INF3580 – Semantic Technologies – Spring 2011

Lecture 8: RDF and RDFS semantics

Martin Giese

15th March 2011

Department of Informatics

University of Oslo

Today's Plan

1. Why we need semantics
2. Model-theoretic semantics from a birds-eye perspective
3. Repetition: Propositional Logic
4. Simplified RDF semantics
5. Open World Semantics

Learning goals:

1. To understand the basic concepts of model-theoretic semantics.
2. To understand a simple semantics for parts of RDF/RDFS
3. To get acquainted with the idiosyncracies of Semantic Web reasoning vs. e.g. SQL, as well as
   - the open/closed world distinction, and
   - the non-unique names assumption

We shall be less concerned with:

1. all the nitty-gritty detail of RDF semantics,
2. characterisation results such as soundness and completeness.

Outline

1. Why we need semantics
2. Model-theoretic semantics from a birds-eye perspective
3. Repetition: Propositional Logic
4. Simplified RDF semantics
5. Open World Semantics

Semantics—why do we need it?

A formal semantics for RDFS became necessary because

1. the previous informal specification
2. left plenty of room for interpretation of conclusions, whence
3. triple stores sometimes answered queries differently, thereby
4. obstructing interoperability and interchangeability.
5. The information content of data once more came to depend on applications

But RDF was supposed to be the data liberation movement!

Another look at the Semantic Web cake

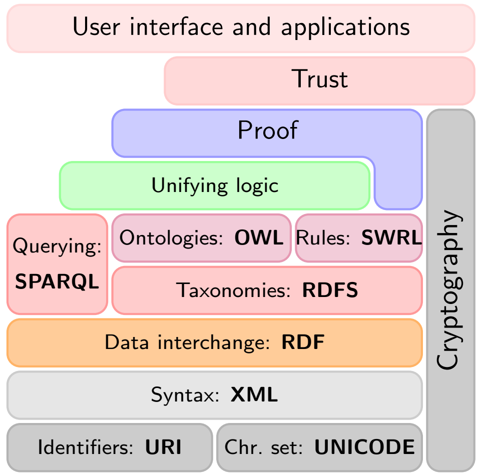


Figure: Semantic Web Stack

Absolute precisision required

RDF is to serve as the foundation of the entire Semantic Web tower.

- It must therefore be sufficiently clear to sustain advanced reasoning, e. g.:
  - type propagation/inheritance,
    - "Tweety is a penguin and a penguin is a bird, so ..."
  - domain and range restrictions,
    - "Martin has a birthdate, and only people have birthdates, so ..."
  - existential restrictions.
    - "all persons have parents, and Martin is a person, so ...."

  .... to which we shall return in later lectures

To ensure that infinitely many conclusions will be agreed upon,

- RDF must be furnished with a model-theory
- that specifies how the different node types should be interpreted
- and in particular what entailment should be taken to mean.

Example: What is the meaning of blank nodes?

Example from SPARQL lecture:

```
SELECT DISTINCT ?name WHERE {
    _:pub dc:creator [foaf:name "Martin Giese"] .
    _:pub dc:creator _:other .
    _:other foaf:name ?name.
}
```

SPARQL must

- match the query to graph patterns
- which involves assigning values to variables and blank nodes

But,

- which values are to count?
- the problem becomes more acute under e.g. type propagation.
- Should a value for `foaf:familyname` match a query for `foaf:name`?
- Are blanks in SPARQL the same as blanks in RDF?
- Complete answers in the course of later lectures. Foundations now.

Outline

Formal semantics

- The study of how to model the meaning of a logical calculus.
- A logical calculus consists of:
  - A finite set of symbols,
  - a grammar, which specifies the formulae,
  - a set of axioms and inference rules from which we construct proofs.
- A logical calculus can be defined apart from any interpretation.
- A calculus that has not been furnished with a formal semantics,
  - is a 'blind' machine, a mere symbol manipulator,
  - the only criterion of correctness is provability.

