Intensional Semantics for RDF Data Structures

Zoran Majkić,
ETF, Applied Mathematics Department, University of Belgrade, Serbia
e-mail: majkic@etf.bg.ac.yu

Abstract—The Resource Description Framework Schema Specification (RDFS) is the foundation of the Semantic Web and an ontology representation language for database structures. Reification and anonymous resources are two of the more interesting features of RDFS, which is an important and advanced step toward natural language specification of data structures. We propose to valorize these original features of this language by embedding it into a formal intensional FOL (First-order Logic) language with abstraction operation which transforms logic formulae into intensional abstracts terms. We show how this embedding is a natural and simple extension of intrinsic intensional semantics of RDFS into intensional FOL language, which improves the RDFS features. Then we show how we can define the view-based mappings for data integration between RDFS database ontologies, based on an introduction of a new kind of the intensional equivalence for intensional abstracts.

I. INTRODUCTION

The Semantic Web [1] and ontologies play very important role in the recent explosion of interest in the World Wide Web. The enormous amount of data has made difficult to find, access, present, and maintain the information required by users. This is because information content is presented primarily in natural language, and needs the formal language specification with natural language features, as, for example, reification. The RDF Schema Specification provides a machine-understandable formalism for defining schemas for descriptive vocabularies like the Dublin Core. It allows designers to specify classes of resource types and properties to convey descriptions of those classes, relationships between those properties and classes, and constraints on the allowed combinations of classes, properties, and values.

Basicall, RDF defines a data model for describing machine-understandable information on the Web. The basic data model consists of three object types: Resources, Properties and Statements. The modeling primitives of RDF are very basic: actually they correspond to binary predicates (RDF-properties) of ground terms (source and value), where, however the predicates may be used as terms so that RDF can not be embedded into the standard extensional first order logic (FOL). RDF has blank-nodes which are essentially globally-quantified existential variables. The statements are also resources, so that statements can be applied recursively to statements, allowing their nesting (reification) but such most innovative feature (early versions of RDF made quite strong claims for RDF reification) is not actually supported by RDF; such problem will be explored in more details in the following (one of the main reasons to introduce the intensional FOL [2]).

Therefore, the RDF Schema (RDFS) specification [3] enriches RDF by giving an externally specified semantics to specific resources, e.g., to rdf:subClassOf, to rdf:Class, etc.. It is only because of this external semantics that RDFS is useful. RDFS is recognizable as an ontology representation language: it talks about classes and RDF-properties (binary relations), range and domain constraints (on RDF-properties), and subclass and subproperty (subsumption) relations.

Relevant work:

RDFS has a very interesting extension to natural language (e.g., reification), but, at least in some logical aspects is a weakly expressive language, and such limitations of RDFS led to the development of new Web ontology languages such as, for example, DAML+OIL adopted by W3C as the basis of a new OWL Web Ontology Language [4].

So, all attempts to integrate RDFS into some more expressive FOL sublanguage with a built-in extensional equality theory (as OWL, largely based on Description Logic, or other interesting languages as Logic Programming, Deductive databases, or Modal Logic languages, etc..) are unsuccessful [5], [6]: these FOL sublanguages can not embed the reification, which is fundamental natural-language property and is more innovative feature of the RDF w.r.t. other ontology specification languages based on the FOL sublanguages. Thus the motivation for this work is just to valorize this very important feature of RDF, for building intelligent database systems.

The difficulty comes from the fact that all FOL sublanguages have the model theory in which individuals are interpreted as elements of some domain, classes are interpreted as subsets of the domain, and RDF-properties are interpreted as binary relations on the domain; the semantics of RDFS, on the other hand, are given by a non-standard model theory, where individuals, classes and RDF-properties are all elements in the domain, RDF-property elements have extension which are binary relations on the domain, and class extensions are only implicitly defined by the rdf:type property.

Such properties can be directly embedded into Higher-order logics (HOL) with a built-in extensional equality theory (under standard HOL semantics, an equation between predicate (or function) symbols is true if and only if these symbols are interpreted via the same relation (extension)), in particular the use of predicates in variable position. Unfortunately, extensional equality of predicates and functions is not decidable, which carries over to the unification problem. For genuine second-order theories (e.g., second-order predicate calculus and Church’s simple theory of types, both under the standard semantics) extensional equality is not even semi-decidable. Moreover, RDFS support reflection on its own syntax: it is defined in terms of classes and RDF-properties which are interpreted in the same way as other classes and RDF-properties, and whose meaning can be extended by statements
in the language. This violates the very principles of set theory, i.e., that set membership should be a well-defined relationship, so when we try to integrate RDFS into some (extensional) FOL sublanguage we meet Russel paradox, result from an attempt to make sets too powerful.

For example, it is not possible in RDFS to provide defined classes - classes that give a formula that determines which resources belong to them. We can provide an even richer theory of classes and RDF-properties, allowing for defined classes and more relationships between classes.

The RDF syntax based on triples (RDF sentences) is a graph data structure, reminiscent of semantic networks. Such graph structure have well-known problems with scoping of quantifiers, negations, and disjunctions (consequently, with logic implication). RDF solves these problems by disallowing all these constructs (anonymous nodes in RDF can be considered to be existentially quantified, but the scope of the quantifier is the entire graph).

There is an attempt [7] of embedding RDF into the F-logic or its subcomponent HiLog [8], which has the second order syntax with intensional equality theory, but from our point of view such approach degenerates and uselessly complicates the original elegant syntax of RDF which is one of the most important reason for its wide and unquestionable success.

The attempt [9], [10] of embedding RDF into (S)KIF is analog to HiLog (has the second order syntax with intensional equality theory). It is incomplete [6], and do not address reification and anonymous resources, but has the similar semantics to RDFS: it is not casual, the same author Patrick Hayes defined the semantics for RDF [11].

