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.

Learning goals:

1. To understand the basic concepts of model-theoretic semantics.
2. To understand a simple semantics for parts of RDF/RDFS

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

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

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

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:

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,

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

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

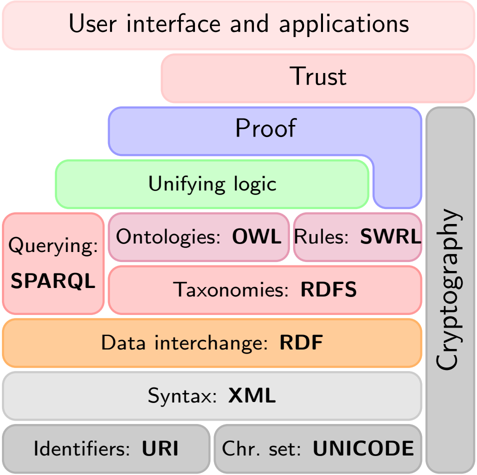


Figure: Semantic Web Stack

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 later lectures

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,

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

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

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.

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: 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.
}
```

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

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

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

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?

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.

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`?

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?

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

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

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:

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

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:

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

Where each line represents an application of an inference rule.

- How do we know that the inference rules are well-chosen?

Derivations

A proof typically looks something like this:

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

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 Repetition: Propositional Logic

4 Simplified RDF semantics

5 Open World Semantics

Propositional Logic: Formulas

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

Propositional Logic: Formulas

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

1 Any letter p, q, r, ... is a formula

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*

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*

  - $(A \wedge B)$ is also a formula (read: “A and B”)

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”)

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*

  - $(A \wedge B)$ is also a formula (read: "A and B")
  - $(A \vee B)$ is also a formula (read: "A or B")
  - $\neg A$ is also a formula (read: "not A")

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*
  - $(A \wedge B)$ is also a formula (read: “A and B”)
  - $(A \vee B)$ 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.

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*
  - $(A \wedge B)$ is also a formula (read: "A and B")
  - $(A \vee B)$ 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: $p \quad (p \wedge \neg r) \quad (q \wedge \neg q) \quad ((p \vee \neg q) \wedge \neg p)$

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*
  - $(A \wedge B)$ is also a formula (read: "A and B")
  - $(A \vee B)$ 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: $p \quad (p \wedge \neg r) \quad (q \wedge \neg q) \quad ((p \vee \neg q) \wedge \neg p)$
- Formulas are just a kind of strings until now:

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

Propositional Logic: Formulas

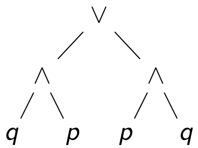
- Formulas are defined "by induction" or "recursively":

1 Any letter p, q, r, … is a formula

2 *if* A and B are formulas, *then*
  - $(A \wedge B)$ is also a formula (read: "A and B")
  - $(A \vee B)$ 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: $p \quad (p \wedge \neg r) \quad (q \wedge \neg q) \quad ((p \vee \neg q) \wedge \neg p)$
- Formulas are just a kind of strings until now:
  - no meaning
  - but every formula can be "parsed" uniquely.

$$((q \wedge p) \vee (p \wedge q))$$



Interpretations

- Logic is about truth and falsity

Interpretations

- Logic is about truth and falsity
- Truth of compound formulas depends on truth of letters.

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!

Interpretations

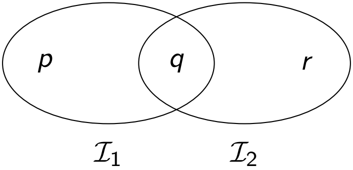
- Logic is about truth and falsity
- Truth of compound formulas depends on truth of letters.
- Idea: put all letters that are “true” into a set!
- Define: An *interpretation* $\mathcal{I}$ is a set of letters.

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 $p \in \mathcal{I}$.

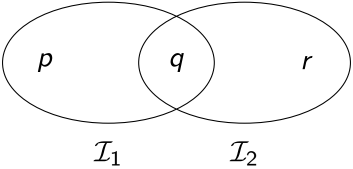
Interpretations

- 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.



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

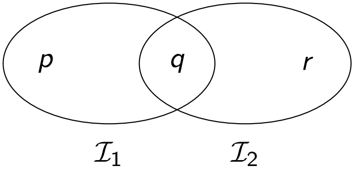
$$\mathcal{I} \models p$$

Semantic Validity $\models$

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

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

- For instance



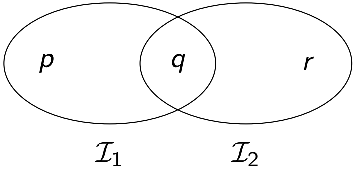
$$\mathcal{I}_1 \models p \qquad \mathcal{I}_2 \not\models p$$

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:

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

Validity of Compound Formulas

- Is $((q \wedge r) \vee (p \wedge q))$ true in $\mathcal{I}$?

Validity of Compound Formulas

- Is $((q \wedge r) \vee (p \wedge q))$ true in $\mathcal{I}$?
- Idea: apply our rule recursively

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,...

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 $\mathcal{I}$,...

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 $\mathcal{I} \models B$

Validity of Compound Formulas

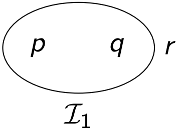
- Is $((q \wedge r) \vee (p \wedge q))$ true in $\mathcal{I}$?
- Idea: apply our rule recursively
- For any formulas A and B,...
- ...and any interpretation $\mathcal{I}$,...
  - ...$\mathcal{I} \models A \wedge B$ if and only if $\mathcal{I} \models A$ and $\mathcal{I} \models B$
  - ...$\mathcal{I} \models A \vee B$ if and only if $\mathcal{I} \models A$ or $\mathcal{I} \models B$ (or both)

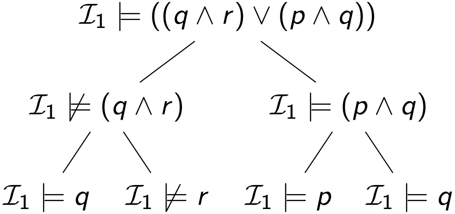
Validity of Compound Formulas

- 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 $\mathcal{I} \not\models A$.

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*

Tautologies

- A formula A that is true in *all* interpretations is called a *tautology*
- also *logically valid*

Tautologies

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

Tautologies

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

$$\models A$$

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

Tautologies

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

$$\models A$$

- $(p \lor \neg p)$ is a tautology
- True whatever p means:

Tautologies

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

$$\models A$$

- $(p \lor \neg p)$ is a tautology
- True whatever p means:
  - The sky is blue or the sky is not blue.

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.

Tautologies

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

$$\models A$$

- $(p \lor \neg p)$ 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.

Tautologies

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

$$\models A$$

- $(p \lor \neg p)$ 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...

Tautologies

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

$$\models A$$

- $(p \lor \neg p)$ 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!

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

Entailment

- Tautologies are true in all interpretations
- Some formulas are true only under certain assumptions

Entailment

- Tautologies are true in all interpretations
- Some formulas are true only under certain assumptions
- *A entails B*, written $A \models B$ if

Entailment

- Tautologies are true in all interpretations
- Some formulas are true only under certain assumptions
- *A entails B*, written $A \models B$ if

  $\mathcal{I} \models B$

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 $\mathcal{I} \models A$

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"

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

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:

$$p \land q \models p$$

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:

$$p \land q \models p$$

- Independent of meaning of p and q:

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:

$$p \land q \models p$$

- Independent of meaning of p and q:
  - If it rains and the sky is blue, then it rains

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

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

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

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

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.

Restricting RDF/RDFS

- We will simplify things by only looking at certain kinds of RDF graphs.

Restricting RDF/RDFS

- We will simplify things by only looking at certain kinds of RDF graphs.
- No triples "about" properties, classes, etc., except RDFS

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:

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`

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`

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.

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)

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:

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 .
```

