Knowledge Representation for the Semantic Web

Winter Quarter 2012

Slides 4 – 01/12/2012

Pascal Hitzler
Kno.e.sis Center
Wright State University, Dayton, OH
http://www.knoesis.org/pascal/

Textbook (required)

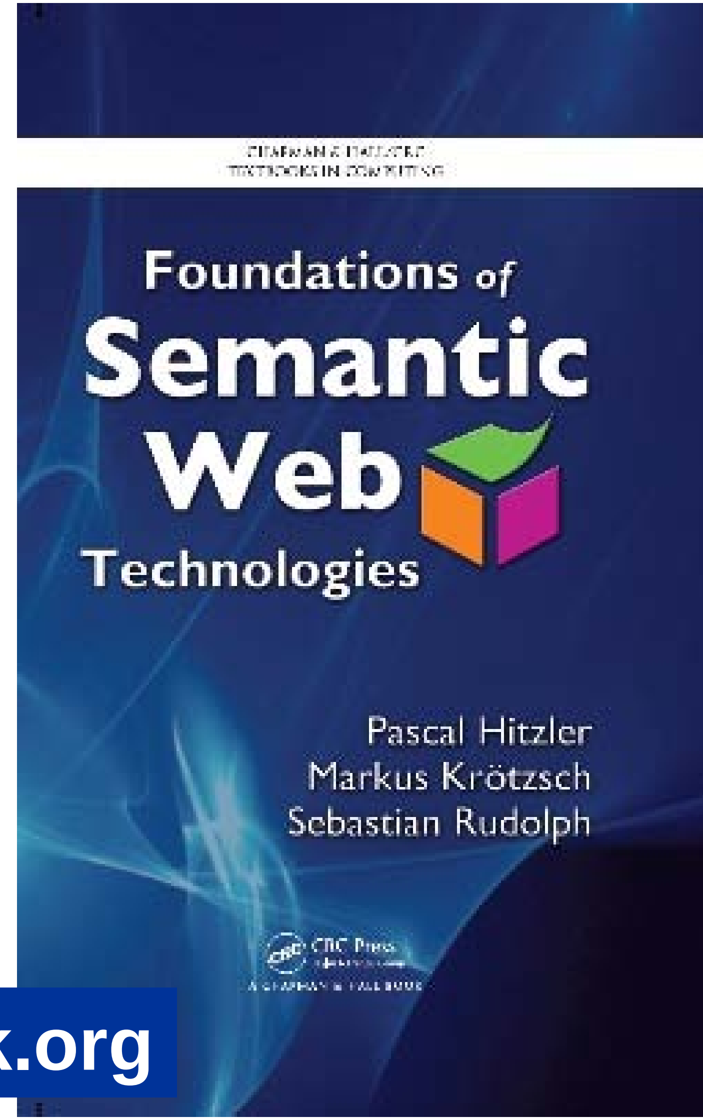
KNO.E.SIS

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

Foundations of Semantic Web Technologies

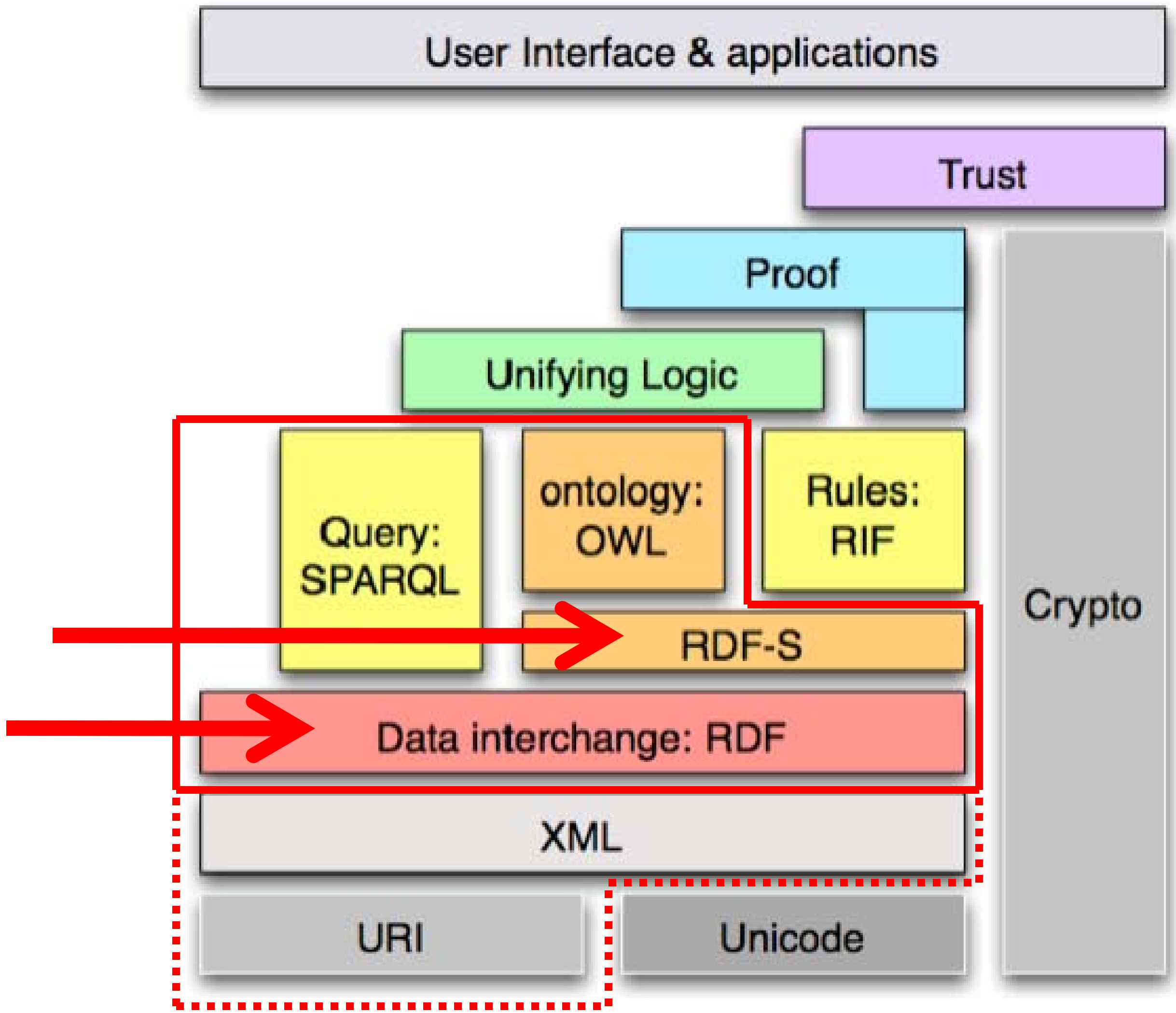
Chapman & Hall/CRC, 2010

Choice Magazine Outstanding Academic Title 2010 (one out of seven in Information & Computer Science)