Derivations

A proof typically looks something like this:

$$\dfrac{\dfrac{\dfrac{P \vdash Q, P \qquad Q, P \vdash Q}{P \rightarrow Q, P \vdash Q} \qquad \dfrac{R \vdash Q, P \qquad Q, R \vdash Q}{P \rightarrow Q, R \vdash Q}}{P \rightarrow Q, P \vee R \vdash Q}}{P \rightarrow Q \vdash (P \vee R) \rightarrow Q}$$

Where each line represents an application of an inference rule.

- How do we know that the inference rules are well-chosen?
- Which manipulations are intuitively meaningful?
- When is a proof *intuitively* acceptable?

Model-theoretic semantics

Basic idea: Asserting a sentence makes a claim about the world:

- A formula therefore limits the set of worlds that are possible.
- We can therefore encode meaning/logical content
  - by describing models of these worlds.
  - thus making *certain aspects* of meaning mathematically tractable
- The exact makeup of models typically varies, but they all
  - express a view on what kinds of things there are,
  - and the basic relations between these things
- By selecting a class of models one selects the basic features of the world
  - as one chooses to see it.
- Whatever these models all share can be said to be entailed by those features.

Outline

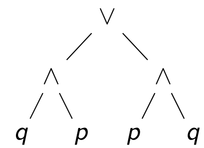
Propositional Logic: Formulas

- Formulas are defined "by induction" or "recursively":

1 Any letter p, q, r,... is a formula
2 *if* A and B are formulas, *then*
   - $(A \land B)$ is also a formula (read: "A and B")
   - $(A \lor B)$ is also a formula (read: "A or B")
   - $\neg A$ is also a formula (read: "not A")

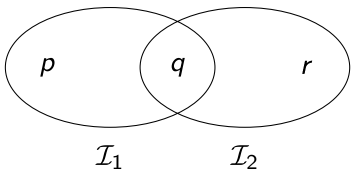
- Nothing else is. Only what rules [1] and [2] say is a formula.
- Examples of formulae: p $\quad (p \land \neg r)$ $\quad (q \land \neg q)$ $\quad ((p \lor \neg q) \land \neg p)$
- Formulas are just a kind of strings until now:
   - no meaning
   - but every formula can be "parsed" uniquely.

$$((q \land p) \lor (p \land q))$$



Interpretations

- Logic is about truth and falsity
- Truth of compound formulas depends on truth of letters.
- Idea: put all letters that are "true" into a set!
- Define: An *interpretation* $\mathcal{I}$ is a set of letters.
- Letter p is true in interpretation $\mathcal{I}$ if $p \in \mathcal{I}$.
- E.g., in $\mathcal{I}_1 = \{p, q\}$, p is true, but r is false.



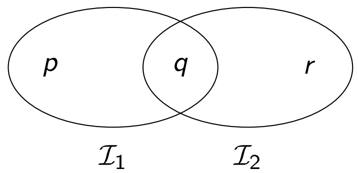
- But in $\mathcal{I}_2 = \{q, r\}$, p is false, but r is true.

Semantic Validity $\models$

- To say that p is true in $\mathcal{I}$, write

$$\mathcal{I} \models p$$

- For instance



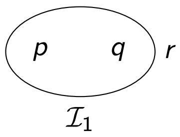
$$\mathcal{I}_1 \models p \qquad \mathcal{I}_2 \not\models p$$

