Semantics of RDF(S) and OWL

Ivan Herman, W3C

(Last update: 28 Sep 2007)

> On formal semantics for RDF(S) and OWL

- RDF(S) and OWL are (relatively) complicated specifications
- An *unambiguous* interpretation of the standard is very important
  - this was missing from the 1st version of RDF which did lead to some problems…
- This means a *formal semantics* of these languages
- All details of a formal semantics are not interesting for all users
  - most programming languages, query languages, etc, have a formal semantics; programmers happily use those tools without knowing about the formal semantics
  - …but it is *very* useful to have an idea about it!

> Even more important for RDF(S) and OWL

- RDFS and OWL are ALSO used for *inferences*
- It is very important to *understand what this means*; and the formal semantics used also defines that

> Role of model theory

- The semantics of RDFS and OWL is based on *model theory*
- Essentially, it represents logic systems by sets; thereby using the apparatus of mathematical sets as a tool for characterization
- In what follows, we give a very short introduction to see how this works
- For this, *forget about OWL and RDFS for a while…*

> Logics: what is it?

- It consists of three steps, essentially:
  1. define a type of logics by defining the terms it can use (i.e., the concepts it has) and what types of relations may exist among terms
  2. define a specific “universe” with a vocabulary
  3. define a series of “facts” or “axioms” within the constraints of that logic type
- I.e., there are many different types of logic; some them have been the subject of active research

> Take a very simple case

- Define the type $\mathcal{E}x$:
  - the terms are "classes" (*A*, *B*, …) and "individuals" (*a*, *b*, …)
  - there are two possible relationships between classes:
    1. "subclassing" i.e., $A \sqsubseteq B$, $B \sqsubseteq C$, etc., and
    2. "equivalence", i.e. $A \equiv B$
  - there is one relationship combining individuals to classes: "typing" i.e. $a{:}A$, $b{:}A$,…
- An example for a universe: the vocabulary: $\{a, A, B\}$ with the axioms $\{a{:}A, A \sqsubseteq B\}$
- The goal is to define a semantics so that we could *infer* $a{:}B$

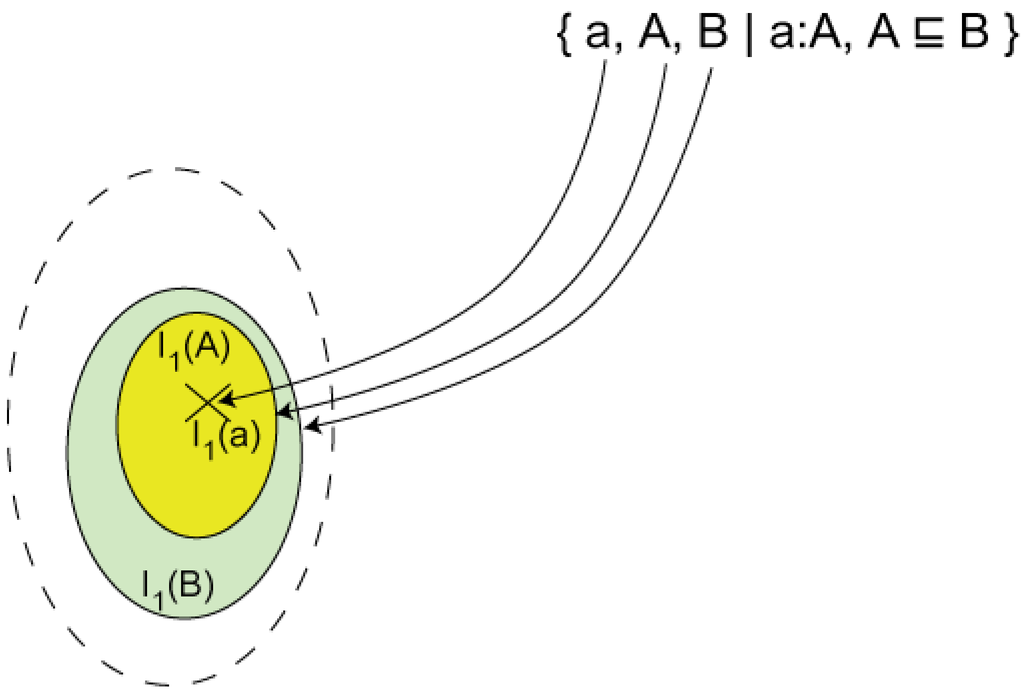
> Interpretations

- An interpretation is a *mapping* from the universe's vocabulary to (mathematical) sets and elements
- For a specific logic family $\mathcal{L}$, an interpretation must follow certain rules to be considered an "$\mathcal{L}$-interpretation"
- For a specific universe, there may be many different $\mathcal{L}$-interpretations!