http://www.semantic-web-book.org

Today: RDF(S) semantics



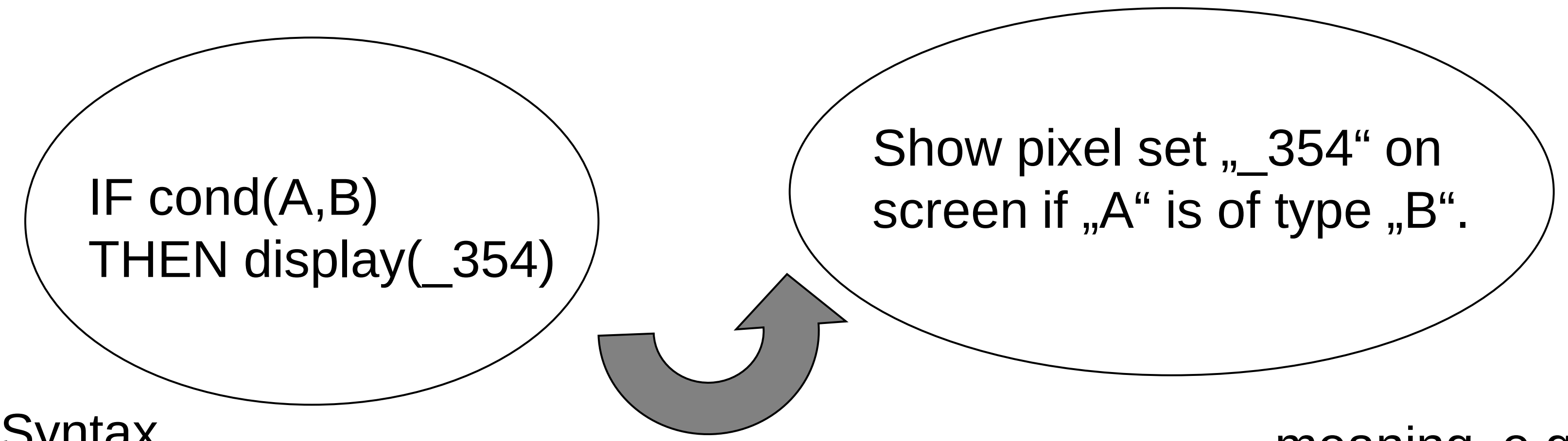
Today's Session: RDF(S) semantics

1. What is Semantics?
2. What is Model-theoretic Semantics?
3. Model-theoretic Semantics for RDF(S)
4. What is Proof-theoretic Semantics?
5. Proof-theoretic Semantics for RDF(S)
6. Class Project
7. Class Presentations

