \top \sqcap \sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n) \sqcap \sigma(A_1) \sqcap \ldots \sqcap \sigma(A_m) \\
\sigma(\mathsf{CN}) & = & \mathsf{CN} \\
\sigma(D_1 \sqcap \ldots \sqcap D_n) & = & \sigma(D_1) \sqcap \ldots \sqcap \sigma(D_n) \\
\sigma(D_1 \sqcup \ldots \sqcup D_n) & = & \sigma(D_1) \sqcup \ldots \sqcup \sigma(D_n) \\
\sigma(\neg D) & = & \neg\sigma(D) \\
\sigma(\mathsf{SN}[a_1, \ldots, a_n]) & = & \sigma(\mathsf{SN}(a_1)) \sqcap \ldots \sqcap \sigma(\mathsf{SN}(a_n)) \\
\sigma(\mathsf{SN}(\exists D_1, \ldots, D_n)) & = & \exists \mathsf{SN}.\sigma(D_1) \sqcap \ldots \sqcap \exists \mathsf{SN}.\sigma(D_n) \\
\sigma(\mathsf{SN}(\forall D_1, \ldots, D_n)) & = & \forall \mathsf{SN}.\sigma(D_1) \sqcap \ldots \sqcap \forall \mathsf{SN}.\sigma(D_n) \\
\sigma(\mathsf{SN}(\leqslant n.D)) & = & \leqslant n \mathsf{SN}.\sigma(D) \\
\sigma(\mathsf{SN}(\geqslant n.D)) & = & \geqslant n \mathsf{SN}.\sigma(D) \\
\sigma(\mathsf{SN}(= n.D)) & = & \leqslant n \mathsf{SN}.\sigma(D) \sqcap \geqslant n \mathsf{SN}.\sigma(D) \\
\sigma(i) & = & P_i \\
\sigma(i_1, \ldots, i_n) & = & P_{i_1} \sqcup \ldots \sqcup P_{i_n} \\
\sigma(d) & = & (\geqslant_d \sqcap \leqslant_d) \\
\sigma(p_d) & = & p_d
\end{array}
$$

Figure 3: Translation of OIL class definitions into $\mathcal{SHIQ}(d)$

$$
\begin{array}{rcl}
\sigma(\mathsf{SN}[\mathsf{RN}_1,\ldots,\mathsf{RN}_n,S_1,\ldots,S_m]) & = & \sigma(\mathsf{SN}[\mathsf{RN}_1,\ldots,\mathsf{RN}_n]) \cup \sigma(\mathsf{SN}[S_1,\ldots,S_m]) \\
\sigma(\mathsf{SN}[\mathsf{RN}_1,\ldots,\mathsf{RN}_n]) & = & \bigcup_{i=1,\ldots,n} \sigma(\mathsf{SN} \sqsubseteq \mathsf{RN}_i) \\
\sigma(\mathsf{SN}[S_1,\ldots,S_m]) & = & \bigcup_{i=1,\ldots,m} \sigma(\mathsf{SN}(S_i)) \\
\sigma(\mathsf{SN} \sqsubseteq \mathsf{RN}) & = & \{\mathsf{SN} \sqsubseteq \mathsf{RN}\} \\
\sigma(\mathsf{SN}(\downarrow [C_1,\ldots,C_n])) & = & \{\exists \mathsf{SN}.\top \sqsubseteq \sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n)\} \\
\sigma(\mathsf{SN}(\uparrow [C_1,\ldots,C_n])) & = & \{\top \sqsubseteq \forall \mathsf{SN}.\sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n)\} \\
\sigma(\mathsf{SN}(^{-}\mathsf{RN})) & = & \{\mathsf{SN}^{-} \sqsubseteq \mathsf{RN}, \mathsf{RN} \sqsubseteq \mathsf{SN}^{-}\} \\
\sigma(\mathsf{SN}([P_1,\ldots,P_n])) & = & \bigcup_{i=1,\ldots,n} \{\sigma(\mathsf{SN}(P_i))\} \\
\sigma(\mathsf{SN}(+)) & = & \{\mathsf{SN} \in \mathbf{S}_+\} \\
\sigma(\mathsf{SN}(\leftrightarrow)) & = & \{\mathsf{SN}^{-} \sqsubseteq \mathsf{SN}, \mathsf{SN} \sqsubseteq \mathsf{SN}^{-}\} \\
\sigma(\mathsf{SN}(\uparrow)) & = & \{\top \sqsubseteq \leqslant 1 \mathsf{SN}\}
\end{array}
$$

