INF3580 – Semantic Technology – Spring 2010

Lecture 5: RDF and Web semantics

Audun Stolpe

23rd February 2010

Department of Informatics

University of Oslo

Today's Plan

1. Why we need semantics
2. Model-theoretic semantics from a birds-eye perspective
3. Recalling classical consequence
4. RDF semantics—main features

Learning goals:

1. To understand the basic concepts of model-theoretic semantics.

Learning goals:

1. To understand the basic concepts of model-theoretic semantics.
2. To understand the RDF meta-modeling architecture

Learning goals:

1. To understand the basic concepts of model-theoretic semantics.
2. To understand the RDF meta-modeling architecture
3. To get acquainted with the idiosyncracies of Semantic Web reasoning vs. e.g. SQL, as well as

Learning goals:

1. To understand the basic concepts of model-theoretic semantics.
2. To understand the RDF meta-modeling architecture
3. To get acquainted with the idiosyncracies of Semantic Web reasoning vs. e.g. SQL, as well as
    - the open/closed world distinction, and

Learning goals:

1. To understand the basic concepts of model-theoretic semantics.
2. To understand the RDF meta-modeling architecture
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

Learning goals:

1. To understand the basic concepts of model-theoretic semantics.
2. To understand the RDF meta-modeling architecture
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:

Learning goals:

1. To understand the basic concepts of model-theoretic semantics.
2. To understand the RDF meta-modeling architecture
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,

Learning goals:

1. To understand the basic concepts of model-theoretic semantics.
2. To understand the RDF meta-modeling architecture
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. Recalling classical consequence
4. RDF semantics—main features

Semantics—why do we need it?

A formal semantics for RDFS became necessary because

Semantics—why do we need it?

A formal semantics for RDFS became necessary because

1. the previous informal specification

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

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

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.

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

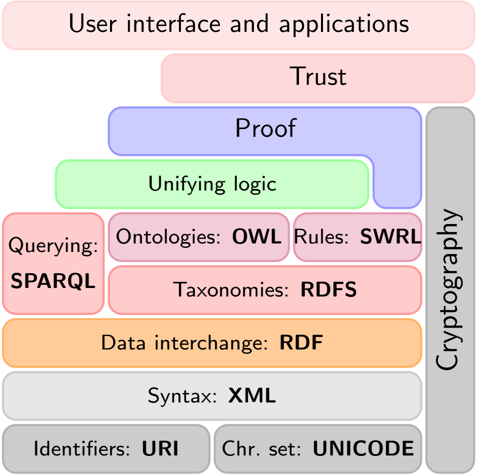
Semantics—why do we need it?

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



User interface and applications

Trust

Proof

Unifying logic

Querying: **SPARQL**

Ontologies: **OWL**

Rules: **SWRL**

Taxonomies: **RDFS**

Data interchange: **RDF**

Syntax: **XML**

Identifiers: **URI**

Chr. set: **UNICODE**

Cryptography

Figure: Semantic Web Stack

Absolute precisision required

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

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.:

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,

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 ..."

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,

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 ..."

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.

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 ...."

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 lecture 6 and onwards

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 lecture 6 and onwards

To ensure that infinitely many conclusions will be agreed upon,

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 lecture 6 and onwards

To ensure that infinitely many conclusions will be agreed upon,

- RDF must be furnished with a model-theory

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 lecture 6 and onwards

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

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 lecture 6 and onwards

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.

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 lecture 6 and onwards

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: What is the meaning of blank nodes?

Example from previous lecture:

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

Example: What is the meaning of blank nodes?

Example from previous lecture:

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

SPARQL must

Example: What is the meaning of blank nodes?

Example from previous 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

Example: What is the meaning of blank nodes?

Example from previous 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

Example: What is the meaning of blank nodes?

Example from previous 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,

- but which values are to count?

Example: What is the meaning of blank nodes?

Example from previous 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,

- but which values are to count?
- the problem becomes more acute under e.g. type propagation.

Example: What is the meaning of blank nodes?

Example from previous 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,

- 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`?

Example: What is the meaning of blank nodes?

Example from previous 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,

- 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?

Example: What is the meaning of blank nodes?

Example from previous 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,

- 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

1 Why we need semantics

2 Model-theoretic semantics from a birds-eye perspective

3 Recalling classical consequence

4 RDF semantics—main features

Formal semantics

- The study of how to model the meaning of a logical calculus.

Formal semantics

- The study of how to model the meaning of a logical calculus.
- A logical calculus consists of:

Formal semantics

- The study of how to model the meaning of a logical calculus.
- A logical calculus consists of:
  - A finite set of symbols,

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,

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.

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.

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,

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,

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.