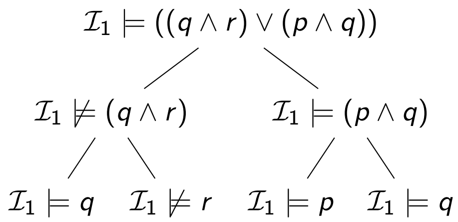
- In other words, for all letters p:

$$\mathcal{I} \models p \quad \text{if and only if} \quad p \in \mathcal{I}$$

Validity of Compound Formulas

- Is $((q \wedge r) \vee (p \wedge q))$ true in $\mathcal{I}$?
- Idea: apply our rule recursively
- For any formulas A and B,...
- ...and any interpretation $\mathcal{I}$,...
  - ...$\mathcal{I} \models A \wedge B$ if and only if $\mathcal{I} \models A$ and $\mathcal{I} \models B$
  - ...$\mathcal{I} \models A \vee B$ if and only if $\mathcal{I} \models A$ or $\mathcal{I} \models B$ (or both)
  - ...$\mathcal{I} \models \neg A$ if and only if $\mathcal{I} \not\models A$.
- For instance





Truth Table

- Semantics of $\neg$, $\wedge$, $\vee$ often given as *truth table*:

A	B	$\neg A$	$A \wedge B$	$A \vee B$
f	f	t	f	f
f	t	t	f	t
t	f	f	f	t
t	t	f	t	t

Tautologies

- A formula A that is true in *all* interpretations is called a *tautology*
- also *logically valid*
- also a *theorem* (of propositional logic)
- written:

$$\models A$$

- $(p \vee \neg p)$ is a tautology
- True whatever p means:
  - The sky is blue or the sky is not blue.
  - P.N. will win the race in 2013 or P.N. will not win the race in 2013.
  - The slithy toves gyre or the slithy toves do not gyre.
- Possible to derive true statements mechanically...
- ...without understanding their meaning!
- ...e.g. using truth tables for small cases.

Entailment

- Tautologies are true in all interpretations
- Some formulas are true only under certain assumptions
- *A entails B*, written $A \models B$ if
  $\mathcal{I} \models B$
  for all interpretations $\mathcal{I}$ with $\mathcal{I} \models A$
- Also: "B is a logical consequence of A"
- Whenever A holds, also B holds
- For instance:

$$p \wedge q \models p$$

- Independent of meaning of p and q:
  - If it rains and the sky is blue, then it rains
  - If P.N. wins the race and the world ends, then P.N. wins the race
  - If 'tis brillig and the slythy toves do gyre, then 'tis brillig
- Also entailment can be checked mechanically, without knowing the

Outline

Taking the structure of triples into account

Unlike propositions, triples have parts, namely:

- subject
- predicates, and
- objects

Less abstractly, these may be:

- URI references
- literal values, and
- blank nodes

Triples are true or false on the basis of what each part refers to.

On what there is: Resources

The RDF data model consists of three object types; resources, properties and literals values:

Resources: All things described by RDF are called resources. A resource may be:

- an entire Web page,
- a part of a Web page,
- a whole collection of pages (a Web site), or
- an object that is not directly accessible via the Web, e.g. a printed book.

Resource contd.

Resources are always named by URIs. Examples:

- `http://purl.org/dc/terms/created`
  - names the concept of a creation date.
- `http://www.wikipedia.org`
  - names Wikipedia, the Web site.
- `http://dblp.l3s.de/d2r/resource/authors/Martin_Giese`
  - names Martin Giese, the person.

Properties

Properties A property is a specific aspect, characteristic, attribute or relation used to describe a resource.

Properties are always named by URIs. Examples.

- `http://xmlns.com/foaf/0.1/knows`
  - names the relationship of knowing people,
- `http://dbpedia.org/property/parent`
  - names the relationship of being a parent,
- `http://www.w3.org/2006/vcard/ns#locality`
  - names the relationship of being the locality of something.

Literal values

Literal values A literal value is a concrete data item, such as an integer or a string.

Plain literals name themselves, i. e.

- “Julius Ceasar” names the string “Julius Ceasar”
- “42” names the string “42”

The semantics of typed and tagged literals is considerably more complex.