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 .
```

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 .
```

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 .
```

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 .
```

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 .
```

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

Short Forms

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

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 .`	$\mathrm{rg}(r, C)$

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

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:

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 .
```

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:

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:

$loves(romeo, juliet)$
$Lady(juliet)$

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

Interpretations for RDF

- To interpret propositional formulas, we need to know how to interpret

Interpretations for RDF

- To interpret propositional formulas, we need to know how to interpret
    - Letters

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

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

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

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

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

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 $\mathcal{I}$

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 $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$

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 $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$

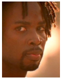
Interpretations for RDF

- 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 $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

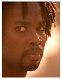
Interpretations for RDF

- 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.

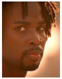
An example "intended" interpretation

- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$

An example "intended" interpretation

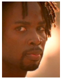
- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $romeo^{\mathcal{I}_1} =$  $juliet^{\mathcal{I}_1} =$ 

An example "intended" interpretation

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

  $Lover^{\mathcal{I}_1} = Beloved^{\mathcal{I}_1} = \{$ ,  $\}$

An example "intended" interpretation

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

  $Lover^{\mathcal{I}_1} = Beloved^{\mathcal{I}_1} = \{$ ,  $\}$
- $loves^{\mathcal{I}_1} = \{\langle$ ,  $\rangle, \langle$ ,  $\rangle\}$