In the RDF model theory is formally expressed the adoption of the intensional semantics, so that standard extensional FOL language is explicitly refused as a candidate for larger logic embedding framework [11]: "The use of the explicit extension mapping also makes it possible for two properties to have exactly the same values, or two classes to contain the same instances, and still be distinct entities. This means that RDFS classes can be considered to be rather more than simple sets; they can be thought of as ‘classifications’ or ‘concepts’ which have a robust notion of identity which goes beyond a simple extensional correspondence. This property of the model theory has significant consequences in more expressive languages built on top of RDF, such as OWL [4], which are capable of expressing identity between properties and classes directly. This ‘intensional’ nature of classes and properties is sometimes claimed to be a useful property of a descriptive language, [11]"

**Motivation and main contributions:**

Our paper is the formal continuation of this approach to the semantics of RDF databases, considering that RDF is not First-order Logic (FOL), nor in syntax nor in semantics: thus, what we will explore is the formal definition for a particular kind of intensional FOL language with abstraction operator, which we consider as a natural logic framework, which uses only the minimum of the logic machinery for the formal and full embedding of RDFS into FOL language. Notice that the abstraction operator is a necessary prerequisite in order to avoid the second-order syntax as in HiLog and KIF, and the confusion between predicate forms and the (intensional) names of the same forms (which in our theory are terms of the language); as we will see this abstraction operation is based on the particular kind of natural language 'sub-sentences', which differently from ordinary 'complete' logic sentences can not have logic values.

Because of all points discussed in precedence, our approach is very close to these last two attempts, but we prefer to use the pure intensional FOL with an abstraction operator [12], [13], as natural and well suited logic embedding of the RDFS data structures which provides all nice features of the FOL language, all logic connectives (RDF uses only conjunction for its triples), predicates, variables and quantifiers.

Other contribution, and a peculiarity of our approach, is that we consider RDF language and its data structures as terms of an algebra of triples, that is, of RDF graphs, and not as logic theory itself: we distinguish the triple structure < r, p, v > from a predicate form p(r, v) in logic: the first one, as is, for example, a tuple of a database, can be (or not) an element of the given RDF ontology (RDF graph), and have no a truth value, while the second is a logic formula which can be true or false. The embedding of such data structures into logic is analog of the embedding of relations and tuples of relational databases into a predicate logic and ground atoms of the FOL. Basically we will avoid the second order syntax by transforming predicates into intensional terms by using an abstraction operation <₁>: thus, we can see each RDF triple as a term, obtained by abstraction of the predicate corresponding to the property element of this RDF triple.

Consequently a RDF triple < r, p, v >, where r, v are instances or values, is equal to the intensional abstract (term) ⌈p(r, v)⌉ of the binary predicate symbol p in this intensional FOL. Thus, apart of this slight syntax difference, each RDF triple is an intensional term in the intensional FOL for RDFS, and the reification structure <₁> < r, p, v >, p₁, v₁ > is an intensional term ⌈p₁(⌈p(r, v)⌉, v₁)⌉, obtained by abstraction from the logic atom p₁(⌈p(r, v)⌉, v₁).

As we see, this logic embedding is very close to the original RDF syntax, and, as we will see in the rest of this paper, the semantics of this intensional FOL fully incorporates the non standard semantics of RDF [11], [6]. In this framework, individuals and data values correspond to FOL constants, classes and datatypes correspond to unary predicates, properties to binary predicates and sub-class/property relationships correspond to implication.

**The Plan of this paper is the following:**

In Section 2 we introduce the intensional FOL language with abstraction operator, by merging the Bealer’s algebraic and Montague’s possible world approach to the semantics of the conceptual data structures of natural languages: what we obtain is a S5 modal intensional FOL language with intensional equality theory. In Section 3 we define the embedding of RDFS into this intensional FOL and we show the advantages of such logic embedding. Finally, in Section 4 we propose an intensional equivalence for this logic which can be used for the semantic mappings in data integration, especially for the P2P database systems and Semantic Web applications.
II. INTENSIONAL FOL WITH ABSTRACTION

The systematic study of intensional entities has been pursued largely in the context of intensional logic; that part of logic in which the principle of substitutivity of equivalent expressions fails.

Intensional entities are such things as concepts, propositions and properties. What make them 'intensional' is that they violate the principle of extensionality; the principle that extensional equivalence implies identity. All (or most) of these intensional entities have been classified at one time or another as kinds of Universals [14]. They begin with the informal theory that universals (properties (unary relations), relations, and propositions) are genuine entities that bear fundamental logical relations to one another.

To study properties, relations and propositions, one defines a family of set-theoretical structures, one defines the intensional algebra, a family of set-theoretical structures most of which are built up from arbitrary objects and fundamental logical operations (conjunction, negation, existential generalization, etc.) on them.

Syntax:
The fundamental entities are intensional abstracts or so called, 'that'-clauses or a gerundive (or infinitive) phrase. We assume that they are singular terms; Intensional expressions like 'believe', mean', 'assert', 'know', are standard two-place predicates that take 'that'-clauses as arguments. Expressions like 'is necessary', 'is true', and 'is possible' are one-place predicates that take 'that'-clauses as arguments. For example, in the intensional sentence "it is necessary that A", where A is a proposition, the 'that' A is denoted by the <A>, where <A> is the intensional abstraction operator which transforms a logic formula into a term. So that the sentence "it is necessary that A" is expressed by the logic atom \( N(<A>) \), where N is the unary predicate 'is necessary'. In this way we are able to have the higher-order syntax for our intensional logic language (predicates appear in variable places of other predicates), as, for example HiLog [8] where the same symbol may denote a predicate, a function, or an atomic formula.

In the FOL with intensional abstraction we have more fine distinction between an atom A and its use as a term "that A", denoted by <A> and considered as intensional "name", inside some other predicate, and, for example, to have the first-order formula \( \neg A \land P(t, <A>) \) instead of the second-order HiLog formula \( \neg A \land P(t, A) \).

Definition 1: The syntax of the First-order Logic language with intensional abstraction \(<\cdot>\), called \( L_{\omega} \) in [2], is as follows:

Logic operators \((\land, \neg, \exists)\); Predicate letters in \( P \) (functional letters are considered as particular case of predicate letters); Variables \( x, y, z, \ldots \); Abstraction \(<\cdot>\), and punctuation symbols (comma, parenthesis). With the following simultaneous inductive definition of term and formula:

1. All variables and constants (0-ary functional letters in \( P \)) are terms.
2. If \( t_1, \ldots, t_k \) are terms, then \( A(t_1, \ldots, t_k) \) is a formula (\( A \in P \) is a k-ary predicate letter).
3. If \( A \) and \( B \) are formulae, then \( (A \land B), \neg A \), and \((\exists x)A \) are formulae.
4. If \( A \) is a formula and \( \alpha = \{x_1, \ldots, x_n\} \), is a sequence of distinct variables (a subset of free variables in \( A \)), then \(<A>_{\alpha} \) is a term. The externally quantifiable variables are the free variables not in \( \alpha \). When \( n = 0 \), \(<A>_{\alpha} \) is a term which denotes a proposition, for \( n \geq 1 \) it denotes a n-ary relation-in-intension.

