A higher-order semantics for OWL 2 QL ontologies

Maurizio Lenzerini, Lorenzo Lepore, Antonella Poggi

Dipartimento di Ingegneria Informatica, Automatica e Gestionale "Antonio Ruberti"- Sapienza Università di Roma
Via Ariosto 25, I-00183 Roma, Italy
lastname@dis.uniroma1.it

Recent OBDA projects have pointed out that one of the drawbacks of OWL 2 is the lack of metamodeling and metaquerying capabilities, i.e., features for specifying and reasoning about metaconcepts and metaproperties [1]. Roughly speaking, a metaconcept is a concept whose instances can be themselves concepts, and a metaproperty is a relationship between metaconcepts. Although OWL 2 provides syntactic support for metamodeling through punning (by which the same name can be used to denote ontology elements of different categories, such as a class and an individual), we argue that the official semantics of OWL 2, the so-called *Direct Semantics* (DS), treats punning in a way that is not adequate for representing metaconcepts and metaproperties. The reason is simply that proper metamodeling requires that the *same* element plays the role of individual and class (or, class and relation), while DS sanctions that an individual and a class with the same name are different elements. This is confirmed by the fact that the *Direct Semantics Entailment Regime* (DSER), which is the logic-based semantics of the SPARQL 1.1 query language when applied to OWL 2 QL, forces queries to obey the so-called *typing constraint*, which rules out the possibility of using the same variable in incompatible positions (for example, in individual and in class position).

The issue of metamodeling in OWL has been investigated in several papers. It is known that the semantics of metamodeling of OWL 2 Full leads to undecidability of basic inference problems [7]. A possible solution to this problem is to enable metamodeling in OWL 2 DL by axiomatizing the higher order knowledge into first order assertions [4], but the process involves the use of complex expressions that are not supported by OWL 2 DL tractable profiles, and therefore seems inapplicable in OBDA. Another possible solution is the stratification of class constructors and axioms to describe metalevels of classes and properties [8], but such stratification poses challenges for the modeler, and rules out interesting ontology patterns. Relevant to our work are recent papers aiming at devising efficient techniques to answer SPARQL 1.1 queries posed to OWL 2 QL ontologies [2, 6, 5]. However, such papers concentrate on DSER, and therefore do not aim at the full power of metamodeling and metaquerying.

The goal of our work is to present a new higher-order semantics for OWL 2 QL, called *HOS* and inspired by [3], allowing us to effectively exploit the capabilities for metamodeling offered by punning, and to show that, based on such semantics, it is possible to define a new SPARQL entailment regime, called *HOSER*, for the class of (meta)queries obtained by relaxing the typing constraint of DSER.

Illustrating scenario. We refer to an `OWL 2 QL` ontology (see an excerpt in Table 1), whose central entity is `:financial_instrument`.Metamodeling comes into play in order to capture the notion of types of financial instruments, modeled by using the metaclass `:type_of_f_i`, whose instances are other classes which are intended to be subclasses of `:financial_instrument` (for example, `:BTP` and `:commercial_paper`, see axioms (3),(4),(6)), and by defining its properties. One notable example of such properties is `:established_by` (see axioms (9),(10),(11)), which associates to each type of financial instrument the law that formally established it (for example, the law `:DR135bis` established `:BTP` as a type of financial instrument, see axiom (5)).Observe that syntactically we are using punning. However, as we said before, the direct semantics of `OWL 2` considers the individual named `:BTP` different from the class with the same name, and therefore is inadequate for metamodeling. This is evident if one considers the query

select $\$x\,\$y$ **where** $\{$`:IT0005069395 rdf:type` $\$x\,.\,\$x$ `:established_by` $\$y\}$

asking for the law that established the type of financial instrument having a specific financial instrument (for example `:IT0005069395`) as instance. This query is not legal under DSER, because the variable $x appears both in individual and in class position. Note that if typing constraint is lifted, thus making the query legal, then it would become empty under DSER is trivially empty since DS cannot interpret any ontology element as both an individual and a class.

`ClassAssertion(:BTP :IT0005069395)`	(1)
`ClassAssertion(:commercial_paper :ZAG000117292)`	(2)
`ClassAssertion(:type_of_Italian_f_i :BTP)`	(3)
`ClassAssertion(:type_of_foreign_f_i :commercial_paper)`	(4)
`ObjectPropertyAssertion(:established_by :BTP :DR135bis)`	(5)
`SubClassOf(:BTP :financial_instrument)`	(6)
`SubClassOf(:commercial_paper :financial_instrument)`	(7)
`SubClassOf(:financial_instrument DataSomeValuesFrom(:duration))`	(8)
`SubClassOf(:type_of_f_i ObjectSomeValuesFrom(:established_by :law))`	(9)
`SubClassOf(:type_of_Italian_f_i ObjectSomeValuesFrom(:established_by :Italian_law))`	(10)
`SubClassOf(ObjectSomeValuesFrom(ObjectInverseOf(:established_by)) :type_of_f_i)`	(11)
`SubClassOf(:type_of_Italian_f_i :type_of_f_i)`	(12)
`SubClassOf(:type_of_foreign_f_i :type_of_f_i)`	(13)