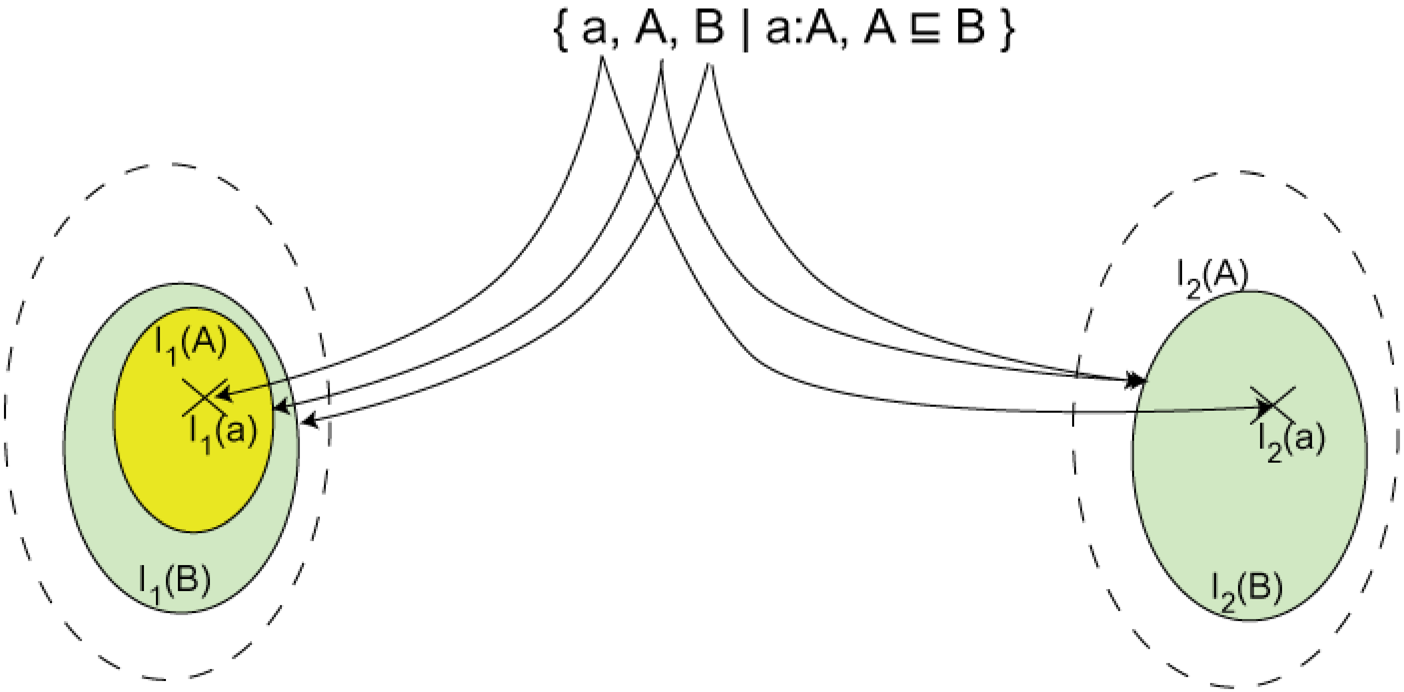
> Example: $\mathcal{E}x$-interpretations

- If a universe U is defined using $\mathcal{E}x$, then an interpretation I of U is an $\mathcal{E}x$-Interpretation if and only if:
  - $A \sqsubseteq B$ if an only if $I(A) \subset I(B)$ (for all A-s and B-s)
  - $A \equiv B$ if an only if $I(A) = I(B)$ (for all A-s and B-s)
  - a:B if an only if $I(a) \in I(B)$
- The interpretation formalizes the intuition of individuals belonging to a classes
- Interpretations may be different! For example: $I_1(A) = I_1(B)$ and $I_2(A) \neq I_2(B)$
  - both are valid $\mathcal{E}x$-Interpretations, though they map to different sets

> Example: $\mathcal{E}x$-interpretations (cont.)



> Example: $\mathcal{Ex}$-interpretations (cont.)



> What is inference?