Restricting RDF/RDFS

- We will simplify things by only looking at certain kinds of RDF graphs.
- No triples “about” properties, classes, etc., except RDFS
- Assume Resources are divided into four disjoint types:
  - *Properties* like `foaf:knows`, `dc:title`
  - *Classes* like `foaf:Person`
  - *Built-ins*, a fixed set including `rdf:type`, `rdfs:domain`, etc.
  - *Individuals* (all the rest, “usual” resources)
- All triples have one of the forms:

```
individual property individual .
individual rdf:type class .

class rdfs:subClassOf class .
property rdfs:subPropertyOf property .
property rdfs:domain class .
property rdfs:range class .
```

- Forget blank nodes and literals for a while!

Short Forms

- Resources and Triples are no longer all alike
- No need to use the same general triple notation
- Use alternative notation

Triples	Abbreviation
`indi prop indi .`	$r(i_1, i_2)$
`indi rdf:type class .`	$C(i_1)$
`class rdfs:subClassOf class .`	$C \sqsubseteq D$
`prop rdfs:subPropOf prop .`	$r \sqsubseteq s$
`prop rdfs:domain class .`	$\mathrm{dom}(r, C)$
`prop rdfs:range class .`	$\mathrm{rg}(r, C)$

- This is called “Description Logic” (DL) Syntax
- Used much in particular for OWL

Example

- Triples:

```
ws:romeo ws:loves ws:juliet .
ws:juliet rdf:type ws:Lady .

ws:Lady rdfs:subClassOf foaf:Person .
ws:loves rdfs:subPropertyOf foaf:knows .
ws:loves rdfs:domain ws:Lover .
ws:loves rdfs:range ws:Beloved .
```

- DL syntax, without namespaces:

$\mathit{loves}(\mathit{romeo}, \mathit{juliet})$
$\mathit{Lady}(\mathit{juliet})$

$\mathit{Lady} \sqsubseteq \mathit{Person}$
$\mathit{loves} \sqsubseteq \mathit{knows}$
$\mathrm{dom}(\mathit{loves}, \mathit{Lover})$
$\mathrm{rg}(\mathit{loves}, \mathit{Beloved})$

Interpretations for RDF

- To interpret propositional formulas, we need to know how to interpret
  - Letters
- To interpret the six kinds of triples, we need to know how to interpret
  - *Individual URIs* as real or imagined objects
  - *Class URIs* as sets of such objects
  - *Property URIs* as relations between these objects
- A *DL-interpretation* $\mathcal{I}$ consists of
  - A set $\Delta^{\mathcal{I}}$, called the *domain* (sorry!) of $\mathcal{I}$
  - For each individual URI i, an element $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
  - For each class URI C, a subset $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - For each property URI r, a relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
- Given these, it will be possible to say whether a triple holds or not.

An example "intended" interpretation

- $\Delta^{\mathcal{I}_1} =$ {  }
- $\mathit{romeo}^{\mathcal{I}_1} =$  $\mathit{juliet}^{\mathcal{I}_1} =$ 
- $\mathit{Lady}^{\mathcal{I}_1} =$ {  } $\mathit{Person}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1}$

  $\mathit{Lover}^{\mathcal{I}_1} = \mathit{Beloved}^{\mathcal{I}_1} =$ {  }
- $\mathit{loves}^{\mathcal{I}_1} =$ {  }

  $\mathit{knows}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$

An example "non-intended" interpretation

- $\Delta^{\mathcal{I}_2} = \mathbb{N} = \{1, 2, 3, 4, \ldots\}$
- $\mathit{romeo}^{\mathcal{I}_2} = 17$
  $\mathit{juliet}^{\mathcal{I}_2} = 32$
- $\mathit{Lady}^{\mathcal{I}_2} = \{2^n \mid n \in \mathbb{N}\} = \{2, 4, 8, 16, 32, \ldots\}$
  $\mathit{Person}^{\mathcal{I}_2} = \{2n \mid n \in \mathbb{N}\} = \{2, 4, 6, 8, 10, \ldots\}$
  $\mathit{Lover}^{\mathcal{I}_2} = \mathit{Beloved}^{\mathcal{I}_2} = \mathbb{N}$
- $\mathit{loves}^{\mathcal{I}_2} = <= \{\langle x, y\rangle \mid x < y\}$
  $\mathit{knows}^{\mathcal{I}_2} = \leq= \{\langle x, y\rangle \mid x \leq y\}$
- Just because names (URIs) look familiar, they don't need to denote what we think!
- In fact, there is *no way* of ensuring they denote only what we think!