Table 1: The ontology

Higher-Order Semantics for `OWL 2 QL`. A *vocabulary* V for an ontology is constituted by a tuple $V = (V_N, V_C, V_{OP}, V_{DP}, V_{DT}, \mathtt{L})$, where V_N is a set of IRIs that includes the reserved vocabulary $\mathtt{R}$ of `OWL 2 QL`, $V_C, V_{OP}, V_{DP}, V_{DT}$ are subsets of V_N, and $\mathtt{L}$ is the set of `OWL 2 QL` literals. The symbols in V_N are called *entity names*, since they are used to name all the entities of the ontology, while V_C (resp., V_{OP}, V_{DP}, V_{DT}) is the subset of V_N used to name those entities playing the role of class (resp., object properties, data properties, datatypes). Any symbol in $V_N \setminus \mathtt{R}$ may denote an entity that simultaneously plays the role of (*i*) an individual, (*ii*) either a class or a datatype, and (*iii*) either an object property or a data property. We denote by V_I the set $V_N \setminus (V_C \cup V_{OP} \cup V_{DP} \cup V_{DT})$, i.e., the set of entity names that can only play the role of individuals in the ontology. The symbols in V_N, also called *atomic expressions*, constitute the building blocks for denoting the entities of an ontology. Entities, however, can be denoted by using general *expressions*. The set Exp_V of expressions over V is the set $V_I \cup Exp_V^{OP} \cup Exp_V^{C} \cup V_{DP} \cup V_{DT}$, where $Exp_V^{OP} = V_{OP} \cup \{\mathtt{ObjectInverseOf}(e_1) \mid e_1 \in V_{OP}\}$, and $Exp_V^{C} = V_C \cup \{\mathtt{ObjectSomeValuesFrom}(e_1 e_2) \mid e_1 \in Exp_V^{OP}, e_2 \in V_C\} \cup \{\mathtt{DataSomeValuesFrom}(e_1 e_2) \mid e_1 \in V_{DP}, e_2 \in V_{DT}\}$. Expressions over V are then used in the axioms forming any ontology defined over the vocabulary V. HOS is based on the notion of interpretation structure, which plays the role of "interpretation domain" in classical logic. Given a vocabulary $V = (V_N, V_C, V_{OP}, V_{DP}, V_{DT})$, an *interpretation structure* for V is a tuple $\Sigma = (\Delta, \cdot^I, \cdot^E, \cdot^R, \cdot^A, \cdot^T)$

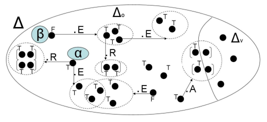

Fig. 1: Interpretation structure

Axiom a	$\mathcal{I} \models a$ **if**
`SubClassOf`$(e_1\, e_2)$	$\cdot^E$ is defined for both $e_1^{\mathcal{I}_o}$ and $e_2^{\mathcal{I}_o}$ and $(e_1^{\mathcal{I}_o})^E \subseteq (e_2^{\mathcal{I}_o})^E$
`ObjectPropertyDomain`$(e_1\, e_2)$	$\cdot^R$ is defined for $e_1^{\mathcal{I}_o}$, $\cdot^E$ is defined for $e_2^{\mathcal{I}_o}$ and $\forall \langle d, d' \rangle \in (e_1^{\mathcal{I}_o})^R : d \in (e_2^{\mathcal{I}_o})^E$
`ClassAssertion`$(e_1\, e_2)$	$\cdot^E$ is defined for $e_1^{\mathcal{I}_o}$, $(e_2^{\mathcal{I}_o})^I = \mathtt{T}$ and $e_2^{\mathcal{I}_o} \in (e_1^{\mathcal{I}_o})^E$
`ObjectPropertyAssertion`$(e_1\, e_2\, e_3)$	$\cdot^R$ is defined for $e_1^{\mathcal{I}_o}$, $(e_2^{\mathcal{I}_o})^I = (e_3^{\mathcal{I}_o})^I = \mathtt{T}$ and $\langle e_2^{\mathcal{I}_o}, e_3^{\mathcal{I}_o} \rangle \in (e_1^{\mathcal{I}_o})^R$
`DataPropertyAssertion`$(e_1\, e_2\, v)$	$\cdot^A$ is defined for $e_1^{\mathcal{I}_o}$, $(e_2^{\mathcal{I}_o})^I = \mathtt{T}$ and $\langle e_2^{\mathcal{I}_o}, v^{\mathcal{I}_o} \rangle \in (e_1^{\mathcal{I}_o})^A$