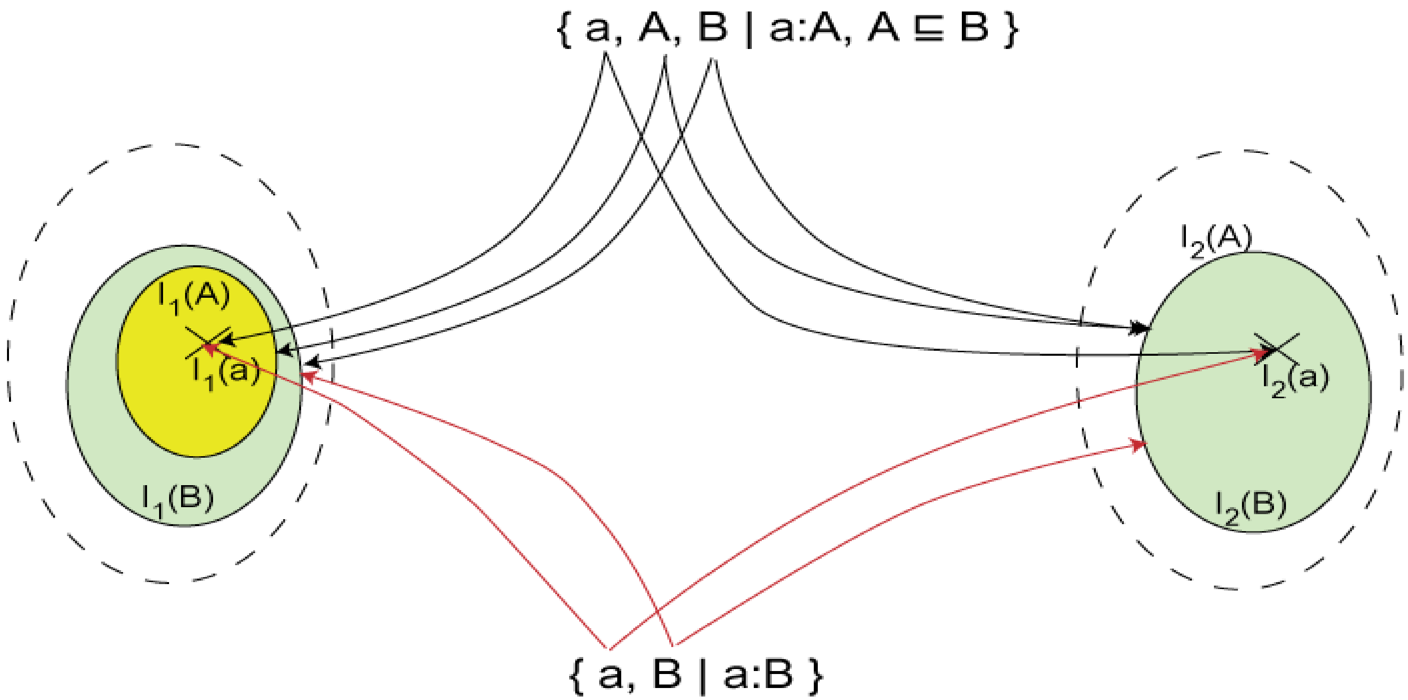
- We have $\boldsymbol{U} = \{a,A,B|a{:}A, A \sqsubseteq B\}$, a universe in $\mathfrak{Ex}$
- We would like to <u>*infer*</u> $\{a,B|a{:}B\}$
  - the usual notation is $\boldsymbol{U} \vDash \{a,B|a{:}B\}$ (or $\boldsymbol{U} \vDash_{\mathfrak{Ex}} \{a,B|a{:}B\}$)
- The formal definition of inference (or "entailement") is:

If $\boldsymbol{V}$ and $\boldsymbol{W}$ are universes in $\mathcal{L}$, then $\boldsymbol{V} \vDash_{\mathcal{L}} \boldsymbol{W}$ if and only if <u>*all*</u> valid $\mathcal{L}$-interpretations of $\boldsymbol{V}$ are also valid $\mathcal{L}$-interpretations of $\boldsymbol{W}$

> What is inference? (cont.)

- In plain English: the definition of an $\mathfrak{Ex}$-interpretation *abstracts* the important aspects of that logic family *and nothing else*
- One can also define equivalence:
  $V \approx W$ means $W \models V$ *and* $V \models W$
- It is easy to see that $\{a,A,B|a{:}A,A \sqsubseteq B\} \models_{\mathfrak{Ex}} \{a,B|a{:}B\}$
  - one has to follow a fairly simple and standard reasoning for sets
  - one can also implement algorithms for general inference checking in $\mathfrak{Ex}$

> Inference example



> A related concept: consistency

- The question: is a universe ***U*** without internal contradictions? Is it *consistent*?
- The formal definition is also based on model theory:

A universe ***U*** in a logic $\mathcal{L}$ is consistent ($\models_{\mathcal{L}}$ ***U***) if and only if there exist at least one $\mathcal{L}$-interpretation for ***U*** (in other words: ***U*** is inconsistent if no $\mathcal{L}$-interpretation can be constructed).