Syntax and Semantics

Syntax: character strings without meaning

Semantics: meaning of the character strings

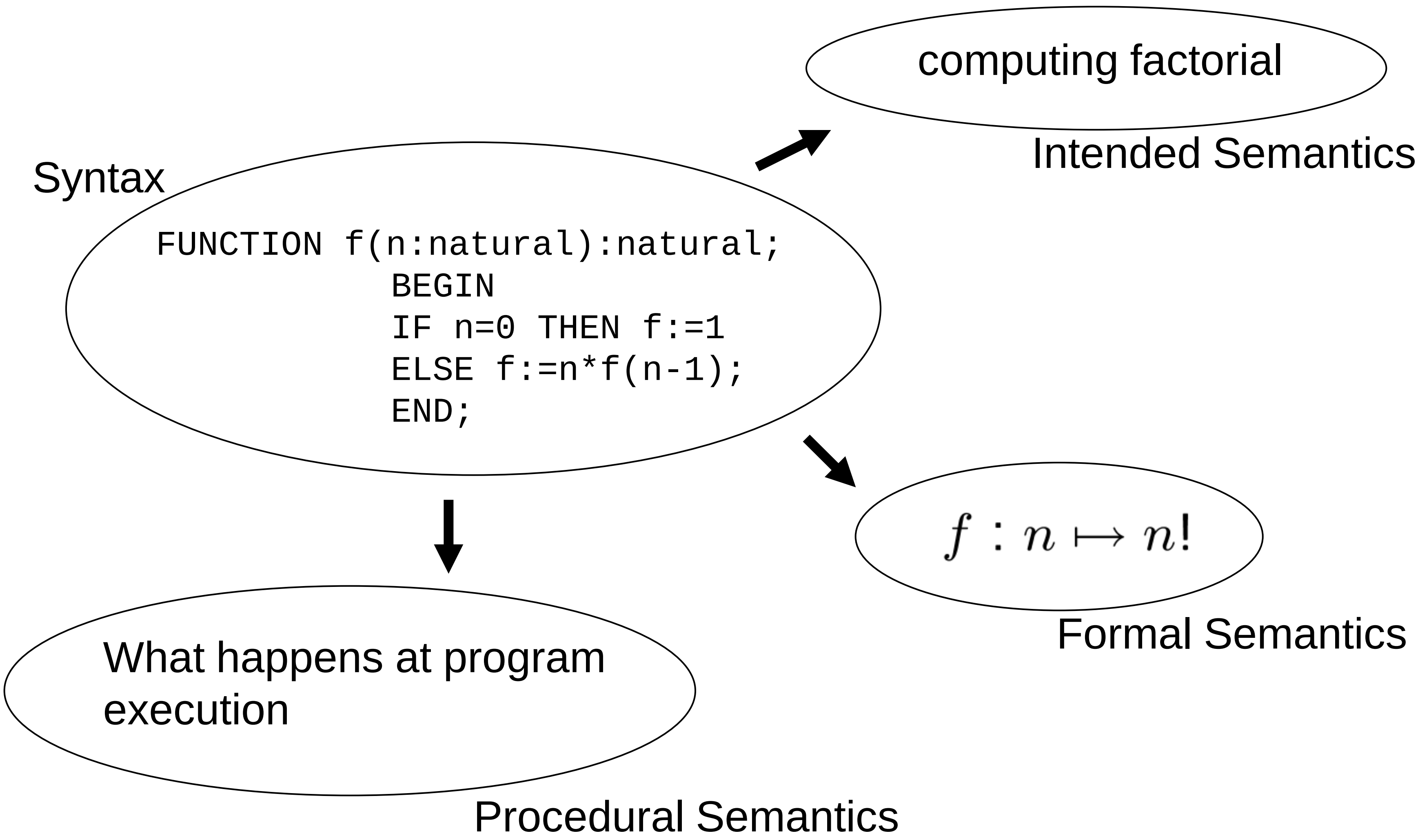


Syntax

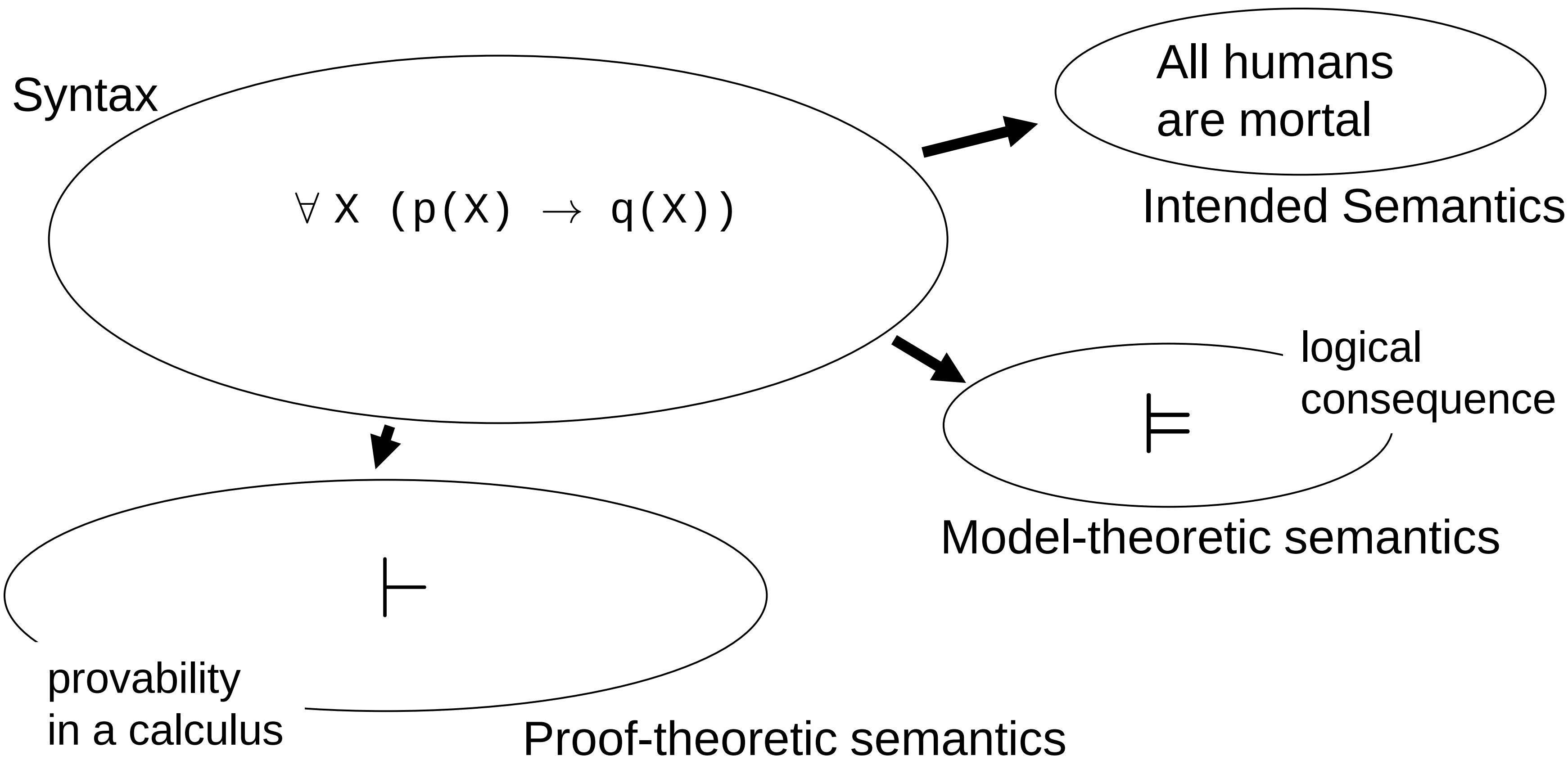
assignment of meaning

meaning, e.g.,
„in the world“

Semantics of Programming Languages



Semantics of Logic



Recall: Implicit knowledge

- **if an RDFS document contains**

```
u   rdf:type   ex:Textbook .
```

and

```
ex:Textbook   rdfs:subClassOf   ex:Book .
```

then

```
u   rdf:type   ex:Book .
```

is *implicitly* also the case: it's a *logical consequence*. (We can also say it is *deduced* (deduction) or *inferred* (inference).
We do not have to state this explicitly.
Which statements are logical consequences is governed by the formal semantics (covered in the next session).

Recall: Implicit knowledge

- **From**

```
ex:Textbook   rdfs:subClassOf   ex:Book .
ex:Book       rdfs:subClassOf   ex:PrintMedia .
```

the following is a logical consequence:

```
ex:Textbook   rdfs:subClassOf   ex:PrintMedia .
```

I.e. rdfs:subClassOf is *transitive*.

What Semantics Is Good For

- **Opinions Differ. Here's my take.**

- **Semantic Web requires a shareable, declarative and *computable* semantics.**
- **I.e., the semantics must be a formal entity which is clearly defined and automatically computable.**

- **Ontology languages provide this by means of their formal semantics.**
- **Semantic Web Semantics is given by a relation – the *logical consequence* relation.**

In other words

We capture the meaning of information

not by specifying its meaning (which is impossible)
but by specifying

how information interacts with other information.

We describe the meaning indirectly through its effects.

Today's Session: RDF(S) semantics

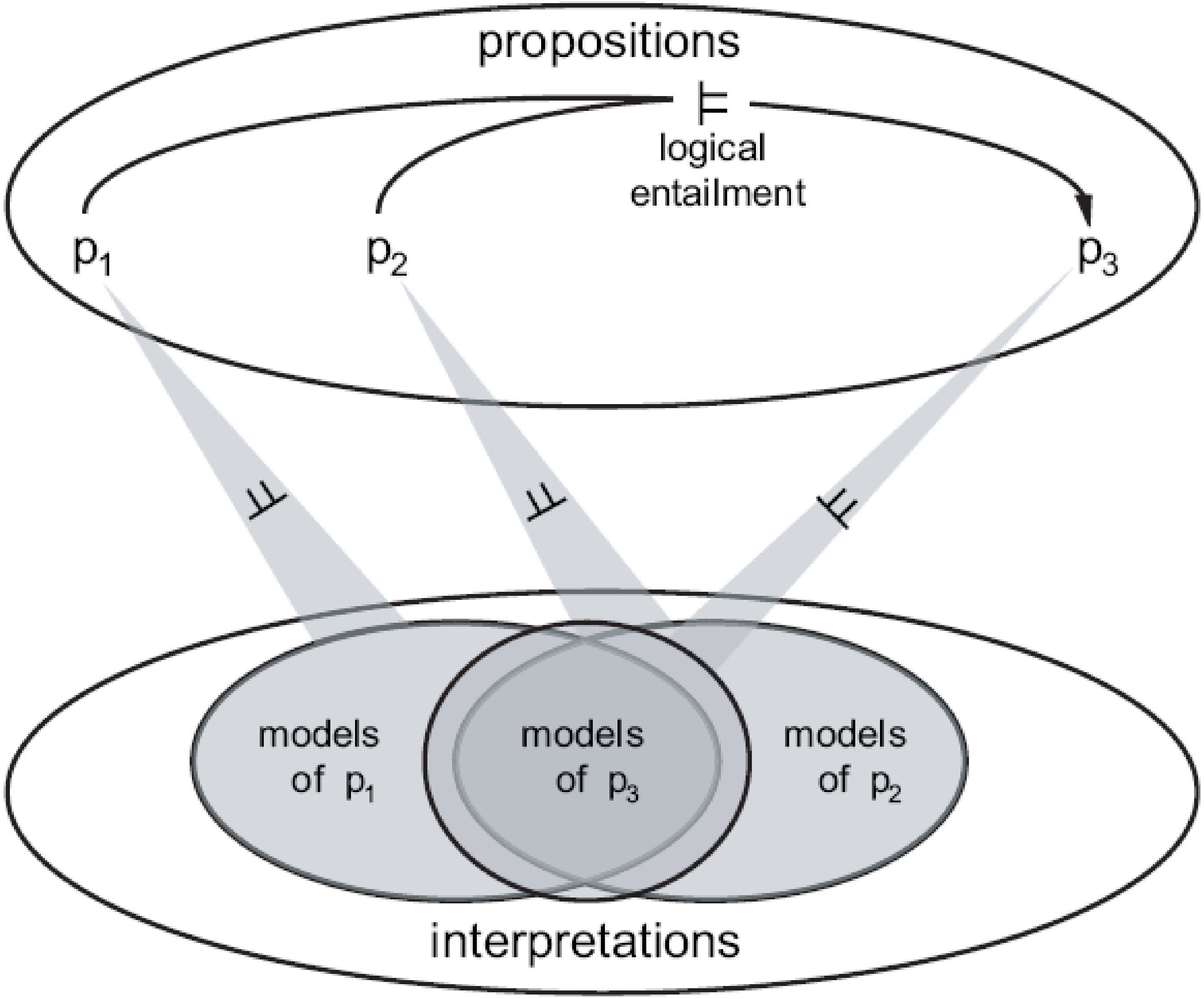
1. What is Semantics?
2. What is Model-theoretic Semantics?
3. Model-theoretic Semantics for RDF(S)
4. What is Proof-theoretic Semantics?
5. Proof-theoretic Semantics for RDF(S)
6. Class Project
7. Class Presentations