Table 2: Satisfaction of `OWL 2 QL` axioms

where: (i) Δ is the disjoint union of two sets: Δ_o, the *object domain*, and Δ_v, the *value domain*; and (ii) $\cdot^E : \Delta_o \to \mathcal{P}(\Delta_o)$, $\cdot^R : \Delta_o \to \mathcal{P}(\Delta_o \times \Delta_o)$, $\cdot^A : \Delta_o \to \mathcal{P}(\Delta_o \times \Delta_v)$, and $\cdot^T : \Delta_o \to \mathcal{P}(\Delta_v)$ are partial functions, and $\cdot^I : \Delta_o \to \{\mathtt{T}, \mathtt{F}\}$ is a total function such that for each $d \in \Delta_o$, if $\cdot^E, \cdot^R, \cdot^A, \cdot^T$ are all undefined for d, then $d^I = \mathtt{T}$. Thus, the interpretation structure is a mathematical representation of a world made up by domain elements (the members of the set Δ) which can be either objects or values. Objects are polymorphic, in the sense that each of them can simultaneously be an individual(this is the case where function $\cdot^I$ is `T`), a set (this is the case where the partial function $\cdot^E$ is defined), a binary relation ($\cdot^R$ is defined), an attribute ($\cdot^A$ is defined), and a value type ($\cdot^T$ is defined). Also, an object that is a set (resp., relation, attribute, value type) is associated to its extension through $\cdot^E$ (resp., $\cdot^R$, $\cdot^A$, $\cdot^T$). Figure 1 shows a representation of an interpretation structure, where α is an individual, a set and a relation, whereas β is a set, but not an individual or a relation. An *interpretation* $\mathcal{I}$ for V is a pair, $\langle \Sigma, \mathcal{I}_o \rangle$, where Σ is an interpretation structure for V, and $\mathcal{I}_o$ is the *interpretation function* for $\mathcal{I}$, i.e., a function that maps every expression in Exp_V into an object in Δ_o, and every literal in `L` into a value in Δ_v, according to a set of suitable conditions. (e.g., $((\mathtt{ObjectInverseOf}(e_1))^{\mathcal{I}_o})^R = ((e_1^{\mathcal{I}_o})^R)^{-1}$). To define the semantics of axioms, we refer to the notion of satisfaction of an axiom with respect to an interpretation $\mathcal{I}$. Some of the rules defining such notion are specified in Table 2 (where e, e_1, e_2, e_3 and v are expressions).

`SPARQL` higher-order semantics entailment regime Defining a `SPARQL` entailment regime requires to specify (α) which is the semantics used to interpret the queried ontology, (β) which queries are legal, and (γ) a definition for the notion of answer to queries (called solution in `SPARQL` jargon). As for (α), we adopt HOS. As for (β), we extend the class of queries which are legal for DSER, by relaxing the typing constraint. As for (γ), we propose to interpret as pure existentials, variables that occur in the body of the query but not in the target, thus following the classical notion of existential variables in logic. Observe that the query mentioned in the scenario is legal for HOSER and that by evaluating it over the scenario ontology, it would return the answer $\langle$`:BTP`, `:DR135bis`$\rangle$. One of the results of our work is an algorithm, based on "blind metagrounding", showing that query answering over `OWL 2 QL` ontologies under HOSER has the same data complexity as under DSER. Nevertheless, such algorithm can be inefficient in practice. Hence, we devised a new algorithm that is polynomial with respect to extensional assertions and more efficient for a huge class of `OWL 2 QL` ontologies (so called *ISA-closed*) that are very common in practice.

References

1. D. Allemang and J. Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Elsevier, 2011.
2. M. Arenas, G. Gottlob, and A. Pieris. Expressive languages for querying the semantic web. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 14–26, 2014.
3. G. De Giacomo, M. Lenzerini, and R. Rosati. Higher-order description logics for domain metamodeling. In *Proc. of AAAI 2011*, 2011.
4. B. Glimm, S. Rudolph, and J. Völker. Integrated metamodeling and diagnosis in OWL 2. In *Proc. of ISWC 2010*, volume 6496 of *LNCS*, pages 257–272. Springer, 2010.
5. I. Kollia and B. Glimm. Optimizing SPARQL query answering over OWL ontologies. *J. Artif. Intell. Res. (JAIR)*, 48:253–303, 2013.
6. R. Kontchakov, M. Rezk, M. Rodriguez-Muro, G. Xiao, and M. Zakharyaschev. Answering SPARQL queries over databases under OWL 2 QL entailment regime. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, pages 552–567, 2014.
7. B. Motik. On the properties of metamodeling in OWL. *J. of Logic and Computation*, 17(4):617–637, 2007.
8. J. Z. Pan, I. Horrocks, and G. Schreiber. OWL FA: A metamodeling extension of OWL DL. In *Proceedings of the OWLED*05 Workshop on OWL: Experiences and Directions, Galway, Ireland, November 11-12, 2005*, 2005.