> Many different types of logics

- Of course, *Ex* is only a toy example
- There are many different types of logics (First Order Logic, F-Logic, Horn Logic,…)
  - they differ in the terms and relations they allow
  - but the model theory background is usually identical

> A slightly more complex logic: $\mathcal{ALH}$

- Like $\mathcal{Ex}$, it operates on “individuals” and “classes”
- But it also have “roles”: binary relations on individuals
- It also has two special symbols: $\top$ and $\bot$ (more or less for “true” and “false”)
- $\mathcal{ALH}$ also allows more relationships for axioms:
  - $A \sqcap B$ and $\neg A$ for classes (and also $A \sqsubseteq B$, like in $\mathcal{Ex}$)
    - intersection and negation of classes, respectively
  - If R is a role, then $A \equiv \forall R.C$ or $B \equiv \exists R.\top$ are also possible
    - defining classes by restrictions, ie, “all values must be in C”, and “there must be at least one value”

> $\mathcal{ALH}$-Interpretations

- Defining $\mathcal{ALH}$-interpretations means defining how a mapping I should behave v.a.v. individuals, classes, roles, and the predefined relationships
- The definition is pretty obvious, though:
  - $I(A \sqcap B) = I(A) \cap I(B)$,…
  - $I(R) \subset \Delta \times \Delta$ (where Δ is the target set of the interpretation)
  - $I(\forall R.C) = \{x \mid \forall y: \langle x,y \rangle \in I(R) \Rightarrow y \in I(C)\}$
  - etc.
- This definition leads to the notions of $\mathcal{ALH}$-inference and $\mathcal{ALH}$-consistency

> Description Logics (note the plural!)

- $\mathcal{ALH}$ is a special form of Description Logic
- A family of logics:
  - have separate categorization for classes, individuals, and roles
  - description logics differ from one another on what relationships are allowed among those
    - the names of description logics types, like $\mathcal{AL}$, $\mathcal{ALH}$, $\mathcal{ALCR}^{+}$, $\mathcal{SHIQ}$, etc, usually refer to what is allowed
      - e.g., the letter $\mathcal{H}$ means that subclassing is allowed, $\mathcal{I}$ means that inverse roles can be defined, etc.
  - semantics is defined using model theory
- Description logics have been developed for knowledge representations, ontologies, formal thesauri, etc.

> Inference in Description Logic

- The notion of inference for a, say, $\mathcal{ALH}$ vocabulary ***V*** and ***W*** is the same as in the general case
- Of course, the algorithms to decide whether ***V*** $\models_{\mathcal{ALH}}$ ***W*** are much more complicated than for $\mathcal{Ex}$
- The most widespread algorithm for Description Logics is the “tableau” algorithm

> The "tableau" algorithm

- The main steps of the algorithm are (for $\mathbf{V} \vDash_{\mathcal{L}} \boldsymbol{W}$):

  1. ***W*** is simplified (brought to "normal form", much like a normal form in predicate logic)
  2. a new vocabulary ***U*** is defined, combining ***V*** and the *negation* of ***W*** such that $\boldsymbol{V} \vDash_{\mathcal{L}} \boldsymbol{W}$ is equivalent with ***U*** *not* being consistent (a form of "reductio ad absurdum")
  3. the algorithm tries to construct an interpretation for ***U***, taking into account all possibilities. If:
     1. it succeeds, then ***U*** is consistent, i.e., $\boldsymbol{V} \vDash_{\mathcal{L}} \boldsymbol{W}$ does *not* hold
     2. if it fails then ***U*** is not consistent, ie, $\boldsymbol{V} \vDash_{\mathcal{L}} \boldsymbol{W}$ holds

- The complexity of the algorithm depends on the expressiveness of the language (mainly in covering all possibilities when constructing an interpretation)

> So what is OWL-DL and OWL-Lite?

- *OWL-DL and OWL-Lite are the syntactic equivalents of Description Logic types*
  - $\mathcal{SHIN}(\mathcal{D})$ and $\mathcal{SHIF}(\mathcal{D})$, respectively, where $\mathcal{D}$ refers to a datatype (XML Schemas in this case)

> So what is OWL-DL and OWL-Lite? (cont.)