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:

Derivations

A proof typically looks something like this:

$$\dfrac{\dfrac{\dfrac{P \vdash Q, P \qquad Q, P \vdash Q}{P \longrightarrow Q, P \vdash Q} \qquad \dfrac{R \vdash Q, P \qquad Q, R \vdash Q}{P \longrightarrow Q, R \vdash Q}}{P \longrightarrow Q, P \lor R \vdash Q}}{P \longrightarrow Q \vdash (P \lor R) \longrightarrow Q}$$

Derivations

A proof typically looks something like this:

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

Where each line represents an application of an inference rule.

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?

Derivations

A proof typically looks something like this:

$$\dfrac{\dfrac{\dfrac{P \vdash Q, P \qquad Q, P \vdash Q}{P \to Q, P \vdash Q} \qquad \dfrac{R \vdash Q, P \qquad Q, R \vdash Q}{P \to Q, R \vdash Q}}{P \to Q, P \lor R \vdash Q}}{P \to Q \vdash (P \lor R) \to Q}$$

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?

Derivations

A proof typically looks something like this:

$$\dfrac{\dfrac{\dfrac{P \vdash Q, P \qquad Q, P \vdash Q}{P \to Q, P \vdash Q} \qquad \dfrac{R \vdash Q, P \qquad Q, R \vdash Q}{P \to Q, R \vdash Q}}{P \to Q, P \lor R \vdash Q}}{P \to Q \vdash (P \lor R) \to Q}$$

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:

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.

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

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.

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

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

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,

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

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

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.

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

1. Why we need semantics
2. Model-theoretic semantics from a birds-eye perspective
3. Recalling classical consequence
4. RDF semantics—main features

The language of classical logic

Sentence variables

The non-logical symbols consists of a countable set of sentence variables

- $P_1, P_2, P_3, \ldots$ (we drop the subscripts when they do not matter)

The language of classical logic

Sentence variables

The non-logical symbols consists of a countable set of sentence variables

- $P_1, P_2, P_3, \ldots$ (we drop the subscripts when they do not matter)

Logical connectives

The logical symbols consists of

- $\wedge$ aka. logical conjunction,
- $\vee$ aka. logical disjunction,
- $\rightarrow$ aka. material implication, and
- $\neg$ aka. logical negation

or some functionally equivalent set

Atomic sentences

- Sentence variables are place-holders for atomic sentences.

Atomic sentences

- Sentence variables are place-holders for atomic sentences.
- Atomic sentences are complete sentences that

Atomic sentences

- Sentence variables are place-holders for atomic sentences.
- Atomic sentences are complete sentences that
  - are either true or false,

Atomic sentences

- Sentence variables are place-holders for atomic sentences.
- Atomic sentences are complete sentences that
    - are either true or false,
    - contain none of the logical connectives.

Atomic sentences

- Sentence variables are place-holders for atomic sentences.
- Atomic sentences are complete sentences that
  - are either true or false,
  - contain none of the logical connectives.
- Examples;
  - "Kilimanjaro is the tallest mountain in Africa"

Atomic sentences

- Sentence variables are place-holders for atomic sentences.
- Atomic sentences are complete sentences that
  - are either true or false,
  - contain none of the logical connectives.
- Examples;
  - "Kilimanjaro is the tallest mountain in Africa"
  - "Popocatepetl is in Canada"

Atomic sentences

- Sentence variables are place-holders for atomic sentences.
- Atomic sentences are complete sentences that
  - are either true or false,
  - contain none of the logical connectives.
- Examples;
  - “Kilimanjaro is the tallest mountain in Africa”
  - “Popocatepetl is in Canada”
  - “The number of planets exceeds 7”

Atomic sentences

- Sentence variables are place-holders for atomic sentences.
- Atomic sentences are complete sentences that
  - are either true or false,
  - contain none of the logical connectives.
- Examples;
  - “Kilimanjaro is the tallest mountain in Africa”
  - “Popocatepetl is in Canada”
  - “The number of planets exceeds 7”

Complex sentences

Complex formulae correspond to combinations of atomic sentences;

Complex sentences

Complex formulae correspond to combinations of atomic sentences;

- “Popocatepetl is in Canada and the number of planets exceeds 7”

Complex sentences

Complex formulae correspond to combinations of atomic sentences;

- “Popocatepetl is in Canada and the number of planets exceeds 7”
- “If Popocatepetl is in Canada then it lies north of Argentina”

Complex sentences

Complex formulae correspond to combinations of atomic sentences;

- "Popocatepetl is in Canada and the number of planets exceeds 7"
- "If Popocatepetl is in Canada then it lies north of Argentina"

Things to note

- The internal structure of atomic sentences remains unanalyzed.

Things to note

- The internal structure of atomic sentences remains unanalyzed.
- The unit of analysis is a complete, true or false sentence.