Validity in Interpretations (RDF)

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models r(i_1, i_2)$ iff $\langle i_1^{\mathcal{I}}, i_2^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$
- $\mathcal{I} \models C(i)$ iff $i^{\mathcal{I}} \in C^{\mathcal{I}}$
- Examples:
  - $\mathcal{I}_1 \models \mathit{loves}(\mathit{juliet}, \mathit{romeo})$ because

    $\langle$  $\rangle \in \mathit{loves}^{\mathcal{I}_1} = \{$  $\}$
  - $\mathcal{I}_1 \models \mathit{Person}(\mathit{romeo})$ because

    $\mathit{romeo}^{\mathcal{I}_1} =$  $\in \mathit{Person}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1}$
  - $\mathcal{I}_2 \not\models \mathit{loves}(\mathit{juliet}, \mathit{romeo})$ because
    $\mathit{loves}^{\mathcal{I}_2} = <$ and $\mathit{juliet}^{\mathcal{I}_2} = 32 \not< \mathit{romeo}^{\mathcal{I}_2} = 17$
  - $\mathcal{I}_2 \not\models \mathit{Person}(\mathit{romeo})$ because
  - $\mathit{romeo}^{\mathcal{I}_2} = 17 \notin \mathit{Person}^{\mathcal{I}_2} = \{2, 4, 6, 8, 10, \ldots\}$

Validity in Interpretations, cont. (RDFS)

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models C \subseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I} \models r \subseteq s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
- $\mathcal{I} \models \mathrm{dom}(r, C)$ iff $\mathrm{dom}\, r^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
- $\mathcal{I} \models \mathrm{rg}(r, C)$ iff $\mathrm{rg}\, r^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
- Examples:
  - $\mathcal{I}_1 \models \mathit{Lover} \sqsubseteq \mathit{Person}$ because

    $\mathit{Lover}^{\mathcal{I}_1} = \{$  $\} \subseteq \mathit{Person}^{\mathcal{I}_1} = \{$  $\}$
  - $\mathcal{I}_2 \not\models \mathit{Lover} \sqsubseteq \mathit{Person}$ because
    $\mathit{Lover}^{\mathcal{I}_2} = \mathbb{N}$ and $\mathit{Person}^{\mathcal{I}_2} = \{2, 4, 6, 8, 10, \ldots\}$

Example: Range/Domain semantics

$$\mathcal{I}_2 \models \mathrm{dom}(\mathit{knows}, \mathit{Beloved})$$

because. . .

$$\mathit{knows}^{\mathcal{I}_2} = \leq = \{\langle x, y \rangle \mid x \leq y\}$$

Therefore, $\mathit{knows}^{\mathcal{I}_2}$ has domain

$$\mathrm{dom}\, \mathit{knows}^{\mathcal{I}_2} = \mathrm{dom} \leq = \{x \in \mathbb{N} \mid x \leq y \text{ for some } y \in \mathbb{N}\} = \mathbb{N}$$

Furthermore,

$$\mathit{Beloved}^{\mathcal{I}_2} = \mathbb{N}$$

And thus:

$$\mathrm{dom}\, \mathit{knows}^{\mathcal{I}_2} \subseteq \mathit{Beloved}^{\mathcal{I}_2}$$