Model-theoretic Semantics

- You need:
  - a language/syntax
  - a notion of *model* for sentences in the language
- Models
  - are made such that each sentence is either true or false in each model
  - If a sentence α is true in a model M, then we write M⊨α
- Logical consequence:
  - β is a logical consequence of α (written α⊨β), if for all M with M⊨α, we also have M⊨β
  - If K is a set of sentences, we write K⊨β if M⊨β for each M⊨K
  - If J is another set of sentences, we write K⊨J if K⊨β for each $\beta \in$ J

  (note that the notation ⊨ is overloaded)

Logical Consequence



Model theory (contrived) example

- **Language:**
  variables ...,w,x,y,z,...
  symbol η
  allowed sentences: $a \eta b$ (for a, b any variables)

- **We want to know:**

  What are the logical consequences of the set

  $\{x \eta y, y \eta z\}$

- **To answer this, we must say what the models in our semantics are.**

Model theory (contrived) example

- **Say, a model I of a set K of sentences consists of**
  - **a set C of cars and**
  - **a function I(·) which maps each variable to a car in C**

  such that, for each sentence a η b in K we have that

  I(a) has more horsepower than I(b).

- **We now claim that {x η y, y η z} ⊨ x η z.**
- **Proof: Consider any model M of {x η y, y η z}.**
  Since M⊨ {x η y, y η z}, we know that

  M(x) has more horsepower than M(y) and
  M(y) has more horsepower than M(z).

  Hence, M(x) has more horsepower than M(z), i.e. M⊨ x η z.

  This argument holds for all models of {x η y, y η z}, therefore

  {x η y, y η z} ⊨ x η z.

Model theory (contrived) example

- Say, a model I of a set K of sentences consists of
  - a set C of cars and
  - a function I(·) which maps each variable to a car in C

  such that, for each sentence a η b in K we have that I(a) has more horsepower than I(b).

- **An *interpretation* I for a our language consists of**
  - **a set C of cars and**
  - **a function I(·) which maps each variable to a car in C.**

(and that's it, i.e. no information whether a sentence is true or false with respect to I).

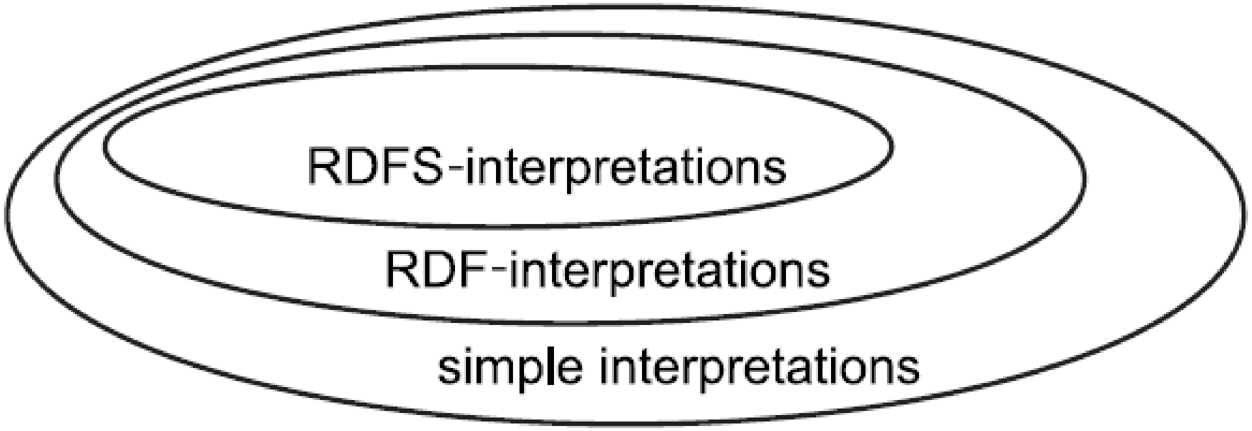
Today's Session: RDF(S) semantics

1. What is Semantics?
2. What is Model-theoretic Semantics?
3. Model-theoretic Semantics for RDF(S)
4. What is Proof-theoretic Semantics?
5. Proof-theoretic Semantics for RDF(S)
6. Class Project
7. Class Presentations

Now let's do this for RDF(S)

- Language: Whatever is valid RDF(S).
- Sentences are triples. (Graphs are sets of triples.)

- Interpretations are given via sets and functions from language vocabularies to these sets.
- Models are defined such that they capture the intended meaning of the RDF(S) vocabulary.
- And there are three different notions:



Simple Interpretations

So we define: a *simple interpretation* $\mathcal{I}$ of a given vocabulary V consists of

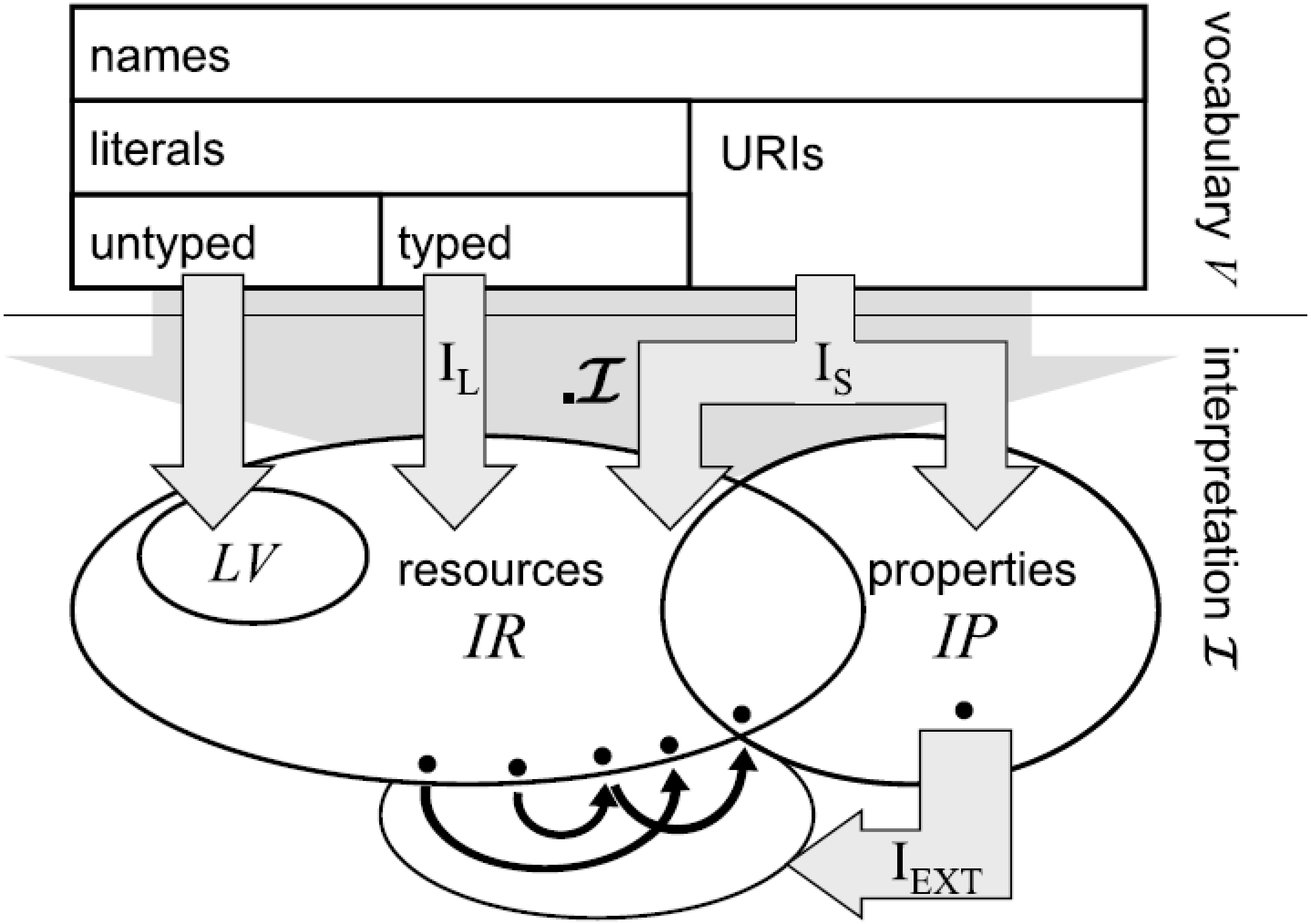
- IR, a non-empty set of *resources*, alternatively called domain or universe of discourse of $\mathcal{I}$,
- IP, the set of *properties* of $\mathcal{I}$ (which may overlap with IR),
- $\mathrm{I}_{\mathrm{EXT}}$, a function assigning to each property a set of pairs from IR, i.e. $\mathrm{I}_{\mathrm{EXT}} : IP \to 2^{IR \times IR}$, where $\mathrm{I}_{\mathrm{EXT}}(p)$ is called the *extension* of the property p,
- I_{S}, a function, mapping URIs from V into the union of the sets IR and IP, i.e. $\mathrm{I}_{\mathrm{S}} : V \to IR \cup IP$,
- I_{L}, a function from the typed literals from V into the set IR of resources and
- LV, a particular subset of IR, called the set of *literal values*, containing (at least) all untyped literals from V.