Things to note

- The internal structure of atomic sentences remains unanalyzed.
- The unit of analysis is a complete, true or false sentence.
- A sentence mirrors a complete state of affairs,

Things to note

- The internal structure of atomic sentences remains unanalyzed.
- The unit of analysis is a complete, true or false sentence.
- A sentence mirrors a complete state of affairs,
  - that so-and-so happens to be the case, not
  - how or in virtue of what so-and-so happens to be the case.

Things to note

- The internal structure of atomic sentences remains unanalyzed.
- The unit of analysis is a complete, true or false sentence.
- A sentence mirrors a complete state of affairs,
    - that so-and-so happens to be the case, not
    - how or in virtue of what so-and-so happens to be the case.
- Sets of formulae represent possible configurations of facts, that is

Things to note

- The internal structure of atomic sentences remains unanalyzed.
- The unit of analysis is a complete, true or false sentence.
- A sentence mirrors a complete state of affairs,
    - that so-and-so happens to be the case, not
    - how or in virtue of what so-and-so happens to be the case.
- Sets of formulae represent possible configurations of facts, that is
    - possible states of the world,

Things to note

- The internal structure of atomic sentences remains unanalyzed.
- The unit of analysis is a complete, true or false sentence.
- A sentence mirrors a complete state of affairs,
  - that so-and-so happens to be the case, not
  - how or in virtue of what so-and-so happens to be the case.
- Sets of formulae represent possible configurations of facts, that is
  - possible states of the world,
  - corresponding to possible assignments of truth-values to sentences

Things to note

- The internal structure of atomic sentences remains unanalyzed.
- The unit of analysis is a complete, true or false sentence.
- A sentence mirrors a complete state of affairs,
    - that so-and-so happens to be the case, not
    - how or in virtue of what so-and-so happens to be the case.
- Sets of formulae represent possible configurations of facts, that is
    - possible states of the world,
    - corresponding to possible assignments of truth-values to sentences
- Hence, an interpretation of a formula is simply

Things to note

- The internal structure of atomic sentences remains unanalyzed.
- The unit of analysis is a complete, true or false sentence.
- A sentence mirrors a complete state of affairs,
  - that so-and-so happens to be the case, not
  - how or in virtue of what so-and-so happens to be the case.
- Sets of formulae represent possible configurations of facts, that is
  - possible states of the world,
  - corresponding to possible assignments of truth-values to sentences
- Hence, an interpretation of a formula is simply
  - an interpretation of its atomic sentences in terms of truth and falsity,

Things to note

- The internal structure of atomic sentences remains unanalyzed.
- The unit of analysis is a complete, true or false sentence.
- A sentence mirrors a complete state of affairs,
    - that so-and-so happens to be the case, not
    - how or in virtue of what so-and-so happens to be the case.
- Sets of formulae represent possible configurations of facts, that is
    - possible states of the world,
    - corresponding to possible assignments of truth-values to sentences
- Hence, an interpretation of a formula is simply
    - an interpretation of its atomic sentences in terms of truth and falsity,
    - that determines the truth-value of the formula as a whole

Propositional semantics

Truth tables

Truth-tables give the meaning of the logical constants:

P_1	P_2	$P_1 \wedge P_2$	$P_1 \vee P_2$	$\neg P_1$
F	F	F	F	T
F	T	F	T	T
T	T	T	T	F
T	F	F	T	F

Propositional semantics

Truth tables

Truth-tables give the meaning of the logical constants:

P_1	P_2	$P_1 \land P_2$	$P_1 \lor P_2$	$\neg P_1$
F	F	F	F	T
F	T	F	T	T
T	T	T	T	F
T	F	F	T	F

Valuations/interpretations/models

A propositional model/interpretation, usually called a valuation, is a function v

- on the set of all formulae,
- into the set $\{T, F\}$,
- that assigns values corresponding to one row in the truth-table

Satisfaction/truth in a model

Satisfiability

- A valuation v satisifes a formula P if $v(P) = T$.
- A formula P is satisfiable if $v(P) = T$ for some model/valuation/interpretation v.
- Intuitively P is satisfiable if it describes a possible configuration.

Example

The formula $P_1 \vee P_2$ is satisfiable:

- It is satisfied by all valuations v such that $v(P_1) = T$, and
- by all valuations v' (possibly the same) such that $v'(P_2) = T$

Validity and entailment

Validity and entailment quantify over the set of all valuations:

Validity and entailment

Validity and entailment quantify over the set of all valuations:

Validity of a formula

- A formula P is valid iff $v(P) = T$ for all v.
- More generally: A formula P is valid in a class of models $\mathcal{M}$ iff $v(P) = T$ for all $v \in \mathcal{M}$

Validity and entailment

Validity and entailment quantify over the set of all valuations:

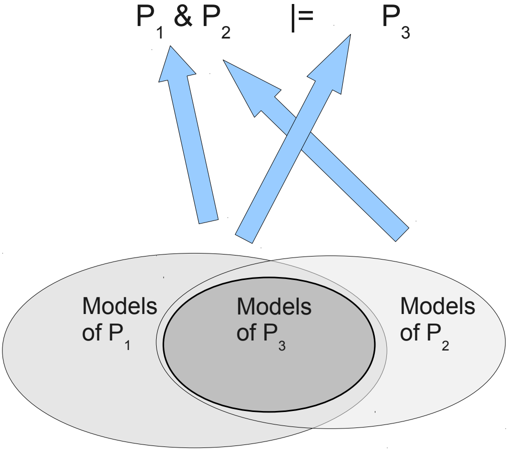
Validity of a formula

- A formula P is valid iff $v(P) = T$ for all v.
- More generally: A formula P is valid in a class of models $\mathcal{M}$ iff $v(P) = T$ for all $v \in \mathcal{M}$

Validity of an inference/entailment

- A set of sentences A entails a formula P, written $A \vDash P$, iff there is no valuation v such that $v(P) = T$ for all $P \in A$ and $v(P) = F$.

Entailment/validity illustrated



More things to note

- Models differ in their particular makeup from logic to logic, but
  - satisfaction,
  - validity, and
  - entailment,

  are largely invariant.
- The concept of satisfaction, i.e. of truth, is the fundamental one.
  - we shall thus have to define the truth of a triple
- Classical semantics is open world in the sense that
  - one model does not in general suffices to draw conclusions,
  - i. e. one model, or set of facts, cannot be assumed to contain complete knowledge
  - we shall come back to this

Outline

1. Why we need semantics
2. Model-theoretic semantics from a birds-eye perspective
3. Recalling classical consequence
4. RDF semantics—main features

Taking the structure of triples into account

Unlike propositions triples have parts, namely:

Taking the structure of triples into account

Unlike propositions triples have parts, namely:

- subject
- predicates, and
- objects

Taking the structure of triples into account

Unlike propositions triples have parts, namely:

- subject
- predicates, and
- objects

Less abstractly, these may be:

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

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

On what there is: Resources

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

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:

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,

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,

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

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.

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:

Resource contd.

Resources are always named by URIs. Examples:

- `http://purl.org/dc/terms/created`
    - names the concept of a creation date.

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.

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

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

Properties

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

Properties are always named by URIs. Examples.

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,

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,

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

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

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.

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"

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"

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.

RDF interpretations in outline

RDF interpretations

An RDF interpretation I of a vocabulary V is defined (in part) by

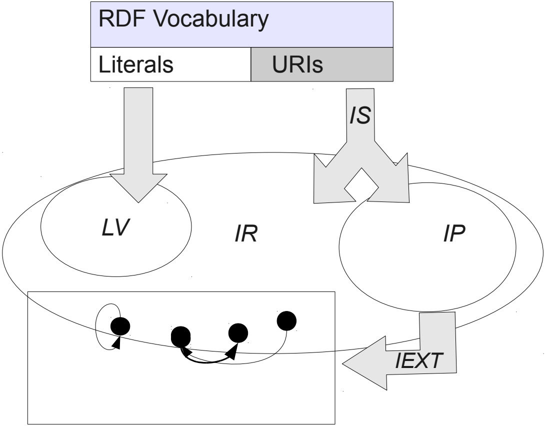
What there is:

- A non-empty set IR of resources, called the domain of I
- A subset $IP \subseteq IR$, called the set of properties of I
- A set $LV \subseteq IR$ of plain literals

The reference or meaning of words in the vocabulary, given by:

- A function IS from URIs in V into IR
- A function $IEXT$ from IP to $IR \times IR$
- Untyped literal values refer to themselves.