- This explains the restrictions in using OWL predicates and classes:
  - strict separation of **`owl:Thing`** and **`owl:Class`** (corresponding to individuals and classes in Description Logic)
  - restricted usage of **`rdf:type`**, **`rdf:subClassOf`**, etc, predicates (they do *not* correspond to general roles in Description Logic)
  - separation of object properties (corresponding to roles) and datatype properties
- It also shows the origin of some of the OWL constructions (eg, creating a class via property restrictions)

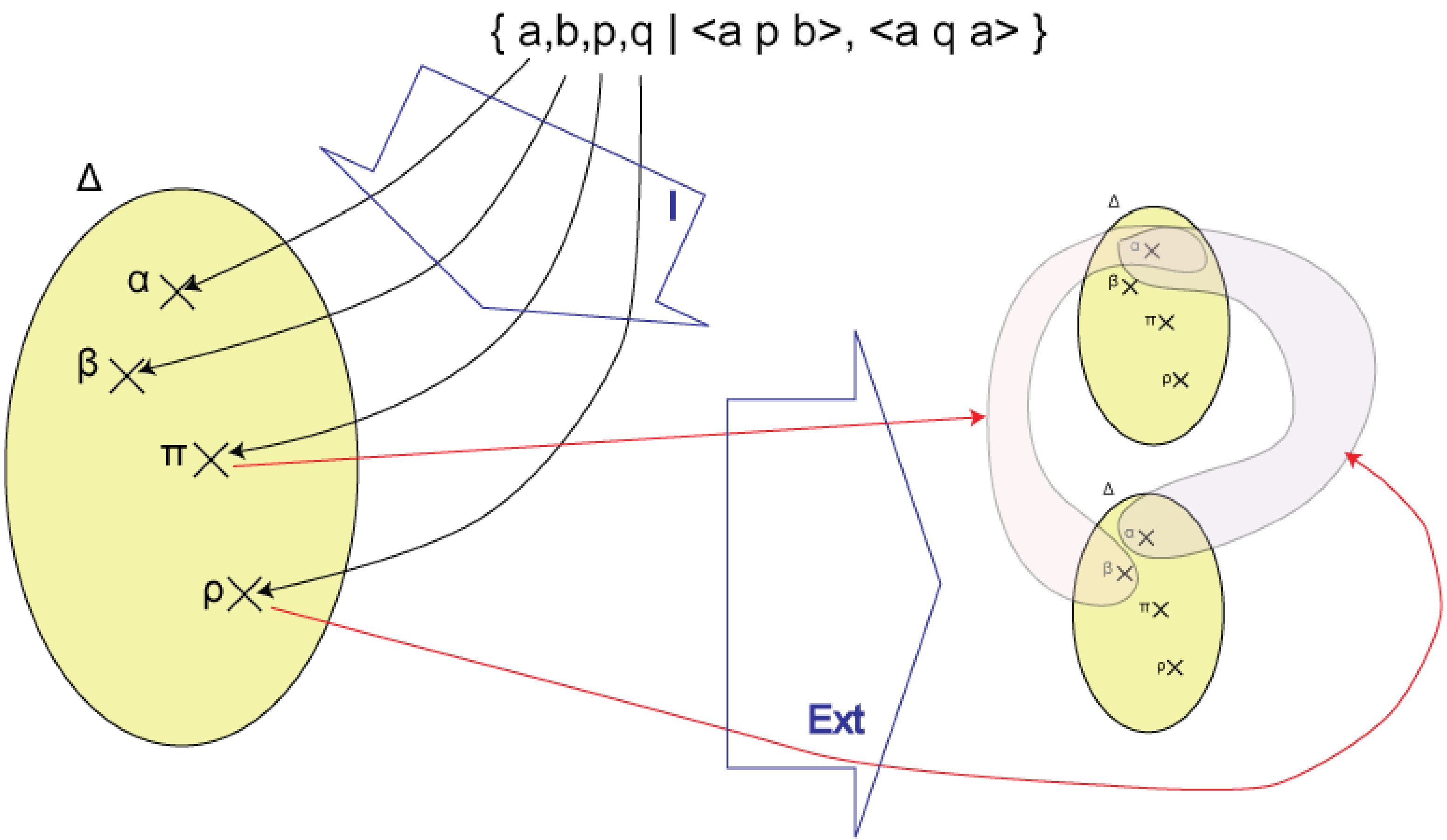
> What about RDFS and OWL-Full?