Simple Interpretations

Now define an interpretation function $\cdot^{\mathcal{I}}$ (written as exponent).

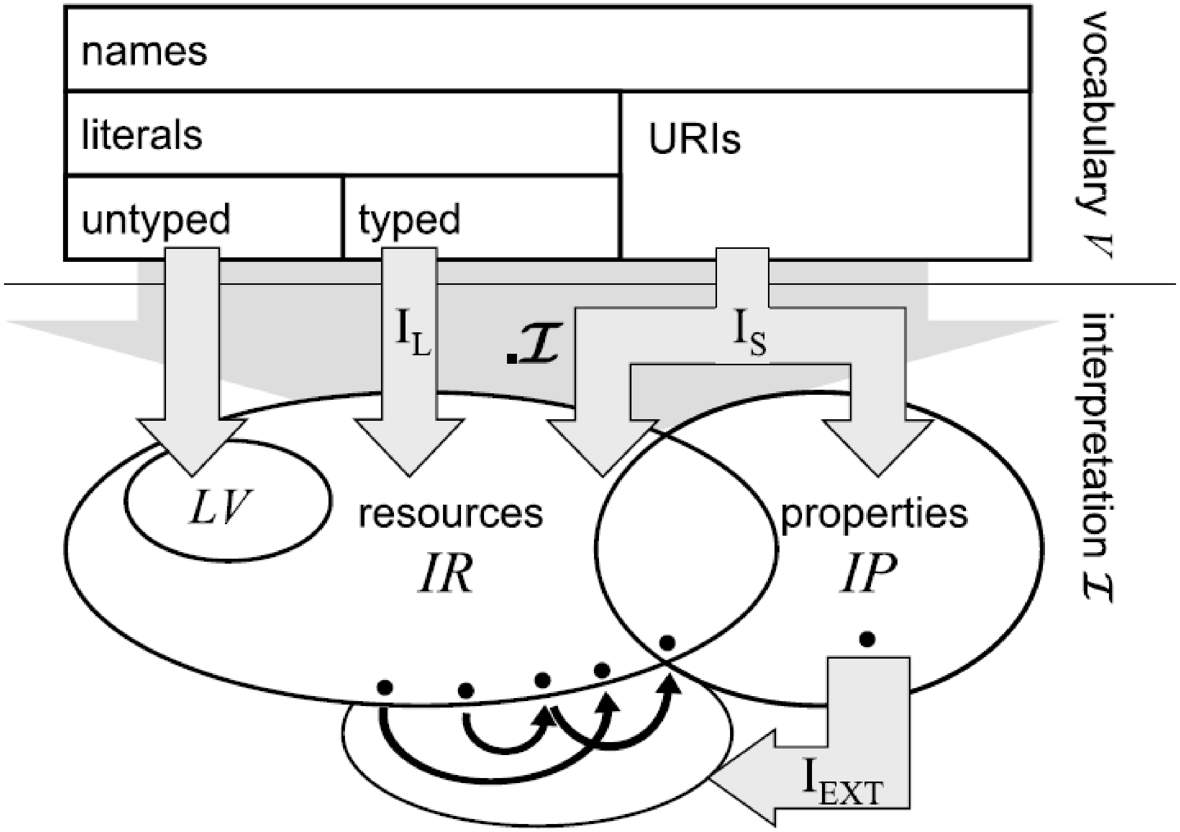
- every untyped literal "a" is mapped to a, formally: $(\texttt{"}a\texttt{"})^{\mathcal{I}} = a$,
- every untyped literal carrying language information "a"@t is mapped to the pair $\langle a, t\rangle$, i.e. $(\texttt{"}a\texttt{"@}t)^{\mathcal{I}} = \langle a, t\rangle$,
- every typed literal l is mapped to $\mathrm{I_L}(l)$, formally: $l^{\mathcal{I}} = \mathrm{I_L}(l)$, and
- every URI u is mapped to $\mathrm{I_S}(u)$, i.e. $u^{\mathcal{I}} = \mathrm{I_S}(u)$.