An occurrence of a variable \( x_i \) in a formula (or a term) is bound (free) iff it lies (does not lie) within a formula of the form \((\exists x_i)A \) (or a term of the form \(<A>x_1, \ldots, x_m> \). A variable is free (bound) in a formula iff it has (does not have) a free occurrence in that formula.

A sentence is a formula having no free variables. The binary predicate letter \( F_2^1 \) is singled out as a distinguished logical predicate and formulae of the form \( F_2^1(t_1, t_2) \) are to be rewritten in the form \( t_1 = t_2 \). The logic operators \( \forall, \exists \) are defined in terms of \((\land, \neg, \exists)\) in the usual way.

For example, "x believes that A" is given by formula \( B(x, <A>) \) ( \( B \) is binary 'believe' predicate), "Being a bachelor is the same thing as being an unmarried man" is given by identity of terms \(<B(x)>_x = <U(x) \land M(x)>_x \) (with \( B \) for 'bachelor', \( U \) for 'unmarried', and \( M \) for 'man', unary predicates).

Thus, analogously to Boolean algebras which are extensional models of propositional logic, we introduce an intensional algebra as follows. We consider a non-empty domain \( D = D_1 \cup D_{1,1} \), where a subdomain \( D_{1,1} \) is made of particulars (extensional entities), and the rest \( D_1 = D_0 \cup D_{1,1} \cup D_n \) is made of universals (\( D_0 \) for propositions (the 0-ary relations-in-intensions), and \( D_n, n \geq 1 \), for n-ary relations-in-intension.

**Definition 2:** (SYNTAX): Intensional algebra is a structure \( Alg_{int} = \{ D, \text{conj}, \text{disj}, \text{impl}, \text{neg}, \text{pred}, \text{exist}, \tau, f, t, \ldots \} \), with binary operations \( \text{conj} : D_1 \times D_1 \rightarrow D_1, \) \( \text{pred} : D_1 \times D_1 \rightarrow D_{1,1}, \) \( \text{neg} : D_i \rightarrow D_i, \) \( \text{exist} : D_{i+1} \rightarrow D_i, \) for each i \( \geq 0 \); the disjunctions and implications are defined in a standard way by \( \text{disj}(u, v) = \text{neg}(\text{conj}(\text{neg}(u), \text{neg}(v))), \) \( \text{impl}(u, v) = \text{disj}(\text{neg}(u), v), \) for any \( u, v \in D_1; \) \( \tau \) is a set of auxiliary operations [14] intended to be semantic counterparts of the syntactical operations of repeating the same variable once or more times within a given formula and of changing around the order of the variables within a given formula;

f, t are empty set {} and set {<>} (with the empty tuple \(<><>) \in D_{1,1} \) i.e. the unique tuple of 0-ary relation) which may be thought of as falsity and truth, as those used in the relational algebra, respectively.

**Remark:** This definition differs from the original work in [14] where t is defined as \( D, \) and \( \text{conj} : D_i \times D_i \rightarrow D_i, \) i \( \geq 0, \) here we are using the relational algebra semantics for the conjunction. So that we are able to support also structural composition for abstracted terms as, for example, \(<A>(x, y) \land B>_{xy} \), which is not possible in the reduced syntactic version of the Bealer’s algebra.
Semantics:
The distinction between intensions and extensions is important especially because we are now able to have and equational theory over intensional entities (as $\ll A \gg$), that is predicate and function "names", that is separate from the extensional equality of relations and functions. Thus, intensional FOL has the simple Tarski first-order semantics, with a decidable unification problem, but we need also the actual-world mapping which maps any intensional entity to its actual-world extension. In what follows we will identify a possible world by a particular mapping which assigns to intensional entities their extensions in such possible world. That is direct bridge between intensional FOL and possible worlds representation [15], [16], [17], [18], where intension of a proposition is a function from possible worlds $\mathcal{W}$ to truth-values, and properties and functions from $\mathcal{W}$ to sets of possible (usually not-actual) objects.

In what follows we will use one simplified S5 modal logic framework (we will not consider the time as one independent parameter as in Montague’s original work) with a model $\mathcal{M} = (\mathcal{W}, \mathcal{R}, D, \mathcal{V})$, where $\mathcal{W}$ is the set of possible worlds, $\mathcal{R}$ is the reflexive, symmetric and transitive accessibility relation between worlds ($\mathcal{R} = \mathcal{W} \times \mathcal{W}$), $D$ is a non-empty domain of individuals given by Definition 2, while $V$ is a function defined for the following two cases:

1. $V : \mathcal{W} \times F \to \bigcup_{n<\omega} D^n$, with $F$ a set of functional symbols of the language, such that for any world $w \in \mathcal{W}$ and a functional symbol $f \in F$, we obtain a function $V(w, f) : D^{arity(f)} \to D$.
2. $V : \mathcal{W} \times P \to \bigcup_{n<\omega} 2^{D^n}$, with $P$ a set of predicate symbols of the language and $2 = \{ t, f \}$ is the set of truth values (true and false, respectively), such that for any world $w \in \mathcal{W}$ and a predicate symbol $p \in P$, we obtain a function $V(w, p) : D^{arity(p)} \to 2$, which defines the extension $[p] = \{ a \in D^{arity(p)} : V(w, p)(a) = t \}$ of this predicate $p$ in the world $w$.

The extension of a formula $A$, w.r.t. a model $\mathcal{M}$, a world $w \in \mathcal{W}$ and an assignment $g : Var \to D$ is denoted by $[A]^M,g$ or by $[A/g]^M,w$ where $A/g$ is a ground formula obtained from $A$ by assigning values to all its free variables. Thus, if $p \in F \cup P$ then for a given world $w \in \mathcal{W}$ and the assignment function for variables $g$, $[p]^M,w,g = V(w, p) : D^n \to 2$, that is, for any set of terms $t_1, \ldots, t_n$, where $n$ is the arity of $p$, we have $[p(t_1, \ldots, t_n)]^M,w,g = V(w, p)([t_1]^M,w,g, \ldots, [t_n]^M,w,g) \in 2$.