Interpretations

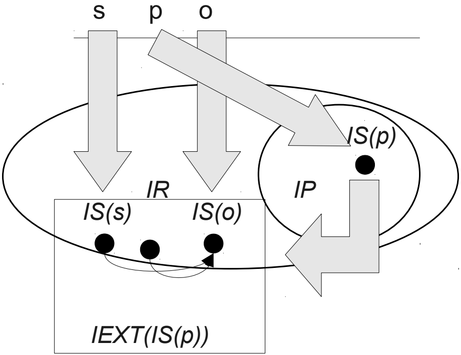


Satisfaction

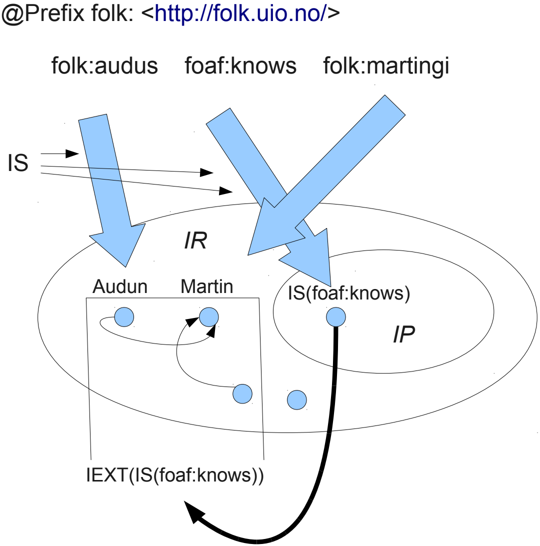
Truth of a triple in an interpretation

An RDF interpretation I satisfies a ground triple `s p o` if

- $\langle IS(s), IS(o)\rangle \in IEXT(IS(p))$



Satisfaction: A somewhat more concrete example

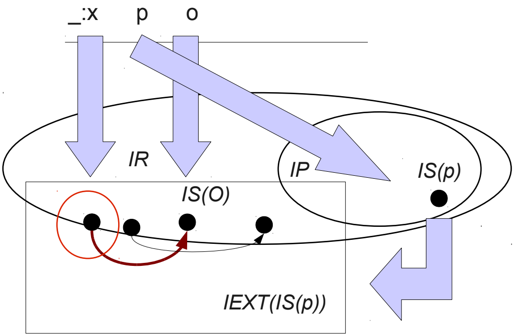


Interpretation of blank nodes

Interpretation of triples containing blank nodes

Let E be any RDF-triple. An interpretation I satisfies E if there is some substitution σ from the blank nodes in E into $IR - IP$, such that

- The ground triple $\sigma(E)$ is satisfied by I



Hang on !?

For those well-versed in FOL all this looks pretty standard:

Hang on !?

For those well-versed in FOL all this looks pretty standard:

- names or constants are interpreted as objects from the domain,

Hang on !?

For those well-versed in FOL all this looks pretty standard:

- names or constants are interpreted as objects from the domain,
- properties are interpreted as relations over the domain, and

Hang on !?

For those well-versed in FOL all this looks pretty standard:

- names or constants are interpreted as objects from the domain,
- properties are interpreted as relations over the domain, and
- satisfaction is those object's being in the relation as a pair.

Hang on !?

For those well-versed in FOL all this looks pretty standard:

- names or constants are interpreted as objects from the domain,
- properties are interpreted as relations over the domain, and
- satisfaction is those object's being in the relation as a pair.

But there is a very important twist:

Hang on !?

For those well-versed in FOL all this looks pretty standard:

- names or constants are interpreted as objects from the domain,
- properties are interpreted as relations over the domain, and
- satisfaction is those object's being in the relation as a pair.

But there is a very important twist:

- Properties are interpreted twice.

Hang on !?

For those well-versed in FOL all this looks pretty standard:

- names or constants are interpreted as objects from the domain,
- properties are interpreted as relations over the domain, and
- satisfaction is those object's being in the relation as a pair.

But there is a very important twist:

- Properties are interpreted twice.
- A property name p is first mapped to a resource $IS(p)$ in IP

Hang on !?

For those well-versed in FOL all this looks pretty standard:

- names or constants are interpreted as objects from the domain,
- properties are interpreted as relations over the domain, and
- satisfaction is those object's being in the relation as a pair.

But there is a very important twist:

- Properties are interpreted twice.
- A property name p is first mapped to a resource $IS(p)$ in IP
- and *then* it is mapped to a relation $IEXT(\ IS(p)\ )$ over $IP \cup IR$.

Hang on !?

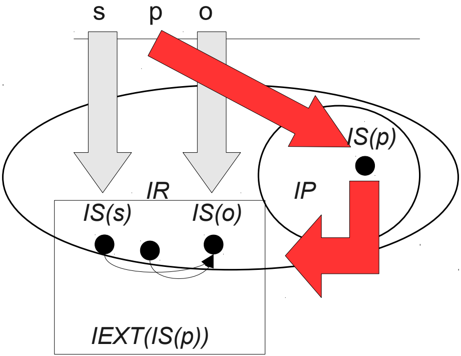
For those well-versed in FOL all this looks pretty standard:

- names or constants are interpreted as objects from the domain,
- properties are interpreted as relations over the domain, and
- satisfaction is those object's being in the relation as a pair.

But there is a very important twist:

- Properties are interpreted twice.
- A property name p is first mapped to a resource $IS(p)$ in IP
- and *then* it is mapped to a relation $IEXT(\ IS(p)\ )$ over $IP \cup IR$.
- So properties act as both objects and relations !?