Simple Interpretations



Simple models

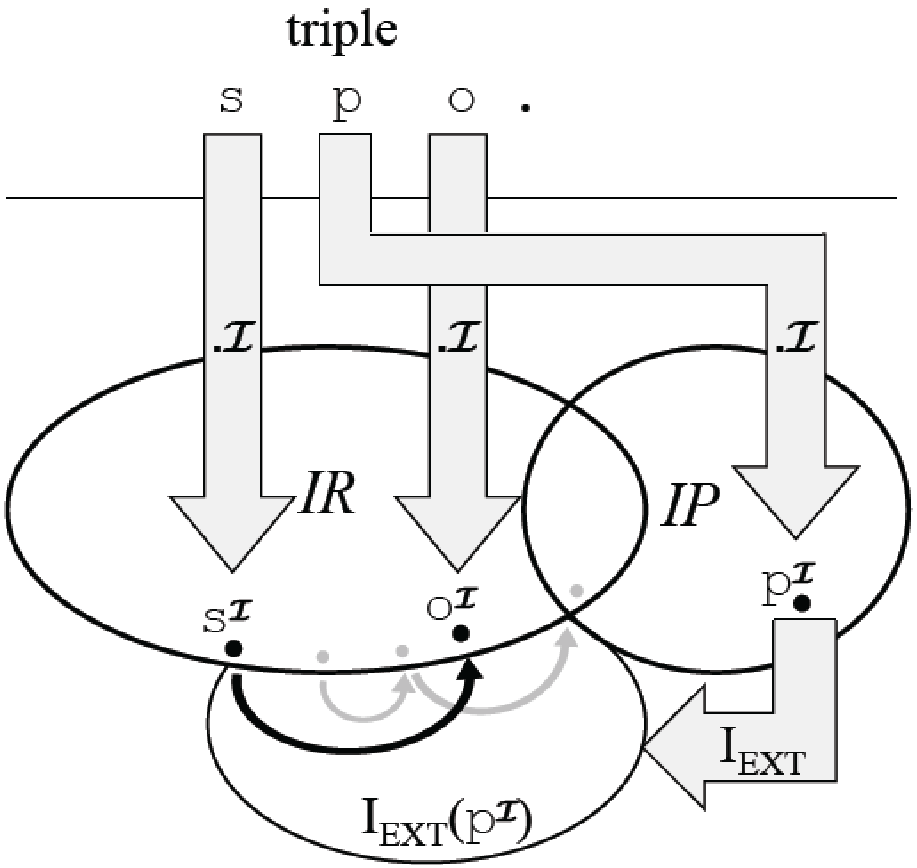
- The truth value $\texttt{s p o.}^{\mathcal{I}}$ of a (grounded*) triple $\texttt{s p o.}$ is true exactly if ($\texttt{s}$, $\texttt{p}$, $\texttt{o}$ are contained in V) and $\langle \texttt{s}^{\mathcal{I}}, \texttt{o}^{\mathcal{I}} \rangle \in \mathrm{I}_{\mathrm{EXT}}(\texttt{p}^{\mathcal{I}})$.



* A grounded triple does not contain a blank node.

Simple models

- The truth value $\texttt{s p o.}^{\mathcal{I}}$ of a (grounded*) triple $\texttt{s p o.}$ is true exactly if ($\texttt{s}$, $\texttt{p}$, $\texttt{o}$ are contained in V) and $\langle \texttt{s}^{\mathcal{I}}, \texttt{o}^{\mathcal{I}} \rangle \in \mathrm{I}_{\mathrm{EXT}}(\texttt{p}^{\mathcal{I}})$.



* A grounded triple does not contain a blank node.

What about blank nodes?

- **Say, A is a function from blank nodes to URIs.
[these URIs need not be contained in the graph we’re looking at]**
- **If, in a graph G, we replace each blank node x by A(x), then we obtain a graph G’ which we call a *grounding* of G.**
- **We know how to do the semantics for the grounded graphs.**
- **So define:
I ⊨ G if and only if I ⊨ G’ for at least one grounding G’ of G.**

Simple entailment

- **A graph G *simply entails* a graph G' if every simple interpretation that is a model of G is also a model of G'.**

- **(Recall that a simple interpretation is a model of a graph G if it is a model of each triple in G.)**

It’s really simple

- **Basically, G $\models$ G’ if and only if G’ can be obtained from G by replacing some nodes in G by blank nodes.**

- **It’s really *simple* entailment.**

RDF-Interpretations Part 1

An *RDF-interpretation* of a vocabulary V is a simple interpretation of the vocabulary $V \cup V_{\mathrm{RDF}}$ that additionally satisfies the following conditions:

- $x \in IP$ exactly if $\langle x, \texttt{rdf:Property}^{\mathcal{I}} \rangle \in \mathrm{I}_{\mathrm{EXT}}(\texttt{rdf:type}^{\mathcal{I}})$.
- if $\texttt{"}s\texttt{"\^{}\^{}rdf:XMLLiteral}$ is contained in V and s is a well-typed XML-Literal, then
  - $\mathrm{I_L}(\texttt{"}s\texttt{"\^{}\^{}rdf:XMLLiteral})$ is the XML value of s;
  - $\mathrm{I_L}(\texttt{"}s\texttt{"\^{}\^{}rdf:XMLLiteral}) \in LV$;
  - $\langle \mathrm{I_L}(\texttt{"}s\texttt{"\^{}\^{}rdf:XMLLiteral}), \texttt{rdf:XMLLiteral}^{\mathcal{I}} \rangle \in \mathrm{I}_{\mathrm{EXT}}(\texttt{rdf:type}^{\mathcal{I}})$
- if $\texttt{"}s\texttt{"\^{}\^{}rdf:XMLLiteral}$ is contained in V and s is an ill-typed XML literal, then
  - $\mathrm{I_L}(\texttt{"}s\texttt{"\^{}\^{}rdf:XMLLiteral}) \notin LV$ and
  - $\langle \mathrm{I_L}(\texttt{"}s\texttt{"\^{}\^{}rdf:XMLLiteral}), \texttt{rdf:XMLLiteral}^{\mathcal{I}} \rangle \notin \mathrm{I}_{\mathrm{EXT}}(\texttt{rdf:type}^{\mathcal{I}})$.

RDF-Interpretations Part 2

- **In addition, each RDF-interpretation has to evaluate all the following triples to true:**

$$
\begin{array}{lll}
\mathtt{rdf:type} & \mathtt{rdf:type} & \mathtt{rdf:Property.} \\
\mathtt{rdf:subject} & \mathtt{rdf:type} & \mathtt{rdf:Property.} \\
\mathtt{rdf:predicate} & \mathtt{rdf:type} & \mathtt{rdf:Property.} \\
\mathtt{rdf:object} & \mathtt{rdf:type} & \mathtt{rdf:Property.} \\
\mathtt{rdf:first} & \mathtt{rdf:type} & \mathtt{rdf:Property.} \\
\mathtt{rdf:rest} & \mathtt{rdf:type} & \mathtt{rdf:Property.} \\
\mathtt{rdf:value} & \mathtt{rdf:type} & \mathtt{rdf:Property.} \\
\mathtt{rdf:\_}i & \mathtt{rdf:type} & \mathtt{rdf:Property.} \\
\mathtt{rdf:nil} & \mathtt{rdf:type} & \mathtt{rdf:List.}
\end{array}
$$

RDFS-Interpretations Part 1

- **Define (for a given RDF-interpretation $\mathcal{I}$):**
  - $\mathrm{I_{CEXT}} : IR \to 2^{IR}$ **:** We define $\mathrm{I_{CEXT}}(y)$ to contain exactly those elements x for which $\langle x, y\rangle$ is contained in $\mathrm{I_{EXT}}(\mathtt{rdf:type}^{\mathcal{I}})$. The set $\mathrm{I_{CEXT}}(y)$ is then also called the *(class) extension* of y.
  - $IC = \mathrm{I_{CEXT}}(\mathtt{rdfs:Class}^{\mathcal{I}})$.
- $IR = \mathrm{I_{CEXT}}(\mathtt{rdfs:Resource}^{\mathcal{I}})$
- $LV = \mathrm{I_{CEXT}}(\mathtt{rdfs:Literal}^{\mathcal{I}})$
- If $\langle x, y\rangle \in \mathrm{I_{EXT}}(\mathtt{rdfs:domain}^{\mathcal{I}})$ and $\langle u, v\rangle \in \mathrm{I_{EXT}}(x)$, then $u \in \mathrm{I_{CEXT}}(y)$.
- If $\langle x, y\rangle \in \mathrm{I_{EXT}}(\mathtt{rdfs:range}^{\mathcal{I}})$ and $\langle u, v\rangle \in \mathrm{I_{EXT}}(x)$, then $v \in \mathrm{I_{CEXT}}(y)$.
- $\mathrm{I_{EXT}}(\mathtt{rdfs:subPropertyOf}^{\mathcal{I}})$ is reflexive and transitive on IP.