For any formula $A$, $\mathcal{M} \models_{w,g} A$ is equivalent to $[A]^M,w,g = t$, means ‘$A$ is true in the world $w$ of a model $\mathcal{M}$ for assignment $g$’.

The additional semantic rules relative to the modal operators $\Box$ and $\Diamond$ are as follows:

- $\mathcal{M} \models_{w,g} \Box A$ iff $\mathcal{M} \models_{w',g} A$ for every world $w'$ in $\mathcal{W}$ such that $wRw'$.
- $\mathcal{M} \models_{w,g} \Diamond A$ iff there exists a world $w'$ in $\mathcal{W}$ such that $wRw'$ and $\mathcal{M} \models_{w',g} A$.

A formula $A$ is said to be true in a model $\mathcal{M}$ if $\mathcal{M} \models_{w,g} A$ for each $g$ and $w \in \mathcal{W}$. A formula is said to be valid if it is true in each model.

Montague defined the intension of a formula $A$ as follows:

$$[A]^M,g \defeq \{ w \to [A]^M,w,g \mid w \in \mathcal{W} \},$$

i.e., as graph of the function $[A]^M,g : \mathcal{W} \to \bigcup_{w \in \mathcal{W}} [A]^M,w,g$. One thing that should be immediately clear is that intensions are more general than extensions: if the intension of an expression is given, one can determine its extension with respect to a particular world but not vice versa, i.e., $[A]^M,w,g = [A]^M,g(w)$.

In particular, if $c$ is a non-logical constant (individual constant or predicate symbol), the definition of the extension of $c$ is,

$$[c]^M,w,g = \defeq \{ w(c) \mid w \in \mathcal{W} \}.$$ 

The extension of variable is supplied by the value assignment $g$ only, and thus does not differ from one world to the other; if $x$ is a variable we have $[x]^M,g = g(x)$.

Thus the intension of a variable will be a constant function on worlds which corresponds to its extension.

Finally, the connection between Bealer’s non-reductionist and Montague’s possible world approach to intensional logic can be given by the isomorphism (its meaning is that basically we can use the extensionalization functions in the place of Montague’s possible worlds):

$$\mathcal{F} : \mathcal{W} \simeq \mathcal{E},$$

where $\mathcal{E}$ is a set of possible extensionalization functions: Each extensionalization function $h \in \mathcal{E}$ assigns to the intensional elements of $\mathcal{D}$ an appropriate extension as follows:

- for each proposition $u \in D_0$, $h(u) \in 2 = \{ f, t \}$ is its extension (true or false value); for each n-ary relation-intension $u \in D_n$, $h(u)$ is a subset of $D^n$ (n-th Cartesian product of $D$); in the case of particular us $u \in D_{-1}$, $h(u) = u$.

We require that the operations $conj, disj$ and $neg$ in this intensional algebra behave in the expected way with respect to each extensionalization function (in fact as set-intesection, set-union and set-complement respectively); for example, for all $u \in D_0$, $h(conj(u, v)) = h(u) \cap h(v)$ iff $h(u) = h(v) = t$ iff $h(u) = h(v) = t$, $h(neg(u)) = t$ iff $h(u) = f$, etc., where $f$ is empty set and $t$ the set $D$, that is

$$h = h_{-1} + h_0 + \sum_{i \geq 1} h_i : \bigcup_{i \geq 1} D_i \to D_{-1} + 2 + \sum_{i \geq 1} P(D^i)$$

where $h_{-1} = id : D_{-1} \to D_{-1}$ is identity, $h_0 : D_0 \to 2$ assigns truth values in $2 = \{ f, t \}$, to all propositions, and $h_i : D_i \to P(D^i), i \geq 1$, assigns extension to all relations-intension, where $P$ is the powerset operator. Thus, the intensions can be seen as names of abstract or concrete entities, while the extensions correspond to various rules that these entities play in different worlds.

Among the possible functions in $\mathcal{E}$ there is a distinguished function $\mathcal{F}$ which is to be thought as the actual extensionalization function: it tells us the extension of the intensional elements in $\mathcal{D}$ in the actual (current) world.

In what follows we will use the join operator $\bowtie$, such that for any two relations $r_1, r_2$ their join is defined by:

$$r_1 \bowtie r_2 = \{ (a, c, b) \mid (a, c) \in r_1 \text{ and } (c, b) \in r_2 \},$$

where $a, c, b$ are tuples (also empty) of constants, so that

$$r_1 \bowtie \{ \} = \{ \} \text{ and } r_1 \bowtie \{ \langle \rangle \} = r_1.$$
Definition 3: (SEMANTICS): The operations of the algebra $A_{int}$ must satisfy the following conditions, for any $h \in \mathcal{E}$, with $f = \{ \}, t = \{ \varepsilon \}$, and $u_1, \ldots, u_i \in \mathcal{D}$.

1. $h(\text{con}(u, v)) = h(u) \bowtie h(v)$, for $u, v \in D_1$.
2.1 $h(\text{neg}(u)) = t$ iff $h(u) = f$, for $u \in D_0$.
2.2 $< u_1, \ldots, u_i > \in h(\text{neg}(u))$ iff $< u_1, \ldots, u_i > \not\in h(u)$, for $u \in D_i, i \geq 1$.
3.1 $h(\text{exist}(u)) = t$ iff $h(u) = t$, for $u \in D_0$.
3.2 $h(\text{exist}(u)) = t$ iff $\exists u_i (u_1 \in h(u))$, for $u \in D_1$.
3.3 $< u_1, \ldots, u_i-1 > \in h(\text{exist}(u))$ iff $\exists u_i (u_1 < u_1, \ldots, u_i-1, u_i > \in h(u))$, for $u \in D_i, i \geq 2$.
4.1 $h(\text{pred}(u, u_1)) = t$ iff $u_1 \in h(u)$, for $u \in D_1$.
4.2 $< u_1, \ldots, u_i-1 > \in h(\text{pred}(u, u_1))$ iff $< u_1, \ldots, u_i-1, u_i > \in h(u)$, for $u \in D_i, i \geq 2$.

Notice that this definition for the conjunction operation is different from the original work in [2] where

1.1 $< u_1, \ldots, u_i > \in h(\text{con}(u, v))$ iff $< u_1, \ldots, u_i > \in h(u) \cap h(v)$, for $u, v \in D_i, i \geq 1$.
1.2 $h(\text{con}(u, v)) = t$ iff $h(u) = h(v) = t$, for $u, v \in D_0$.