- The situation is more complicated…
  - the interpretations so far mapped individuals on elements of a set, classes on subsets, roles on binary relations
  - in RDFS, there is no such strict separation: one may make statements on properties, have class of classes, etc.
  - consequence: *RDFS (and OWL-Full) requires a more complex interpretation than the ones so far*
  - if such interpretation can be found, the notion of inference, equivalence, consistency, etc, are similar

> Graph interpretations

- The interpretation for an RDF graph is based on extension functions:
  - properties (predicates) are *all* part of the vocabulary (in contrast to Description Logic)
  - an interpretation has two steps:
    1. all elements of the vocabulary are mapped by a mapping *I* on a set Δ,
       - i.e., predicates are mapped on elements, too
    2. a separate extension function is defined (**Ext**), that for each predicate maps to $\Delta \times \Delta$
  - extra constraints on the interpretation ensure a “proper” mapping of predicates

> Graph interpretations (cont.)



> RDF Interpretations

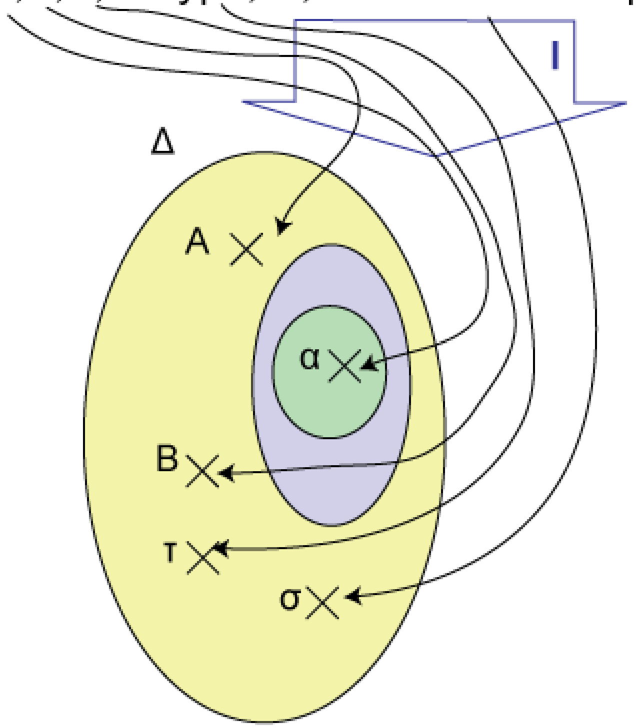
- RDF interpretation are graph interpretations, with extra restrictions:
  - all vocabularies must include certain properties (**`rdf:type`**, **`rdf:Property`**, **`rdf:first`**, **`rdf:rest`**, etc)
  - a number of "axiomatic" triples should also be part of the vocabularies (e.g., **`<rdf:type rdf:type rdf:Property>`**)
  - the semantics of these should be reflected in the interpretation, e.g.:
    - if *p* is a property in the RDF Graph, then: $\langle I(p), I(\texttt{rdf:Property})\rangle \in \mathbf{Ext}(I(\texttt{rdf:type}))$
    - if $\langle a,p,c\rangle$ is in the Graph, then $\langle I(a),I(c)\rangle \in \mathbf{Ext}(I(p))$,
    - Etc.
- This defines a clear semantics and consistency for RDF