Figure 4: Translation of OIL slot definitions into $\mathcal{SHIQ}$

In addition, the set of disjointness axioms $\mathcal{D}$ is defined as:

$$\bigcup_{j=1}^{n-1} \bigcup_{k=j+1}^{n} \{P_j \sqsubseteq \neg P_k\}$$

where $i_i, \ldots, i_n$ are the individuals used in $\mathcal{O}$ and P_i is the $\mathcal{SHIQ}(d)$ primitive concept used to represent i.

Slot definitions

A slot definition is a pair $\langle \mathsf{SN}, X \rangle$, where SN is a slot name and X is a slot description. A slot description X consists of an optional **subslot-of** component, itself a list of one or more slot names $\mathsf{RN}_1, \ldots, \mathsf{RN}_n$, followed by a list of zero or more global slot constraints (e.g., **inverse**) $S_1, \ldots, S_m$. We will write such a slot definition as:

$$\mathsf{SN}[\mathsf{RN}_1, \ldots, \mathsf{RN}_n, S_1, \ldots, S_m]$$

Each global constraint S_i on SN can be either:

- A **domain** constraint with a list of one or more class-expressions, written $\downarrow [C_1, \ldots, C_n]$.
- A **range** constraint with a list of one or more class-expressions, written $\uparrow [C_1, \ldots, C_n]$.
- An **inverse** constraint with a slot name RN, written $^{-}\mathsf{RN}$.
- A **properties** constraint with a list of one or more properties, written $[P_1, \ldots, P_n]$. Valid properties are **transitive**, written $+$, **symmetrical**, written $\leftrightarrow$ and **functional**, written $\uparrow$.

We can now define how the function $\sigma(\cdot)$ maps an OIL slot definition into a set of $\mathcal{SHIQ}$ axioms. The definition is given in Figure 4, where RN and SN are slot names (or $\mathcal{SHIQ}$ role names), C_i is a class expression, S_i is a global slot constraint and P_i is a property.

$$
\begin{array}{rcll}
(R^-)^{\mathcal{I}} & = & \{\langle x,y\rangle \mid \langle y,x\rangle \in R^{\mathcal{I}}\} & \text{(inverse roles)} \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} & \text{(conjunction)} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} & \text{(disjunction)} \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} & \text{(negation)} \\
(\exists R.C)^{\mathcal{I}} & = & \{x \mid \exists y.\langle x,y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} & \text{(value constraint)} \\
(\forall R.C)^{\mathcal{I}} & = & \{x \mid \forall y.\langle x,y\rangle \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\} & \text{(value-type constraint)} \\
(\exists R.p)^{\mathcal{I}} & = & \{x \mid \exists y.\langle x,y\rangle \in R^{\mathcal{I}} \text{ and } y \in \Delta^{\mathcal{D}} \text{ and } p(y)\} & \text{(concrete value)} \\
(\forall R.p)^{\mathcal{I}} & = & \{x \mid \forall y.\langle x,y\rangle \in R^{\mathcal{I}} \text{ implies } y \in \Delta^{\mathcal{D}} \text{ and } p(y)\} & \text{(concrete value-type)} \\
(\geqslant nR.C)^{\mathcal{I}} & = & \{x \mid \sharp\{y.\langle x,y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \geqslant n\} & \text{(minimum cardinality)} \\
(\leqslant nR.C)^{\mathcal{I}} & = & \{x \mid \sharp\{y.\langle x,y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \leqslant n\} & \text{(maximum cardinality)} \\
(\geqslant nR.p)^{\mathcal{I}} & = & \{x \mid \sharp\{y.\langle x,y\rangle \in R^{\mathcal{I}} \text{ and } y \in \Delta^{\mathcal{D}} \text{ and } p(y)\} \geqslant n\} & \text{(concrete min. card.)} \\
(\leqslant nR.p)^{\mathcal{I}} & = & \{x \mid \sharp\{y.\langle x,y\rangle \in R^{\mathcal{I}} \text{ and } y \in \Delta^{\mathcal{D}} \text{ and } p(y)\} \leqslant n\} & \text{(concrete max. card.)} \\
(\leqslant 1R)^{\mathcal{I}} & = & \{x \mid \sharp\{y.\langle x,y\rangle \in R^{\mathcal{I}} \text{ and } y \in (\Delta^{\mathcal{D}} \cup \Delta^{\mathcal{I}})\} \leqslant 1\} & \text{(functional restriction)}
\end{array}
$$