Once one has found a method for specifying the denotations of singular terms of $\mathcal{L}_w$ (take in consideration the particularity of abstracted terms), the Tarski-style definitions of truth and validity for $\mathcal{L}_w$ may be given in the customary way. An intensional interpretation $I$ [14] maps each i-ary predicate letter of $\mathcal{L}_w$ to i-ary relations-in-intention in $D_1$. It can be extended to all formulae in usual way. What is being south specifically is a method for characterizing the denotations of singular terms of $\mathcal{L}_w$ in such a way that a given singular term $\ll A \gg_{x_1, \ldots, x_m}$ will denote an appropriate property, relation, or proposition, depending on the value of $m$.

Thus, the mapping of intensional abstracts (terms) in $A_{BS} \subseteq \mathcal{L}_w$ into $D_1$, given in original version of Bealer [14], will be called denotation $\text{den} : A_{BS} \rightarrow D_1$, such that the denotation of $\ll A \gg$ is equivalent to the meaning of a proposition $A$, that is, $\text{den}(\ll A \gg) = I(A) \in D_0$.

In the case when $A$ is an atom $F^{m}(x_1, \ldots, x_m)$ then $\text{den}(\ll F^{m}(x_1, \ldots, x_m) \gg_{x_1, \ldots, x_m}) = I(F^m) \in D_m$.

The denotation of a more complex abstract $\ll A \gg_{\alpha}$ is defined in terms of the denotation(s) of the relevant syntactically simpler abstract(s) [14].

For example $I(A(x) \land B(x)) = \text{con}(I(A(x)), I(B(x)))$, $I(\neg p) = \text{neg}(I(p))$. A sentence $A$ is true relative to $I$ and the intensional algebra, iff its actual extension is equal to $t$, that is, $Tr(\ll A \gg)$ iff $\kappa(I(A)) = t$, where $Tr$ is unary predicate for true sentences.

For the predicate calculus with individual constants (variables with fixed assignment, proper names, and intensional abstracts) we introduced an additional binary algebraic operation $\text{pred}$ (singular predication), such that for any two $u, v \in \mathcal{D}$, for any extensionalization function $h$ holds $h(\text{pred}(u, v)) = t$ iff $v \in h(u)$. So we are able to assign appropriate intensional value (propositional meaning) to a ground atom $I(c) \in \mathcal{L}_w$ with individual constant $c$, that is, $I(A(c)) = \text{pred}(I(A(x)), I(c))$ is a term in this intensional algebra with $I(A(x)) \in D_1$ and $I(c) \in D_1$.

So that $h(I(A(c))) = h(\text{pred}(I(A(x)), I(c))) = t$ iff $I(c) \in h(I(A(x)))$, that is, in the extension $h$, $A(c)$ is true (that is, the extension of the propositional meaning of $A(c)$) iff the interpretation of $c$ is in the extension of the interpretation of the predicate $A(x)$. Or, for example, for a given formula with intensional abstract, $B(\ll A(x, y) \gg_{x, y}) \in \mathcal{L}_w$, we have that $h(I(B(\ll A(x, y) \gg_{x, y}))) = h(\text{pred}(I(B(z)), \text{den}(\ll A(x, y) \gg_{x, y}))) = t$ iff $\text{den}(\ll A(x, y) \gg_{x, y}) \in h(I(B(z)))$, where $I(B(z)) \in D_1$ and $\text{den}(\ll A(x, y) \gg_{x, y}) \in D_2$.

We can connect $\mathcal{E}$ with a possible-world semantics, where $w_0 = F^{-1}(k)$ denotes the actual world in which intensional elements have the extensions defined by k. Such a correspondence, not present in original intensional theory [19], is a natural identification of intensional logics with modal Kripke based logics.

Definition 4: (Model): A model for the intensional FOL is the S5 Kripke structure $M_{int} = (W, R, D, V)$, with intensional identity defined as follows:

$\ll A \gg_{\alpha} \equiv \ll B \gg_{\alpha}$ iff $\square (A \equiv B)$

where $W = \{ F^{-1}(h) | h \in \mathcal{E} \}$, $R = W \times W$. The symbol $\square$ is the universal "necessity" S5 modal operator.

Remark: this semantics is equivalent to the algebraic semantics for $\mathcal{L}_w$ in [2] for the case of the conception where intensional entities are considered to be identical if and only if they are necessarily equivalent. Moreover, for this intensional FOL holds the soundness an completeness: For all formulae $A$ in $\mathcal{L}_w$, $A$ is valid if and only if $A$ is a theorem of this first-order S5 modal logic with intensional equality [2].

Example 1: Let two predicate forms $A(x)$ and $B(x)$ be intensionally equal, that is $I(A(x)) = I(B(x))$, then for any $h \in \mathcal{E}$ holds that $h(I(A(x))) = h(I(B(x)))$, i.e., have the same extension, thus $A(x) \equiv B(x)$ is true, (or $(A(x) \Rightarrow B(x)) \land (B(x) \Rightarrow A(x))$ is true), in each world $F^{-1}(h)$. Consequently $\square (A(x) \equiv B(x))$ is true, and from the definition holds the intensional identity for their intensional abstracts, $\ll A(x) \gg_x = \ll B(x) \gg_x$, and finally, $\text{den}(\ll A(x) \gg_x) = \text{den}(\ll B(x) \gg_x)$.

Vice versa, if $\square (A(x) \equiv B(x))$ then $\ll A(x) \gg_x = \ll B(x) \gg_x$, and $\text{den}(\ll A(x) \gg_x) = \text{den}(\ll B(x) \gg_x)$, and from the fact that a denotation of $\ll A(x) \gg_x$ is equal to the meaning of $A(x)$, that is, equal to $I(A(x))$, we obtain that $I(A(x)) = I(B(x))$, and consequently $A(x)$ and $B(x)$ are intensionally equal: so the necessity modal formula $\square (A(x) \equiv B(x))$ corresponds to the intensional equality of $A(x)$ and $B(x)$.