> RDFS Interpretations

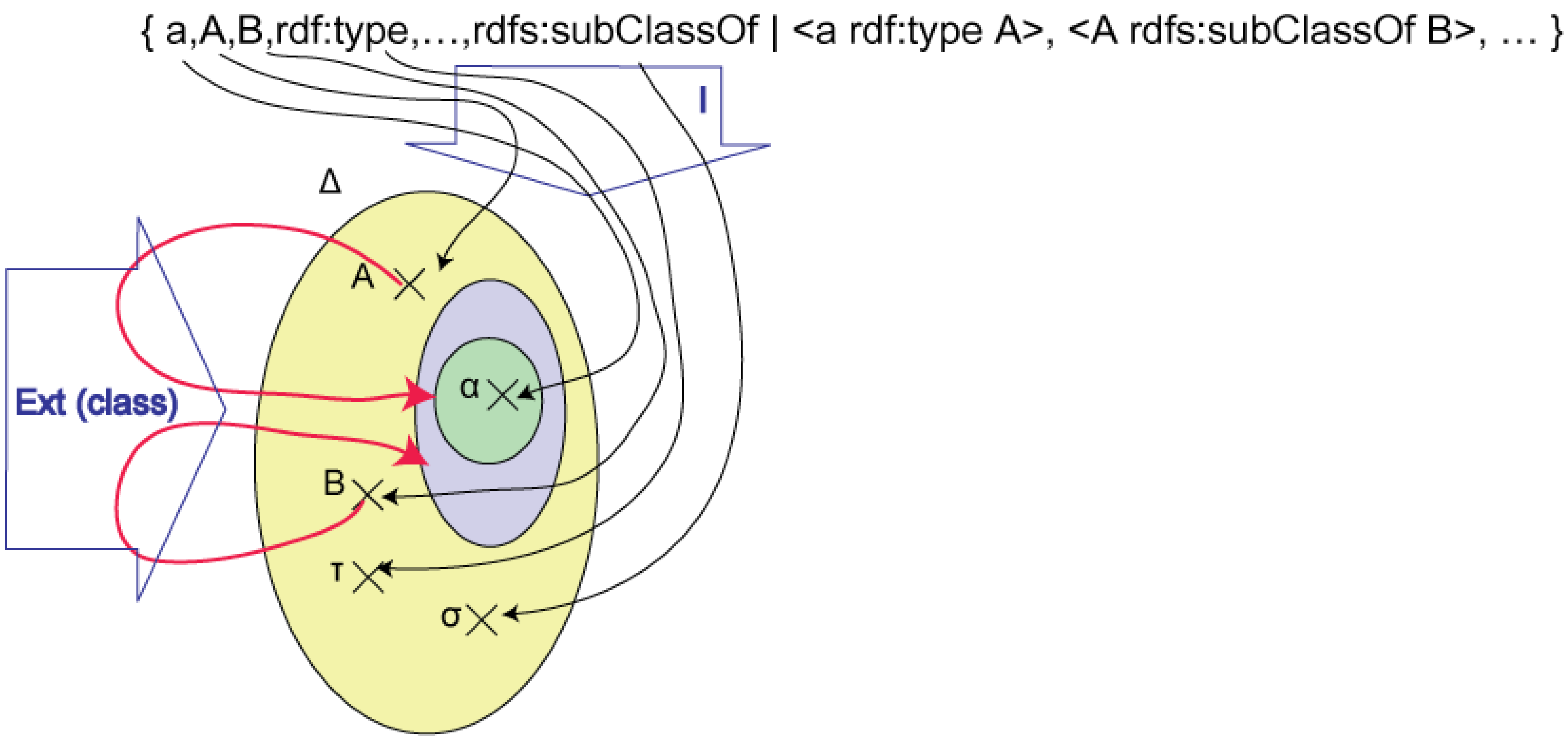
- Much like RDF interpretations, but with some extra complications, e.g.:
  - an extra extension function must be defined for classes: each class is mapped to a subset of Δ by this extension
  - the vocabulary must include all the other properties defined in RDFS (**`rdfs:subClassOf`**, **`rdfs:range`**, **`rdfs:domain`**, etc)
  - more "axiomatic" triples must be added to the vocabulary (e.g., **`<rdf:type rdfs:range rdfs:Class>`**)
  - the interpretation and extension functions should reflect the extra semantics of all those, e.g., if <*A* **`rdfs:subClassOf`** *B*> then **Ext**(*I(A)*) ⊂ **Ext**(*I(B)*)
- It is a lot of details to handle, and must be done with care, but it is fairly mechanical…
- Typed literals bring in an extra level of complications