Interpretation of Sets of Triples

- Given an interpretation $\mathcal{I}$
- And a set of triples $\mathcal{A}$ (any of the six kinds)
- $\mathcal{A}$ is valid in $\mathcal{I}$, written
  $$\mathcal{I} \models \mathcal{A}$$
- iff $\mathcal{I} \models A$ for all $A \in \mathcal{A}$.
- Then $\mathcal{I}$ is also called a model of $\mathcal{A}$.
- Examples:
  $$\mathcal{A} = \{\mathit{loves}(\mathit{romeo}, \mathit{juliet}), \mathit{Lady}(\mathit{juliet}), \mathit{Lady} \sqsubseteq \mathit{Person}, \mathit{loves} \sqsubseteq \mathit{knows}, \mathrm{dom}(\mathit{loves}, \mathit{Lover}), \mathrm{rg}(\mathit{loves}, \mathit{Beloved})\}$$
- Then $\mathcal{I}_1 \models \mathcal{A}$ and $\mathcal{I}_2 \models \mathcal{A}$

Entailment

- Given a set of triples $\mathcal{A}$ (any of the six kinds)
- And a further triple T (also any kind)
- T is entailed by $\mathcal{A}$, written $\mathcal{A} \models T$
- iff
  - For any interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$
  - $\mathcal{I} \models T$.
- $\mathcal{A} \models \mathcal{B}$ iff $\mathcal{I} \models \mathcal{B}$ for all $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$
- Example:
- $\mathcal{A} = \{\ldots, \mathit{Lady}(\mathit{juliet}),\ \mathit{Lady} \sqsubseteq \mathit{Person}, \ldots\}$ as before
- $\mathcal{A} \models \mathit{Person}(\mathit{juliet})$ because...
- in *any* interpretation $\mathcal{I}$...
- if $\mathit{juliet}^{\mathcal{I}} \in \mathit{Lady}^{\mathcal{I}}$ and $\mathit{Lady}^{\mathcal{I}} \subseteq \mathit{Person}^{\mathcal{I}}$ ...
- then by set theory $\mathit{juliet}^{\mathcal{I}} \in \mathit{Person}^{\mathcal{I}}$

Countermodels

- If $\mathcal{A} \not\models T$,...
- then there is an $\mathcal{I}$ with
  - $\mathcal{I} \models \mathcal{A}$
  - $\mathcal{I} \not\models T$
- Vice-versa: if $\mathcal{I} \models \mathcal{A}$ and $\mathcal{I} \not\models T$, then $\mathcal{A} \not\models T$
- Such an $\mathcal{I}$ is called a *counter-model* (for the assumption that $\mathcal{A}$ entails T)
- To show that $\mathcal{A} \models T$ does *not* hold:
  - Describe an interpretation $\mathcal{I}$ (using your fantasy)
  - Prove that $\mathcal{I} \models \mathcal{A}$ (using the semantics)
  - Prove that $\mathcal{I} \not\models T$ (using the semantics)

Countermodel Example

- $\mathcal{A}$ as before:

$$\mathcal{A} = \{\mathit{loves}(\mathit{romeo}, \mathit{juliet}),\ \mathit{Lady}(\mathit{juliet}),\ \mathit{Lady} \sqsubseteq \mathit{Person},\ \mathit{loves} \sqsubseteq \mathit{knows},\ \mathrm{dom}(\mathit{loves}, \mathit{Lover}),\ \mathrm{rg}(\mathit{loves}, \mathit{Beloved})\}$$

- Does $\mathcal{A} \models \mathit{Lover} \sqsubseteq \mathit{Beloved}$?
- Holds in $\mathcal{I}_1$ and $\mathcal{I}_2$.
- Try to find an interpretaion with $\Delta^{\mathcal{I}} = \{a, b\}$, $a \neq b$.
- Interpret $\mathit{romeo}^{\mathcal{I}} = a$ and $\mathit{juliet}^{\mathcal{I}} = b$
- Then $\langle a, b \rangle \in \mathit{loves}^{\mathcal{I}}$, $a \in \mathit{Lover}^{\mathcal{I}}$, $b \in \mathit{Beloved}^{\mathcal{I}}$.
- With $\mathit{Lover}^{\mathcal{I}} = \{a\}$ and $\mathit{Beloved}^{\mathcal{I}} = \{b\}$, $\mathcal{I} \not\models \mathit{Lover} \sqsubseteq \mathit{Beloved}$!
- Choose