Figure 5: $\mathcal{SHIQ}(d)$ semantics

Instance OIL

Instance OIL extends Standard OIL with the possibility to define instances of classes and roles. These definitions are of the form (**instance-of** i $C_1, \ldots, C_n$) and (**related** S i j), where C (possibly subscripted) is a class expression, S is a slot name and i and j are individual names. An axiom (**instance-of** i $C_1, \ldots, C_n$) is equivalent to (**instance-of** i (C_1 **and** $\ldots$ **and** C_n). These axioms are simply treated as "syntactic sugar" for axioms of the form

covered-by (**one-of** i) C

(i.e., $P_i \sqsubseteq C$), and

covered-by (**one-of** i) (**slot-constraint** S **has-value** (**one-of** j))

(i.e., $P_i \sqsubseteq \exists S.P_j$) respectively.

The semantics of $\mathcal{SHIQ}(d)$

The meaning of a $\mathcal{SHIQ}(d)$ terminology, and of the common inference problems, is given in terms of a Tarski style model theoretic semantics using *interpretations* [2, 6, 4]. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \Delta^{\mathcal{D}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$, called the *abstract domain* of $\mathcal{I}$, a set $\Delta^{\mathcal{D}}$, called the *concrete domain* of $\mathcal{I}$, and a *valuation function* $\cdot^{\mathcal{I}}$. The abstract and concrete domains must be disjoint, i.e., $\Delta^{\mathcal{I}} \cap \Delta^{\mathcal{D}} = \emptyset$.

The valuation function $\cdot^{\mathcal{I}}$ maps every concept to a subset of $\Delta^{\mathcal{I}}$ and every role to a subset of $\Delta^{\mathcal{I}} \times (\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{D}})$ such that, for all concepts C, D, roles R, S, concrete predicate expressions p and non-negative integers n, the equations in Figure 5 are satisfied (where $\sharp M$ denotes the cardinality of a set M). Concrete predicate expressions have the obvious interpretation as shown in Figure 6. The ordering on strings is the standard lexicographic one.

The concrete domain is treated differently from the abstract domain because it is considered to be already sufficiently structured by the predicates **min**, **max** etc. Therefore, it is not appropriate to form new classes of concrete objects (values) using the concept language [1].

$$
\begin{array}{rcl}
(p_1 \sqcap \ldots \sqcap p_n)(y) & \leftrightarrow & p_1(y) \wedge \ldots \wedge p_n(y) \\
(p_1 \sqcup \ldots \sqcup p_n)(y) & \leftrightarrow & p_1(y) \vee \ldots \vee p_n(y) \\
\neg(p_1 \sqcap \ldots \sqcap p_n)(y) & \leftrightarrow & \neg p_1(y) \vee \ldots \vee \neg p_n(y) \\
\neg(p_1 \sqcup \ldots \sqcup p_n)(y) & \leftrightarrow & \neg p_1(y) \wedge \ldots \wedge \neg p_n(y) \\
\neg \geqslant_x(y) & \leftrightarrow & <_x(y) \\
\neg \leqslant_x(y) & \leftrightarrow & >_x(y) \\
\neg >_x(y) & \leftrightarrow & \leqslant_x(y) \\
\neg <_x(y) & \leftrightarrow & \geqslant_x(y)
\end{array}
$$