> RDFS Interpretations

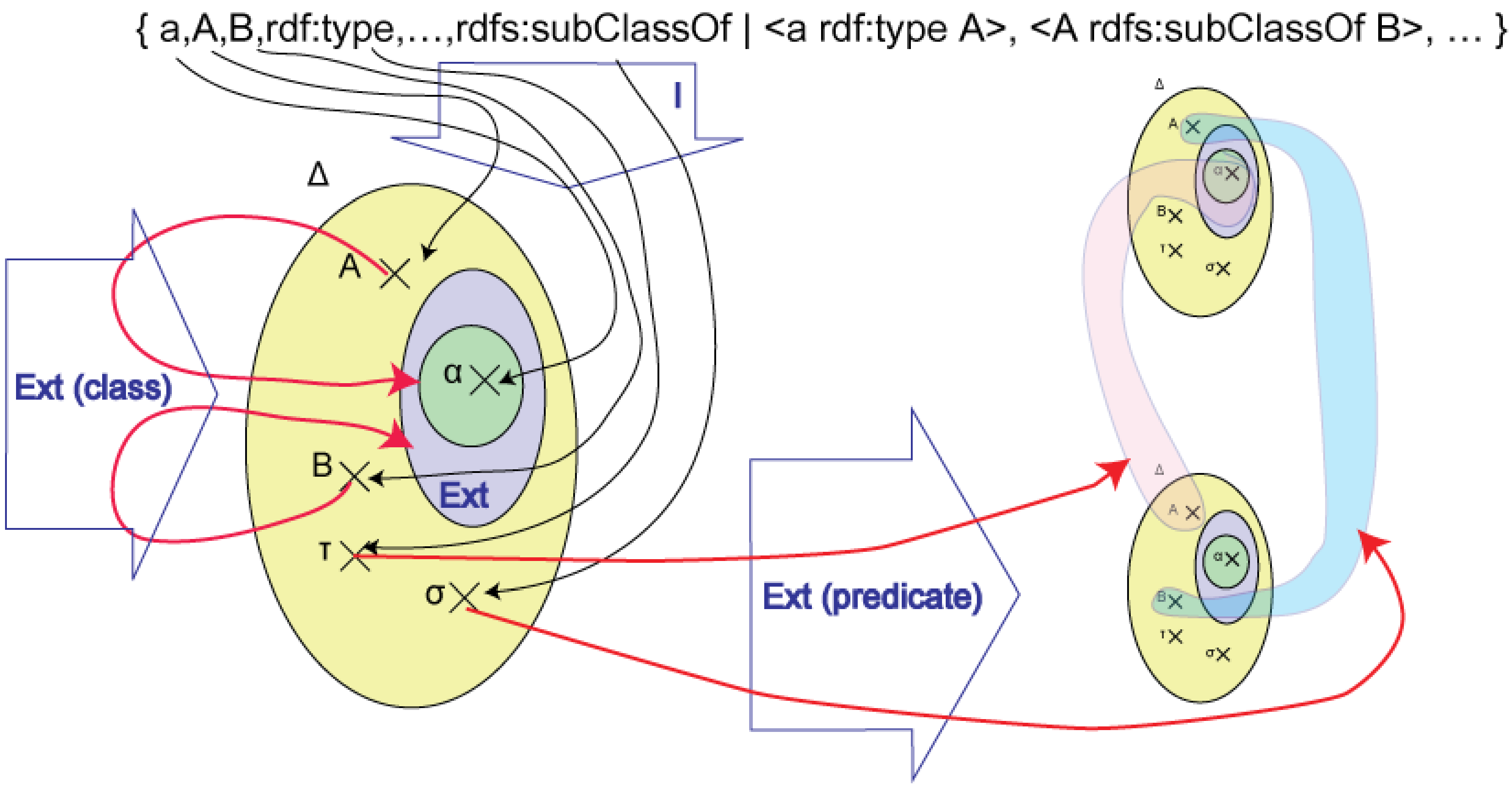
{ a,A,B,rdf:type,…,rdfs:subClassOf | <a rdf:type A>, <A rdfs:subClassOf B>, … }



> RDFS Interpretations



> RDFS Interpretations



> RDFS Inference, Consistency, …

- Using the RDFS Interpretations, the notion of consistency, inference, etc, follows the general model theory definitions
- The “RDF Semantics” document also includes an algorithm for $R \vDash_{\mathrm{RDFS}} S$:
  - extend R with new triplets recursively as long as there is a change, using a set of “entailement rules”; this yields *E(R)*
  - $R \vDash_{\mathrm{RDFS}} S$ if and only if *S* is a subgraph of *E(R)*
    (there are some extra complications with typed literals and blank nodes)
  - it can be proven that this algorithm works
- In the algorithmic world, this is referred to as “forward chaining”

> Entailment rule

- An example for an entailment rule in RDFS:

```
If:
  uuu rdfs:subClassOf xxx .
  vvv rdf:type uuu .
Then add:
  vvv rdf:type xxx .
```

> Finally: OWL-Full

- The semantics of OWL-Full can be constructed similarly to RDFS
  - but it gets *much* more complex…
- Whether an inference can be decided for OWL-Full is unknown
  - one of the source of complication: the RDFS/OWL-Full type of interpretation treats the “core” RDFS terms (eg, type, class) similarly to the users’
    - eg, one can define rdf:type to be functional…
- There is research going on to define subsets of OWL-Full that can be combined with the basic RDFS semantics (i.e., not in direction of Description Logic), still yielding decidable logic families
  - see, e.g., H.J. ter Horst’s paper at ISWC2004 or in the Journal of Web Semantics (October 2005)