Another look at satisfaction



The RDF meta-modelling architecture

The double interpretation of properties enables

The RDF meta-modelling architecture

The double interpretation of properties enables

- the RDF language to talk about itself,

The RDF meta-modelling architecture

The double interpretation of properties enables

- the RDF language to talk about itself,
- properties may participate in their own extensions,

The RDF meta-modelling architecture

The double interpretation of properties enables

- the RDF language to talk about itself,
- properties may participate in their own extensions,
- i.e. they may be used to restrict their own use

The RDF meta-modelling architecture

The double interpretation of properties enables

- the RDF language to talk about itself,
- properties may participate in their own extensions,
- i.e. they may be used to restrict their own use

It is therefore possible to define higher-level languages in RDF itself.

The RDF meta-modelling architecture

The double interpretation of properties enables

- the RDF language to talk about itself,
- properties may participate in their own extensions,
- i.e. they may be used to restrict their own use

It is therefore possible to define higher-level languages in RDF itself.

Some RDFS axioms

```
rdfs:domain rdfs:domain rdf:Property
```

The RDF meta-modelling architecture

The double interpretation of properties enables

- the RDF language to talk about itself,
- properties may participate in their own extensions,
- i.e. they may be used to restrict their own use

It is therefore possible to define higher-level languages in RDF itself.

Some RDFS axioms

```
rdfs:domain rdfs:domain rdf:Property

rdfs:subPropertyOf rdfs:domain rdf:Property
```

The RDF meta-modelling architecture

The double interpretation of properties enables

- the RDF language to talk about itself,
- properties may participate in their own extensions,
- i.e. they may be used to restrict their own use

It is therefore possible to define higher-level languages in RDF itself.

Some RDFS axioms

```
rdfs:domain rdfs:domain rdf:Property

rdfs:subPropertyOf rdfs:domain rdf:Property

rdf:type rdfs:range rdfs:Class
```

The RDF meta-modelling architecture

The double interpretation of properties enables

- the RDF language to talk about itself,
- properties may participate in their own extensions,
- i.e. they may be used to restrict their own use

It is therefore possible to define higher-level languages in RDF itself.

Some RDFS axioms

```
rdfs:domain rdfs:domain rdf:Property
rdfs:subPropertyOf rdfs:domain rdf:Property
rdf:type rdfs:range rdfs:Class
```

Enabling effects of the meta-modelling architecture

- Both RDFS and OWL are RDF vocabularies, i. e.

Enabling effects of the meta-modelling architecture

- Both RDFS and OWL are RDF vocabularies, i. e.
  - they are defined in RDF and can be treated as plain RDF.

Enabling effects of the meta-modelling architecture

- Both RDFS and OWL are RDF vocabularies, i. e.
  - they are defined in RDF and can be treated as plain RDF.
- No divide between higher and lower levels in the Semantic Web stack:

Enabling effects of the meta-modelling architecture

- Both RDFS and OWL are RDF vocabularies, i. e.
    - they are defined in RDF and can be treated as plain RDF.
- No divide between higher and lower levels in the Semantic Web stack:
    - OWL and RDFS can be queried as plain RDF with SPARQL,

Enabling effects of the meta-modelling architecture

- Both RDFS and OWL are RDF vocabularies, i. e.
  - they are defined in RDF and can be treated as plain RDF.
- No divide between higher and lower levels in the Semantic Web stack:
  - OWL and RDFS can be queried as plain RDF with SPARQL,
  - OWL and RDFS can use RDF parsers,

Enabling effects of the meta-modelling architecture

- Both RDFS and OWL are RDF vocabularies, i. e.
    - they are defined in RDF and can be treated as plain RDF.
- No divide between higher and lower levels in the Semantic Web stack:
    - OWL and RDFS can be queried as plain RDF with SPARQL,
    - OWL and RDFS can use RDF parsers,
    - ontologies can be persisted in triple stores,

Enabling effects of the meta-modelling architecture

- Both RDFS and OWL are RDF vocabularies, i. e.
  - they are defined in RDF and can be treated as plain RDF.
- No divide between higher and lower levels in the Semantic Web stack:
  - OWL and RDFS can be queried as plain RDF with SPARQL,
  - OWL and RDFS can use RDF parsers,
  - ontologies can be persisted in triple stores,
  - plain RDF graphs can be treated as data for OWL ontologies

Limitative results of the meta-modelling architecture

RDFS (the RDF schema language) does not have a classical extensional semantics:

Limitative results of the meta-modelling architecture

RDFS (the RDF schema language) does not have a classical extensional semantics:

- Two `rdfs properties` may have the same extension and not be equal,

Limitative results of the meta-modelling architecture

RDFS (the RDF schema language) does not have a classical extensional semantics:

- Two `rdfs properties` may have the same extension and not be equal,
- Two `rdfs classes` may be subsets of one another and not be equal,

Limitative results of the meta-modelling architecture

RDFS (the RDF schema language) does not have a classical extensional semantics:

- Two `rdfs properties` may have the same extension and not be equal,
- Two `rdfs classes` may be subsets of one another and not be equal,

OWL, as we shall see, adopts an extensional semantics, so

Limitative results of the meta-modelling architecture

RDFS (the RDF schema language) does not have a classical extensional semantics:

- Two `rdfs properties` may have the same extension and not be equal,
- Two `rdfs classes` may be subsets of one another and not be equal,

OWL, as we shall see, adopts an extensional semantics, so

- An `owl:Class` is not the same as an `rdf:Class`,

Limitative results of the meta-modelling architecture

RDFS (the RDF schema language) does not have a classical extensional semantics:

- Two `rdfs properties` may have the same extension and not be equal,
- Two `rdfs classes` may be subsets of one another and not be equal,

OWL, as we shall see, adopts an extensional semantics, so

- An `owl:Class` is not the same as an `rdf:Class`,
- RDFS inference engines cannot in general do OWL reasoning,

Limitative results of the meta-modelling architecture

RDFS (the RDF schema language) does not have a classical extensional semantics:

- Two `rdfs properties` may have the same extension and not be equal,
- Two `rdfs classes` may be subsets of one another and not be equal,

OWL, as we shall see, adopts an extensional semantics, so

- An `owl:Class` is not the same as an `rdf:Class`,
- RDFS inference engines cannot in general do OWL reasoning,
- nor vice versa

Limitative results of the meta-modelling architecture

RDFS (the RDF schema language) does not have a classical extensional semantics:

- Two `rdfs properties` may have the same extension and not be equal,
- Two `rdfs classes` may be subsets of one another and not be equal,

OWL, as we shall see, adopts an extensional semantics, so

- An `owl:Class` is not the same as an `rdf:Class`,
- RDFS inference engines cannot in general do OWL reasoning,
- nor vice versa
- although, they will usually not crash, just ignore information

Limitative results of the meta-modelling architecture

RDFS (the RDF schema language) does not have a classical extensional semantics:

- Two `rdfs properties` may have the same extension and not be equal,
- Two `rdfs classes` may be subsets of one another and not be equal,

OWL, as we shall see, adopts an extensional semantics, so

- An `owl:Class` is not the same as an `rdf:Class`,
- RDFS inference engines cannot in general do OWL reasoning,
- nor vice versa
- although, they will usually not crash, just ignore information

RDF entailment is undecidable in general. OWL-full, for instance

Limitative results of the meta-modelling architecture

RDFS (the RDF schema language) does not have a classical extensional semantics:

- Two `rdfs properties` may have the same extension and not be equal,
- Two `rdfs classes` may be subsets of one another and not be equal,

OWL, as we shall see, adopts an extensional semantics, so

- An `owl:Class` is not the same as an `rdf:Class`,
- RDFS inference engines cannot in general do OWL reasoning,
- nor vice versa
- although, they will usually not crash, just ignore information

RDF entailment is undecidable in general. OWL-full, for instance

- combines OWL-expressivity with RDFS metamodeling,

Limitative results of the meta-modelling architecture

RDFS (the RDF schema language) does not have a classical extensional semantics:

- Two `rdfs properties` may have the same extension and not be equal,
- Two `rdfs classes` may be subsets of one another and not be equal,

OWL, as we shall see, adopts an extensional semantics, so

- An `owl:Class` is not the same as an `rdf:Class`,
- RDFS inference engines cannot in general do OWL reasoning,
- nor vice versa
- although, they will usually not crash, just ignore information

RDF entailment is undecidable in general. OWL-full, for instance

- combines OWL-expressivity with RDFS metamodeling,
- and is for that reason undecidable

Open and closed world reasoning

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

RDF-entailment

An RDF graph G entails a graph E if every interpretation I that satisfies G also satisfies E.

Just as with propositional semantics, therefore:

Open and closed world reasoning

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

RDF-entailment

An RDF graph G entails a graph E if every interpretation I that satisfies G also satisfies E.

Just as with propositional semantics, therefore:

- one model does not in general suffice to decide entailment,

Open and closed world reasoning

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

RDF-entailment

An RDF graph G entails a graph E if every interpretation I that satisfies G also satisfies E.

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,

Open and closed world reasoning

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

RDF-entailment

An RDF graph G entails a graph E if every interpretation I that satisfies G also satisfies E.

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:

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

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.

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",

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 althogh we cannot find information about such an album,

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 althogh we cannot find information about such an album,
- or we may find a similarly named album by another band,

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 althogh 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.

Ramifications of the closed world assumption

Open world semantics becomes an issue for negative information.

- Imagine a relational database for an airline's flights:

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,

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.

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:

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

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:

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 be proved to be true is assumed false,

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 be proved to be true is assumed false,
- Negation as failure characterises;

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 be proved to be true is assumed false,
- Negation as failure characterises;
  - Negation in logic programming, e.g. Prolog.