It is easy to verify that the intensional equality means that in every possible world $w \in W$ the intensional entities $A$ and $B$ have the same extensions (as in Montague’s approach), moreover:

Proposition 1: (Bealer-Montague connection): For any intensional entity $\ll A \gg$ its extension in a possible world $w \in W$ is equal to $F(w)(\text{den}(\ll A \gg)) = [A]_{\text{int}}^{M, \theta}(w)$. 
Proof: Directly from the definition of the identification of a possible world \( w \) of Montague’s approach with the extensional function \( h = \mathcal{F}(w) \in E \) in the Bealer’s approach, where \([A]_n^{M_d}\) is the "functional" intension of Montague, and \(<A>\) is an intensional abstract term of Bealer’s logic.

III. RDF DATA STRUCTURES AND INTENSIONAL FOL SEMANTICS

In this section we will make a confrontation of the actual RDF MT (Model Theory) and the proposed intensional logic embedding of RDF into modal S5 FOL \( L_\omega \).

**RDF Model Theory:** An interpretation \( I_{RDF} \) of the RDF vocabulary \( V \), that is, RDF model theory, is a triple \( < IR, IEXT, IS > \), where

1. A non-empty set \( IR \) is the domain (of resources), called domain of universe of the \( I_{RDF} \).
2. A set \( IP \), called the set of properties of \( I_{RDF} \).
3. A mapping \( IEXT : IP \rightarrow \mathcal{P}(IR \times IR) \), mapping each property in \( IP \subseteq IR \) into the set of pairs which identify the arguments for which the property is true.
4. A mapping \( IS \) from URI references in \( V \) into \( IR \).
5. A mapping \( IL \) from typed literals in \( V \) into \( IR \).
6. A distinguished subset \( LV \) (of real entities that ‘exist’) of \( IR \), called the set of literal values, which contains all the plain literals in \( V \).

For example, the denotations of ground RDF graphs are as follows:

1. If \( E \) is a plain literal "aaa" in \( V \) then \( I_{RDF}(E) = aaa; \)
2. If \( E \) is a plain literal "aaa"@ttt in \( V \) then \( I_{RDF}(E) = < aaa, ttt >; \)
3. If \( E \) is a typed literal in \( V \) then \( I_{RDF}(E) = IL(E); \)
4. If \( E \) is a URI reference in \( V \) then \( I_{RDF}(E) = IS(E); \)
5. If \( E \) is a ground triple \( < r, p, v > \) then \( I_{RDF}(E) = \frac{1}{t} \) if \( r, p \) and \( v \) are in \( V \), \( I_{RDF}(p) \) is in \( IP \) and \( I_{RDF}(r), I_{RDF}(v) \) is in \( IEXT(I_{RDF}(p)); \) otherwise \( I_{RDF}(E) = f. \)
6. If \( E \) is a ground RDF graph then \( I_{RDF}(E) = f \) if \( I_{RDF}(E') = f \) for some triple \( E' \) in \( E \), otherwise \( I_{RDF}(E) = t. \)

**Embedding of RDFS into intensional FOL:**

In what follows we will first show how to embed the RDF triples into the intensional FOL defined in Section II, and show that it is a conservative extension of RDF.

**Definition 5:** Each RDFS ontology can be embedded into the intensional FOL \( L_\omega \) as follows:

1. The individuals and data values correspond to FOL constants, the classes and datatypes correspond to unary predicates, the properties to binary predicates and the subclass/property relationships correspond to the implication.
2. Each RDF triple \( < r, p, v > \) can be represented as a formula of the intensional FOL \( L_\omega \) by:
   1. If \( p \) is the special URI reference \( rdf:type \), then \( < r, p, v > \iff v(r) \).
   2. If \( p \) is the special URI reference \( rdfs:subClassOf \), then \( < r, p, v > \iff r(x) \Rightarrow v(x) \).

Otherwise:

**2.3.1 case when \( r \) and \( v \) are classes,**

\[ < r, p, v > \iff p(<r(x)>_x, <v(y)>_y), \]

where in the right side, \( p \) is a binary predicate symbol introduced for the property, while \( r, v \) are unary predicate symbols introduced for the classes, subject and object respectively.

**2.3.2 case when \( r \) is a class and \( v \) is an instance or value,**

\[ < r, p, v > \iff p(<r(x)>_x, v); \]

**2.3.3 case when \( v \) is a class and \( r \) is an instance or value,**

\[ < r, p, v > \iff p(r, <v(y)>_y); \]

**2.3.4 case when \( r \) and \( v \) are the instances or the values (ground triple),**

\[ < r, p, v > \iff p(v). \]

3. The RDFS ontology graph \( O = \{ < r_i, p_i, v_i > | 1 \leq i \leq n, n \geq 2 \} \) is embedded into the formula \( \alpha_1 \land ... \land \alpha_n \), where \( \alpha_i \) is the embedding of a triple \( < r_i, p_i, v_i > \), \( 1 \leq i \leq n \).

A blank node \( x \) in a triple \( < x, p, v > \) is embedded into the formula \( \exists(x)(p(x, <v(y)>_y) \iff v \) is a class, into the formula \( \exists x[p(x, v)] \) otherwise.

The reified RDF statements are embedded into the intensional terms by nidifying abstraction operator. For example, \( < < r, p, v >, p_1, v_1 > \) is embedded into \( L_\omega \) as a formula \( p_1(<p(r,v)>, v_1) \).

Notice that this embedding of RDF triples, we consider them as intensional terms, and not as logic sentences as in current RDF interpretation and its model theory, which identify a ground triple \( < r, p, v > \) with a logic predicate form \( p(r, v) \). As we pointed in the introduction, such identification will be epistemically equivalent to identifying a tuple of a relation \( R \), with the ground atom \( R(t) \), which is not acceptable: tuples of constants are considered as the terms from which we can build logic predicate forms and sentences.

Other advantage in this embedding is that we do not need to define the meaning of a predicate \( rdfs:subClassOf \) by the axioms for reflexivity and transitivity as in RDFS. We do not need more the RDF axiom for the relationship between \( rdfs:subClassOf \) and \( rdf:type \) also.