RDFS-Interpretation Part 2

- If $\langle x, y\rangle \in \mathrm{I_{EXT}}(\texttt{rdfs:subPropertyOf}^{\mathcal{I}})$,
  then $x, y \in IP$ and $\mathrm{I_{EXT}}(x) \subseteq \mathrm{I_{EXT}}(y)$.
- If $x \in IC$,
  then $\langle x, \texttt{rdfs:Resource}^{\mathcal{I}}\rangle \in \mathrm{I_{EXT}}(\texttt{rdfs:subClassOf}^{\mathcal{I}})$.
- If $\langle x, y\rangle \in \mathrm{I_{EXT}}(\texttt{rdfs:subClassOf}^{\mathcal{I}})$,
  then $x, y \in IC$ and $\mathrm{I_{CEXT}}(x) \subseteq \mathrm{I_{CEXT}}(y)$.
- $\mathrm{I_{EXT}}(\texttt{rdfs:subClassOf}^{\mathcal{I}})$ is reflexive and transitive on IC.
- If $x \in \mathrm{I_{CEXT}}(\texttt{rdfs:ContainerMembershipProperty}^{\mathcal{I}})$,
  then $\langle x, \texttt{rdfs:member}^{\mathcal{I}}\rangle \in \mathrm{I_{EXT}}(\texttt{rdfs:subPropertyOf}^{\mathcal{I}})$.
- If $x \in \mathrm{I_{CEXT}}(\texttt{rdfs:Datatype}^{\mathcal{I}})$,
  then $\langle x, \texttt{rdfs:Literal}^{\mathcal{I}}\rangle \in \mathrm{I_{EXT}}(\texttt{rdfs:subClassOf}^{\mathcal{I}})$

RDFS-Interpretations Part 3

- **Furthermore, all of the following must be satisfied.**

```
rdf:type             rdfs:domain     rdfs:Resource .
rdfs:domain          rdfs:domain     rdf:Property .
rdfs:range           rdfs:domain     rdf:Property .
rdfs:subPropertyOf   rdfs:domain     rdf:Property .
rdfs:subClassOf      rdfs:domain     rdfs:Class .
rdf:subject          rdfs:domain     rdf:Statement .
rdf:predicate        rdfs:domain     rdf:Statement .
rdf:object           rdfs:domain     rdf:Statement .
rdfs:member          rdfs:domain     rdfs:Resource .
rdf:first            rdfs:domain     rdf:List .
rdf:rest             rdfs:domain     rdf:List .
rdfs:seeAlso         rdfs:domain     rdfs:Resource .
rdfs:isDefinedBy     rdfs:domain     rdfs:Resource .
```

RDFS-Interpretations Part 4

- **Furthermore, all of the following must be satisfied.**

```
rdfs:comment        rdfs:domain   rdfs:Resource .
rdfs:label          rdfs:domain   rdfs:Resource .
rdf:value           rdfs:domain   rdfs:Resource .

rdf:type            rdfs:range    rdfs:Class .
rdfs:domain         rdfs:range    rdfs:Class .
rdfs:range          rdfs:range    rdfs:Class .
rdfs:subPropertyOf  rdfs:range    rdf:Property .
rdfs:subClassOf     rdfs:range    rdfs:Class .
rdf:subject         rdfs:range    rdfs:Resource .
rdf:predicate       rdfs:range    rdfs:Resource .
rdf:object          rdfs:range    rdfs:Resource .
rdfs:member         rdfs:range    rdfs:Resource .
rdf:first           rdfs:range    rdfs:Resource .
rdf:rest            rdfs:range    rdf:List .
rdfs:seeAlso        rdfs:range    rdfs:Resource .
rdfs:isDefinedBy    rdfs:range    rdfs:Resource .
rdfs:comment        rdfs:range    rdfs:Literal .
rdfs:label          rdfs:range    rdfs:Literal .
rdf:value           rdfs:range    rdfs:Resource .
```

RDFS-Interpretations Part 5

- **Furthermore, all of the following must be satisfied.**

```
rdfs:ContainerMembershipProperty
                         rdfs:subClassOf      rdf:Property .
rdf:Alt                  rdfs:subClassOf      rdfs:Container .
rdf:Bag                  rdfs:subClassOf      rdfs:Container .
rdf:Seq                  rdfs:subClassOf      rdfs:Container .

rdfs:isDefinedBy         rdfs:subPropertyOf   rdfs:seeAlso .

rdf:XMLLiteral           rdf:type             rdfs:Datatype .
rdf:XMLLiteral           rdfs:subClassOf      rdfs:Literal .
rdfs:Datatype            rdfs:subClassOf      rdfs:Class .

rdf:_i                   rdf:type
                           rdfs:ContainerMembershipProperty .
rdf:_i                   rdfs:domain          rdfs:Resource .
rdf:_i                   rdfs:range           rdfs:Resource .
```

Today's Session: RDF(S) semantics

1. What is Semantics?
2. What is Model-theoretic Semantics?
3. Model-theoretic Semantics for RDF(S)
4. What is Proof-theoretic Semantics?
5. Proof-theoretic Semantics for RDF(S)
6. Class Project
7. Class Presentations

Back to our contrived example

- Say, a model I of a set K of sentences consists of
  - a set C of cars and
  - a function I(·) which maps each variable to a car in C

  such that, for each sentence a η b in K we have that
  I(a) has more horsepower than I(b).

- **Can we find an algorithm to compute all logical consequences of a set of sentences?**
- **Algorithm Input: set K of sentences**
  1. **The algorithm non-deterministically selects two sentences from K. If the first sentence is a η b, and the second sentence is b η c, then add a η c to K.**

     IF a η b $\in$ K and b η c $\in$ K THEN K$\leftarrow$ {a η c}
  2. **Repeat step 1 until no selection results in a change of K.**
  3. **Output: K**

Back to the example

- The algorithm produces only logical consequences: it is sound with respect to the model-theoretic semantics.
- The algorithm produces all logical consequences: it is complete with respect to the model-theoretic semantics.
- The algorithm always terminates.
- The algorithm is non-deterministic.
- What is the computational complexity of this algorithm?

What do we gain?

- **Recall:**
  - **β is a logical consequence of α (written $\alpha \vDash \beta$), if for all M with $M \vDash \alpha$, we also have $M \vDash \beta$ are**

- **Implementing model-theoretic semantics directly is not feasible: We would have to deal with *all* models of a knowledge base. Since there are a lot of cars in this world, we would have to check a lot of possibilities.**

- **Proof theory reduces model-theoretic semantics to symbol manipulation! It removes the models from the process.**

Deduction rules

IF $a\ \eta\ b \in K$ and $b\ \eta\ c \in K$ THEN $K \leftarrow \{a\ \eta\ c\}$

is a so-called *deduction rule*. Such rules are usually written schematically as

$$\frac{a\ \eta\ b \qquad b\ \eta\ c}{a\ \eta\ c}$$

Today's Session: RDF(S) semantics

1. What is Semantics?
2. What is Model-theoretic Semantics?
3. Model-theoretic Semantics for RDF(S)
4. What is Proof-theoretic Semantics?
5. Proof-theoretic Semantics for RDF(S)
6. Class Project
7. Class Presentations

First, some notation

- a and b can refer to arbitrary URIs (i.e. anything admissible for the predicate position in a triple),
- $\_{:}n$ will be used for the ID of a blank node,
- u and v refer to arbitrary URIs or blank node IDs (i.e. any possible subject of a triple),
- x and y can be used for arbitrary URIs, blank node IDs or literals (i.e. anything admissible for the object position in a triple), and
- l may be any literal.