$$\mathit{loves}^{\mathcal{I}} = \mathit{knows}^{\mathcal{I}} = \{\langle a, b \rangle\} \qquad \mathit{Lady}^{\mathcal{I}} = \mathit{Person}^{\mathcal{I}} = \{b\}$$

to complete the count-model while satisfying $\mathcal{I} \models \mathcal{A}$

Outline

Open and closed world reasoning

RDF semantics is open-world: Entailment is defined in terms of all models:

RDF-entailment

An RDF graph $\mathcal{A}$ entails a graph $\mathcal{B}$ if every interpretation $\mathcal{I}$ that satisfies $\mathcal{A}$ also satisfies $\mathcal{B}$.

Just as with propositional semantics, therefore:

- one model does not in general suffice to decide entailment
- one model cannot in general be assumed to represent complete knowledge

Why open world semantics?

Remember the AAA rule:

Anyone can say Anything about Anything

- Anyone can write a page saying what they please,
- information may be discovered at any time,
- data may be produced at any time
- conclusions in general are drawn from distributed data

Hence, we will rarely be able to conclude e.g.

- that Radiohead does not have an album called "Dark Continent",
- because although we cannot find information about such an album,
- or we may find a similarly named album by another band,
- we may yet discover new information as we go.

Ramifications of the closed world assumption

Open world semantics becomes an issue for negative information.

- Imagine a relational database for an airline's flights:
  - If a direct flight between Kautokeino and Jakutsk cannot be found,
  - the RDBMS will assume that no such flight exists.
- This makes sense, because:
  - A database for an airline is usually complete wrt their flights
- This kind of reasoning is known as negation as failure:
  - what cannot be proved to be true is assumed false,
- Negation as failure characterises;
  - Negation in logic programming, e.g. Prolog.
  - negation in relational database management systems,
  - default reasoning in general.

Sensitivity to the *absence of information*

A closed world system is sensitive to the absence of information:

- If it is not in the data, then conclude that it does not hold.
- If "Dark Continent" by Radiohead cannot be found, there isn't one.
- If I can find the names of all planets except for Jupiter, then there are 7 planets.

You do not want this behaviour from SPARQL:

- If you merge information from more sources, Jupiter may show up.
- Perhaps Radiohead releases "Dark Continent" tomorrow.

Therefore SPARQL is based on classical semantics, whence

- it is not sensitive to absence, whence
- it makes little sense to provide for negative queries,

because you'll never get an answer anyway.

The non-unique names assumption

Closely related to the AAA rule and the OWA is the ACAA rule:

The ACAA rule

- Anyone can Call Anything Anything,
- Identifiers cannot be assumed to be unique,
- Different names do not necessarily mean different objects

For instance;

- Even though five names may be registered with the same address,
- we cannot conclude that the household has at least 5 members.

In order to make such inference we must;

- explicitly state which names denote different objects,
- with `owl:differentFrom`,
- more about this later in lecture 11.

Take aways

1. Model-theoretic semantics yields an unambigous notion of entailment,
2. which is necessary in order to liberate data from applications.
3. Shown today: A simplified semantics for parts of RDF
   1. Only RDF/RDFS vocabulary to talk “about” predicates and classes
   2. Literals and blank nodes next time
4. Open world semantics
   1. is required by the open nature of the Web,
   2. but makes classical negation of little use in queries.

Supplementary reading

RDF semantics:

- `http://www.w3.org/TR/rdf-mt/`

The metamodelling architecture of Web Ontology Languages:

- `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.7263`

On closed world reasoning in SPARQL:

- `http://clarkparsia.com/pellet/icv`