**Example 2:** Let us consider some of the properties of RDFS which can not be embedded into standard (extensional) FOL, but are perfectly accepted by intensional FOL \( L_\omega \). On the one hand, \( rdfs:Resource \) is an instance of \( rdfs:Class \); on the other hand, \( rdfs:Class \) is a sub-class of \( rdfs:Resource \). Thus \( rdfs:Resource \) is an instance of its sub-class: it is a contradiction for a standard extensional FOL. But in the intensional FOL \( L_\omega \), we have that, for example in the actual world with the extensionalization function \( k \), and an intensional interpretation \( I: I(rdfs:Resource) \in D_1 \) is an instance of a denotation of the intensional term \( I(rdfs:Class) \in D_1 \), that is \( I(rdfs:Resource) \in k(I(rdfs:Class)) \), while the denotation of \( I(rdfs:Class) \) is a subset of a denotation of the \( I(rdfs:Resource) \), i.e., \( k(I(rdfs:Class)) \subseteq k(I(rdfs:Resource)) \subseteq D_1 \). So we have the following fact in the intensional logic \( L_\omega \):

\[ I(rdfs:Resource) \in k(I(rdfs:Class)) \subseteq k(I(rdfs:Resource)). \]
Analysis of the embedding:

Let us consider the correspondence between RDF non standard model, with an interpretation \( I_{\text{RDF}} : V \rightarrow IR \), where \( V \) is the RDF vocabulary (in our case it is also part of the syntax of the intensional FOL language \( L_\omega \)), and \( IR \) is the domain of RDF resources (in our case \( IR = D_{-1} \cup D_0 \cup D_1 \cup D_2 \subseteq D \), with \( IP = D_2 \) (the set of RDF properties is the set of binary relation-ini-intensions), and \( LV \) (the set of literal values of RDF) is a subset of \( D_{-1} \).

In the figure in the next page is schematically represented the embedding of the non standard RDFS model into the intensional FOL logic model: in grey color are represented the original RDF model components. As we can see there are two main differences:

1. The original RDF MT has a partial intensional structure: only its properties in \( IP \) (embedded into \( D_2 \subseteq D \)) are intensional entities, and for them there exists the extensionalization function \( I_{\text{EXT}} \). In Intensional FOL language all intensional entities have an extension: thus, instead of the complex derivation of the Class extension \( I_{\text{EXT}} \) in RDF MT, defined through the extension of \( IS(\text{rdf:type}): I_{\text{EXT}}(x) = \{ y \in I_{\text{EXT}}(IS(\text{rdf:type})) \} \)

we can directly find the extension by the mapping component \( h_1 \) of the extensionalization function \( h \) of the intensional FOL model.

2. The other difference is that in a language \( L_\omega \), for any ground RDF triple \( < r, p, v > \), we have the logic atom \( p(r, v) \) and its intensional abstraction, that is, a term \( \langle p(r, v) \rangle \), such that the denotation of this term \( \text{den}(\langle p(r, v) \rangle) \) is equal to the meaning of the atom \( p(r, v) \), that is, \( \text{den}(\langle p(r, v) \rangle) = I(p(r, v)) \). So, the RDF interpretation of the ground triple \( < r, p, v > \), \( I_{\text{RDF}}(< r, p, v >) \), where \( r, v \) are individual constants, corresponds to the following extension, in the actual world, of the intensional 'name' \( I(p(r, v)) = \text{den}(\langle p(r, v) \rangle) \) of the ground atom \( p(r, v) \), where \( I \) is the intensional interpretation for \( L_\omega \):

\[
I_{\text{RDF}}(< r, p, v >) = k(I(p(r, v))) = k(\text{pred}(I(p), I(r, v))) = t \quad \text{iff} \quad I(r, v) = \langle I(r), I(v) \rangle \in k(I(p)).
\]

3. The domain \( IR \) in RDFS MT has no any algebraic structure, differently from the algebraic structure over a domain \( D \), where, for example, \( I(A \land B) = \text{conj}(I(A), I(B)) \), and \( \text{den}(\langle A \land B \rangle) = \text{conj}(\text{den}(\langle A \rangle), \text{den}(\langle B \rangle)) \). So that the denotation of all complex formulae in \( L_\omega \) can be computed from the denotation of its subformulae (the denotation relation of all the complex intensional terms in \( L_\omega \) is recursively defined).

4. While the different extensions of the actual RDFS framework, for example, by introducing the predicates with arity bigger than two, would modify its non standard model theory, in the simple and linear model structure of the intensional FOL \( L_\omega \) we do not need any change for such extensions.

Other advantages of the assumption of this Intensional FOL \( L_\omega \) for embedding RDFS, w.r.t. the actual RDF model theory are as follows:

1. We do not force the RDF triples and graphs to have logic values, but consider them what they are: complex data structures.

2. With FOL language \( L_\omega \) we have the full logic inference, based on the intensional equality theory. We can enrich original RDFS graphs also with other n-ary predicates, which can be useful for more compact ontology representations. We can also introduce integrity constraints for such ontologies, which can use more complex logic formulae with quantifiers that RDF language.

In what follows we will prove that disadvantages of the actual non standard RDF model theory (RDF MT), taken from [20], does not hold in its embedding in the intensional FOL \( L_\omega \):

1. **Too Few Entailment**: Let us consider two formulae, \( q(x) \equiv \text{Student}(x) \land \text{Employee}(x) \land \text{European}(x) \), and \( q_1(x) \equiv \text{Student}(x) \land \text{Employee}(x) \), such that holds \( q(\text{John}) \). Than, in RDF MT, since every concept is also an object, there is no guarantee that if "John is an instance of the class \( \text{Student} \cap \text{Employee} \) \( \cap \text{European} \)" also "John is an instance of the class \( \text{Student} \cap \text{Employee} \)". But by the embedding into intensional FOL \( L_\omega \), we have that \( q(\text{John}) \Rightarrow q_1(\text{John}) \) is true, that is, in the actual world,

\[
k(I(q(x))) = k(\langle q(\text{Student} \land \text{Employee} \land \text{European}) \rangle) = k(\langle q(\text{Student}) \rangle) \cap k(\langle \text{Student} \rangle) \cap k(\langle \text{Employee} \rangle) \cap k(\langle \text{European} \rangle)
\]
= \kappa(I(q_1(x))).

2. "Contradiction" Classes: Such problem do not exist when RDFS is embedded into the intensional FOL: we can not define a class eg:C as an instance of itself and add a cardinality constraint "\(= 0\)" on the rdf:type property of it: \(C(<C>)\), with \(\kappa(I(C(<C>)))\) \(\neq \kappa(I(C(x))))\), is a formula in \(L_\omega\) which defines that the class eg:C as an instance of itself.