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 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,

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 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.

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 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:

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.

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.

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 but Jupiter, then there are 8 planets.

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 but Jupiter, then there are 8 planets.

You do not want this behaviour from SPARQL:

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 but Jupiter, then there are 8 planets.

You do not want this behaviour from SPARQL:

- If you merge information from more sources, Jupiter may show up.

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 but Jupiter, then there are 8 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.

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 but Jupiter, then there are 8 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

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 but Jupiter, then there are 8 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

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 but Jupiter, then there are 8 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,

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 but Jupiter, then there are 8 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

The non-unique names assumption

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

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

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;

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 adress,

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 adress,
- we cannot conclude that the houshold has at least 5 members.

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 adress,
- we cannot conclude that the houshold has at least 5 members.

In order to make such inference we must;

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 adress,
- we cannot conclude that the houshold has at least 5 members.

In order to make such inference we must;

- explicitly state which names denote different objects,

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 adress,
- we cannot conclude that the houshold has at least 5 members.

In order to make such inference we must;

- explicitly state which names denote different objects,
- with `owl:differentFrom`,

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 adress,
- we cannot conclude that the houshold 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 7.

Take aways

Take aways

1. Model-theoretic semantics yields an unambigous notion of entailment,

Take aways

1. Model-theoretic semantics yields an unambigous notion of entailment,
2. which is necessary in order to liberate data from applications.

Take aways

1. Model-theoretic semantics yields an unambigous notion of entailment,
2. which is necessary in order to liberate data from applications.
3. RDF semantics has two important characteristics:

Take aways

1. Model-theoretic semantics yields an unambigous notion of entailment,
2. which is necessary in order to liberate data from applications.
3. RDF semantics has two important characteristics:
    1. Open world semantics, and

Take aways

1. Model-theoretic semantics yields an unambigous notion of entailment,
2. which is necessary in order to liberate data from applications.
3. RDF semantics has two important characteristics:
    1. Open world semantics, and
    2. the double interpretation of properties

Take aways

1. Model-theoretic semantics yields an unambigous notion of entailment,
2. which is necessary in order to liberate data from applications.
3. RDF semantics has two important characteristics:
    1. Open world semantics, and
    2. the double interpretation of properties
4. The double interpretation of properties

Take aways

1. Model-theoretic semantics yields an unambigous notion of entailment,
2. which is necessary in order to liberate data from applications.
3. RDF semantics has two important characteristics:
    1. Open world semantics, and
    2. the double interpretation of properties
4. The double interpretation of properties
    1. makes RDFS and OWL definable in RDF, but

Take aways

1. Model-theoretic semantics yields an unambigous notion of entailment,
2. which is necessary in order to liberate data from applications.
3. RDF semantics has two important characteristics:
    1. Open world semantics, and
    2. the double interpretation of properties
4. The double interpretation of properties
    1. makes RDFS and OWL definable in RDF, but
    2. makes RDF entailment undecidable in the general case.

Take aways

1. Model-theoretic semantics yields an unambigous notion of entailment,
2. which is necessary in order to liberate data from applications.
3. RDF semantics has two important characteristics:
    1. Open world semantics, and
    2. the double interpretation of properties
4. The double interpretation of properties
    1. makes RDFS and OWL definable in RDF, but
    2. makes RDF entailment undecidable in the general case.
5. Open world semantics

Take aways

1. Model-theoretic semantics yields an unambigous notion of entailment,
2. which is necessary in order to liberate data from applications.
3. RDF semantics has two important characteristics:
    1. Open world semantics, and
    2. the double interpretation of properties
4. The double interpretation of properties
    1. makes RDFS and OWL definable in RDF, but
    2. makes RDF entailment undecidable in the general case.
5. Open world semantics
    - is required by the open nature of the Web,

Take aways

1. Model-theoretic semantics yields an unambigous notion of entailment,
2. which is necessary in order to liberate data from applications.
3. RDF semantics has two important characteristics:
    1. Open world semantics, and
    2. the double interpretation of properties
4. The double interpretation of properties
    1. makes RDFS and OWL definable in RDF, but
    2. makes RDF entailment undecidable in the general case.
5. Open world semantics
    - is required by the open nature of the Web,
    - but makes classical negation of little use in queries.

Take aways

1. Model-theoretic semantics yields an unambigous notion of entailment,
2. which is necessary in order to liberate data from applications.
3. RDF semantics has two important characteristics:
    1. Open world semantics, and
    2. the double interpretation of properties
4. The double interpretation of properties
    1. makes RDFS and OWL definable in RDF, but
    2. makes RDF entailment undecidable in the general case.
5. Open world semantics
    - is required by the open nature of the Web,
    - 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