Simple Entailment Rules

$$\frac{u \quad a \quad x \quad .}{u \quad a \quad \_{:}n \quad .}\ \mathrm{se1}$$

$$\frac{u \quad a \quad x \quad .}{\_{:}n \quad a \quad x \quad .}\ \mathrm{se2}$$

_:n must not be contained in the graph the rule is applied to

Additional RDF-entailment Rules

$$\frac{}{u\ \ a\ \ x}\ \text{rdfax}$$

for all RDF axiomatic triples $u\ \ a\ \ x\ .$

$$\frac{u\ \ a\ \ l\ \ .}{u\ \ a\ \ \_{:}n\ \ .}\ \text{lg}$$

where _:n does not yet occur in the graph

$$\frac{u\ \ a\ \ y\ \ .}{a\ \ \texttt{rdf:type}\ \ \texttt{rdf:Property}\ \ .}\ \text{rdf1}$$

$$\frac{u\ \ a\ \ l\ \ .}{\_{:}n\ \ \texttt{rdf:type}\ \ \texttt{rdf:XMLLiteral}}\ \text{rdf2}$$

where _:n does not yet occur in the graph, unless it has been introduced by a preceding application of the lg rule

Additional RDFS-entailment Rules - I

$$\frac{}{u \quad a \quad x} \text{ rdfax}$$

for all RDFS axiomatic triples $u \quad a \quad x \, .$

$$\frac{u \quad a \quad l \quad .}{\texttt{\_:}n \quad \texttt{rdf:type rdfs:Literal} \quad .} \text{ rdfs1}$$

with _:n as usual

$$\frac{a \quad \texttt{rdfs:domain} \quad x \quad . \qquad u \quad a \quad y \quad .}{u \quad \texttt{rdf:type} \quad x \quad .} \text{ rdfs2}$$

$$\frac{a \quad \texttt{rdfs:range} \quad x \quad . \qquad u \quad a \quad v \quad .}{v \quad \texttt{rdf:type} \quad x \quad .} \text{ rdfs3}$$

$$\frac{u \quad a \quad x \quad .}{u \quad \texttt{rdf:type rdfs:Resource} \quad .} \text{ rdfs4a}$$

$$\frac{u \quad a \quad v \quad .}{v \quad \texttt{rdf:type rdfs:Resource} \quad .} \text{ rdfs4b}$$

Additional RDFS-entailment Rules - II

$$\frac{u\ \texttt{rdfs:subPropertyOf}\ v\ . \qquad v\ \texttt{rdfs:subPropertyOf}\ x\ .}{u\ \texttt{rdfs:subPropertyOf}\ x\ .}\ \text{rdfs5}$$

$$\frac{u\ \texttt{rdf:type rdf:Property}\ .}{u\ \texttt{rdfs:subPropertyOf}\ u\ .}\ \text{rdfs6}$$

$$\frac{a\ \texttt{rdfs:subPropertyOf}\ b\ . \qquad u\ a\ y\ .}{u\ b\ y\ .}\ \text{rdfs7}$$

$$\frac{u\ \texttt{rdf:type rdfs:Class}\ .}{u\ \texttt{rdfs:subClassOf rdfs:Resource}\ .}\ \text{rdfs8}$$

$$\frac{u\ \texttt{rdfs:subClassOf}\ x\ . \qquad v\ \texttt{rdf:type}\ u\ .}{v\ \texttt{rdf:type}\ x\ .}\ \text{rdfs9}$$

$$\frac{u\ \texttt{rdf:type rdfs:Class}\ .}{u\ \texttt{rdfs:subClassOf}\ u\ .}\ \text{rdfs10}$$

Additional RDFS-entailment Rules - III

$$\frac{u \ \texttt{rdfs:subClassOf} \ v \ . \qquad v \ \texttt{rdfs:subClassOf} \ x \ .}{u \ \texttt{rdfs:subClassOf} \ x \ .} \ \text{rdfs11}$$

$$\frac{u \ \texttt{rdf:type rdfs:ContainerMembershipProperty} \ .}{u \ \texttt{rdfs:subPropertyOf rdfs:member} \ .} \ \text{rdfs12}$$

$$\frac{u \ \texttt{rdf:type rdfs:Datatype} \ .}{u \ \texttt{rdfs:subClassOf rdfs:Literal} \ .} \ \text{rdfs13}$$

$$\frac{u \quad a \quad \_{:}n \quad .}{u \quad a \quad l \quad .} \ \text{gl}$$

where _:n identifies a blank node introduced by an earlier "weakening" of the literal l via the rule lg

Completeness?

- **The deduction rules for simple and RDF entailment are sound and complete.**

- **The deduction rules for RDFS entailment are sound.**

  The spec says, they are also complete, but they are not:

```
ex:isHappilyMarriedTo   rdfs:subPropertyOf      _:bnode .
_:bnode                 rdfs:domain             ex:Person .
ex:markus               ex:isHappilyMarriedTo   ex:anja .
```

has as logical consequence

```
ex:markus               rdf:type                ex:Person .
```

but this is not derivable using the deduction rules.

Complexity

Simple, RDF, and RDFS entailment are NP-complete problems.

If we disallow blank nodes, all three entailment problems are polynomial.

Does RDFS semantics do what it should?

Does

```
ex:speaksWith    rdfs:domain       ex:Homo .
ex:Homo          rdfs:subClassOf   ex:Primates .
```

entail

```
ex:speaksWith    rdfs:domain       ex:Primates .
```

?

RDF next version

A new W3C working group is currently under way:

http://www.w3.org/2011/rdf-wg/

- bugfixing (e.g., incompleteness of inference rules)
- new features for RDF and RDFS
  - blank node identifiers (i.e., URIs)
  - working with multiple graphs
- JSON serialization
- Turtle syntax

forthcoming:

- Semantics and other docs.

Today's Session: RDF(S) semantics

1. What is Semantics?
2. What is Model-theoretic Semantics?
3. Model-theoretic Semantics for RDF(S)
4. What is Proof-theoretic Semantics?
5. Proof-theoretic Semantics for RDF(S)
6. Class Project
7. Class Presentations

Class project: next step

- keep bugfixing
- find, for your RDF Schema ontology, each of the following:
  - a triple which is RDFS-entailed, but not RDF-entailed
  - a triple which is RDF-entailed, but not simply entailed
  - a triple which is simply entailed
- For each of them, write down a justification why it is entailed.

- send to me by Friday 20th of January
  - the current version of your Turtle RDF Schema document
  - the three entailed triples with explanations.

Today's Session: RDF(S) semantics

1. What is Semantics?
2. What is Model-theoretic Semantics?
3. Model-theoretic Semantics for RDF(S)
4. What is Proof-theoretic Semantics?
5. Proof-theoretic Semantics for RDF(S)
6. Class Project
7. Class Presentations

Class presentations – first topics

<nothing yet>

Class planning (tentative)

Tuesday 10th of January: RDF Schema
Thursday 12th of January: RDF and RDFS Semantics
Tuesday 17th of January: RDF and RDFS Semantics
Thursday 19th of January: Exercise Session

Then several OWL sessions.