3. Size of the Universe: The semantic consequences of the RDFS thesis include that all RDF properties (predicate symbols) are elements of discourse (that is, of \(IR\)) is valid also in this embedding: It is not possible, for example, to have an interpretation \(I\) such that John is a member of Person, but not a member of Car, and there is only one object in the universe: considering that all of them are intensionally different entities, we must have their denotations in \(D\), in order to specify that \(Person(John)\), and \(\neg Car(John)\) are true in this interpretation: in that case \(I(Person(John)) = pred(I(Person), I(John)) = pred('Person', 'John') = t\) iff \(\exists 'John' \in \kappa('Person')\), and \(I(\neg Car(John)) = \neg neg(pred(I(Car), I(John))) = t\) iff \(\exists 'John' \notin \kappa('Car')\).

This is characteristic for all kind of logics which distinguish the intension from the extension of its entities, and differentiate them from the logics with extensional equality theory.

IV. INTENSIONAL VIEW-BASED DATA INTEGRATION

In what follows we consider an RDF ontology as a basic data strata of a peer database. In order to define the logical views over such RDF ontologies we will use the logic theory obtained by embedding this RDF-ontology \(O\) into intensional FOL with abstraction.

It has become customary to define the notion of RDF programming via a long list of triples (binary RDF-properties) which resource must posses. In relational language, instead, the data are conceptually grouped around n-ary predicates with a set of attributes which together describe different properties attributed to such logical concept.

Views are an established technology for both relational and object-oriented databases. They are mainly used to provide data customization, that is, the adaptation of content to meet the demands of specific applications and users, so that they present the key technology for integrating heterogeneous and distributed systems, facilitating interoperability by hiding the foibles of each information component and gluing individual components together to form an integrated P2P application system.

The simple idea is to see a view \(\varphi(x_1, ..., x_n)\) over an RDF-ontology, expressed as a formula in some query language for RDF databases (for example, as a SELECT-FROM-WHILE structure in [21], [22], as n-ary relation-in-intension (intensional name) whose extension in the actual world (a possible world corresponds to the possible RDF database extension) is equal to its query answer.

Example 3: (from [23]) To find all (sculpture, museum) pairs, in an RDF peer database, where the sculpture was created by Rodin, the museum houses the given sculpture, and the museum Web site was not modified since Jan 1, 2001, we can define the following query: rdql – query =:

\[\text{SELECT ?sculpture, ?museum WHERE (?sculptor, <ns1:name>, "Rodin"),
\text{(?sculptor, <ns1:creates>, ?sculpture),
\text{(?sculpture, <ns1:exhibited>, ?museum),
\text{(?museum, <ns1:last-modified>, ?date),
AND ?date < 2001-01-01 USING ns1 FOR http://www.icom.com/schema1 >
and the correspondent view, by:
\varphi(x_1, x_2) = \text{CREATEVIEW view – name AS rdql – query,
where \varphi(x_1, x_2) is a virtual predicate, obtained by the following FOL translation (Def.1):
\varphi(x_1, x_2) \leftarrow \text{name(y, "Rodin")} \land \text{creates(y, x_1) \land exhibited(x_1, x_2)}
\land \text{last-modified(x_2, z) \land z < 2001-01-01,}
where \(x_1, x_2\) are free variables for sculpture and museum respectively.}

Remark: Instead of a view-name we can use directly the intensional term \(\lnot \varphi(x_1, x_2)\). The extension of this intensional view in n-ary relation obtained as the answer to this query in the actual world.

The peer database in this framework is just the logic theory, defined as union of the RDF database’s FOL-embedding and a number of views defined over it (they constitute a virtual user-type interface). Such embedding of a RDF database, together with its view-extension can be used as mean for intensional matching with other peer databases in a Web P2P networks [24], [25]. There are the following nice properties for this RDF peer database framework:

1. The FOL embedding of standard RDF database together with views corresponds to the definite logic program, thus the database model of such RDF peer database has a unique Herbrand model, differently from standard Data Integration Systems (DIS) with relational schema ontology, which usually suffer incomplete information and very high number of Herbrand models (possible completions of this withdraw): so, the query answering from peer databases with RDF database is very efficient (polynomial complexity).

2. The defined views can be materialized and there are efficient algorithms [23] for maintenance of RDF views when a new RDF data triples are inserted in a peer database, when some of them are deleted, or modified.

3. From the theoretical point of view, the possibility to transform original RDF based peer database into the more expressive, but decidable, FOL sublanguages, is important if we want to add also integrity constraints over a peer database ontology (for example, in the simplest case, the key constraints over a view).

Now we can introduce the new intensional equivalence relation between intensional entities, which can be used as intensional mapping between databases: we can define the set of intensionally equivalent views over two different databases
as semantic mapping between them. Such approach to the mapping between RDF ontologies has been recently proposed in [26].

**Definition 6:** (Intensional Equivalence) : the two intensional entities $A \triangleright_{\alpha} B$ are intensionally equivalent $A \triangleright_{\alpha} B$ iff $A =_{\alpha} B$.

This equivalence defines the QUOTIENT rule-based extensions also. Web architecture, for rich database structures able to support language which can be used for the logic layer in the Semantic FOL language.

between different RDFS concepts and data structures, and the will give a clear light on the nature of deep interdependency of RDFS data structures into the intensional FOL language view-based mappings in the data integration and a Web based kind of the intensional sional semantics RDFS databases. We defined also the new embedding and a very natural extension of intrinsically inten-

This definition of equivalence relation is the flat-accumulation only world also, and vice versa. In what concerns this paper we will consider only the actual world $w_0 = F^{-1}(\{k\})$. Moreover, the set of basic intensional equivalences are designed by users/developers, thus the definition above is only of theo-

This definition of equivalence relation is the flat-accumulation case presented in [27], [28], [29]: if the first predicate is true in some world then the second must be true in some world also, and vice versa. In what concerns this paper we will consider only the actual world $w_0 = F^{-1}(\{k\})$. Moreover, the set of basic intensional equivalences are designed by users/developers, thus the definition above is only of theoretical interest but useful to understand the meaning of the intensional equivalence: the inference deduction of this logic, able to derive all other intensionally equivalent formulæ, is that of the S5 modal logic with intensional equality theory. The quotient intensional FOL $L_{\omega}/\approx$ is fundamental for query answering in intensional P2P database mapping systems: given a query $q(x)$ over a peer $P_i$, the answer to this query is defined as the extension of the intensional concept $\langle q(x) \rangle_{\omega}$, in the intensional P2P logic $L_{\omega}/\approx$.

**REFERENCES**