Figure 6: Concrete expression semantics

In order to avoid considering roles such as R^{--} (i.e., the inverse of an inverse) we will define a function Inv such that $Inv(R)$ is R^- and $Inv(R^-)$ is R. A role R is *directly subsumed* by a role S w.r.t. a terminology $\mathcal{T}$ iff either $\{R \sqsubseteq S\} \subseteq \mathcal{T}$ or $\{Inv(R) \sqsubseteq Inv(S)\} \subseteq \mathcal{T}$. A role R is *subsumed* by a role S w.r.t. $\mathcal{T}$ (written $\mathcal{T} \models R \sqsubseteq S$) iff R is directly subsumed by a S or there is a role S' such that R is directly subsumed by a S' and $\mathcal{T} \models S' \sqsubseteq S$. A role R is *equivalent* to a role S w.r.t. $\mathcal{T}$ (written $\mathcal{T} \models R \doteq S$) iff $\mathcal{T} \models R \sqsubseteq S$ and $\mathcal{T} \models S \sqsubseteq R$. A role R is transitive in $\mathcal{T}$ iff $\{S \in \mathbf{S}_+\} \subseteq \mathcal{T}$ for some role S such that $R \doteq S$ or $Inv(R) \doteq S$ (this defines $\mathbf{S}_+$, the set of transitive role names). A role R is a *simple* role in $\mathcal{T}$ iff there is no role S such that S is transitive in $\mathcal{T}$ and $\mathcal{T} \models S \sqsubseteq R$.

An interpretation $\mathcal{I}$ *satisfies* a $\mathcal{SHIQ}$ terminology $\mathcal{T}$ iff for every axiom $R \sqsubseteq S$ in $\mathcal{T}$, $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$, for every axiom $C \sqsubseteq D$ in $\mathcal{T}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and for every transitive role S in $\mathcal{T}$, $S^{\mathcal{I}} = (S^{\mathcal{I}})^+$. Such an interpretation is called a *model* of $\mathcal{T}$ (written $\mathcal{I} \models \mathcal{T}$).

A concept C is satisfiable with respect to a $\mathcal{SHIQ}$ terminology $\mathcal{T}$ (written $\mathcal{T} \models C \neq \bot$) iff there is a model $\mathcal{I}$ of $\mathcal{T}$ with $C^{\mathcal{I}} \neq \emptyset$. A concept C is subsumed by a concept D w.r.t. $\mathcal{T}$ (written $\mathcal{T} \models C \sqsubseteq D$) $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for each model $\mathcal{I}$ of $\mathcal{T}$.

An OIL ontology $\mathcal{O}$ is called *consistent* iff $\sigma(\mathcal{O}) \models \top \neq \bot$. A class CN in an ontology $\mathcal{O}$ is called *consistent* iff $\sigma(\mathcal{O}) \models \sigma(\mathsf{CN}) \neq \bot$. A class CN is a *subclass* of a class DN in an ontology $\mathcal{O}$ iff $\sigma(\mathcal{O}) \models \sigma(\mathsf{CN}) \sqsubseteq \sigma(\mathsf{DN})$.

References

[1] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 452–457, 1991.

[2] F. Baader, H.-J. Heinsohn, B. Hollunder, J. Muller, B. Nebel, W. Nutt, and H.-J. Profitlich. Terminological knowledge representation: A proposal for a terminological logic. Technical Memo TM-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (DFKI), 1991.

[3] I. Horrocks. Benchmark analysis with fact. In *Proc. of TABLEAUX 2000*, number 1847 in Lecture Notes in Artificial Intelligence, pages 62–66. Springer-Verlag, 2000.

[4] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.

[5] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In David MacAllester, editor, *Proc. of CADE-2000*, number 1831 in Lecture Notes in Artificial Intelligence, pages 482–496. Springer-Verlag, 2000.

[6] P. F. Patel-Schneider and B. Swartout. Description logic specification from the KRSS effort, June 1993.