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

An example “non-intended” interpretation

- $\Delta^{\mathcal{I}_2} = \mathbb{N} = \{1, 2, 3, 4, \ldots\}$

An example "non-intended" interpretation

- $\Delta^{\mathcal{I}_2} = \mathbb{N} = \{1, 2, 3, 4, \ldots\}$
- $\mathit{romeo}^{\mathcal{I}_2} = 17$
  $\mathit{juliet}^{\mathcal{I}_2} = 32$

An example "non-intended" interpretation

- $\Delta^{\mathcal{I}_2} = \mathbb{N} = \{1, 2, 3, 4, \ldots\}$
- $romeo^{\mathcal{I}_2} = 17$
  $juliet^{\mathcal{I}_2} = 32$
- $Lady^{\mathcal{I}_2} = \{2^n \mid n \in \mathbb{N}\} = \{2, 4, 8, 16, 32, \ldots\}$
  $Person^{\mathcal{I}_2} = \{2n \mid n \in \mathbb{N}\} = \{2, 4, 6, 8, 10, \ldots\}$
  $Lover^{\mathcal{I}_2} = Beloved^{\mathcal{I}_2} = \mathbb{N}$

An example "non-intended" interpretation

- $\Delta^{\mathcal{I}_2} = \mathbb{N} = \{1, 2, 3, 4, \ldots\}$
- $romeo^{\mathcal{I}_2} = 17$
  $juliet^{\mathcal{I}_2} = 32$
- $Lady^{\mathcal{I}_2} = \{2^n \mid n \in \mathbb{N}\} = \{2, 4, 8, 16, 32, \ldots\}$
  $Person^{\mathcal{I}_2} = \{2n \mid n \in \mathbb{N}\} = \{2, 4, 6, 8, 10, \ldots\}$
  $Lover^{\mathcal{I}_2} = Beloved^{\mathcal{I}_2} = \mathbb{N}$
- $loves^{\mathcal{I}_2} = < = \{\langle x, y\rangle \mid x < y\}$
  $knows^{\mathcal{I}_2} = \leq = \{\langle x, y\rangle \mid x \leq y\}$

An example "non-intended" interpretation

- $\Delta^{\mathcal{I}_2} = \mathbb{N} = \{1, 2, 3, 4, \ldots\}$
- $romeo^{\mathcal{I}_2} = 17$
  $juliet^{\mathcal{I}_2} = 32$
- $Lady^{\mathcal{I}_2} = \{2^n \mid n \in \mathbb{N}\} = \{2, 4, 8, 16, 32, \ldots\}$
  $Person^{\mathcal{I}_2} = \{2n \mid n \in \mathbb{N}\} = \{2, 4, 6, 8, 10, \ldots\}$
  $Lover^{\mathcal{I}_2} = Beloved^{\mathcal{I}_2} = \mathbb{N}$
- $loves^{\mathcal{I}_2} =<= \{\langle x, y\rangle \mid x < y\}$
  $knows^{\mathcal{I}_2} =\leq= \{\langle x, y\rangle \mid x \leq y\}$

- Just because names (URIs) look familiar, they don't need to denote what we think!

An example “non-intended” interpretation

- $\Delta^{\mathcal{I}_2} = \mathbb{N} = \{1, 2, 3, 4, \ldots\}$
- $romeo^{\mathcal{I}_2} = 17$
  $juliet^{\mathcal{I}_2} = 32$
- $Lady^{\mathcal{I}_2} = \{2^n \mid n \in \mathbb{N}\} = \{2, 4, 8, 16, 32, \ldots\}$
  $Person^{\mathcal{I}_2} = \{2n \mid n \in \mathbb{N}\} = \{2, 4, 6, 8, 10, \ldots\}$
  $Lover^{\mathcal{I}_2} = Beloved^{\mathcal{I}_2} = \mathbb{N}$
- $loves^{\mathcal{I}_2} = < = \{\langle x, y\rangle \mid x < y\}$
  $knows^{\mathcal{I}_2} = \leq = \{\langle x, y\rangle \mid x \leq y\}$

- 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:

Validity in Interpretations (RDF)

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models r(i_1, i_2)$ iff $\langle i_1^{\mathcal{I}}, i_2^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$

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 $i^{\mathcal{I}} \in C^{\mathcal{I}}$

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:

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

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

$\Big\langle$ ,  $\Big\rangle \in \mathit{loves}^{\mathcal{I}_1} = \Big\{ \Big\langle$ ,  $\Big\rangle, \Big\langle$ ,  $\Big\rangle \Big\}$

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

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

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

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
    $\langle$ ,  $\rangle \in \mathit{loves}^{\mathcal{I}_1} = \{ \langle$ ,  $\rangle, \langle$ ,  $\rangle \}$
  - $\mathcal{I}_1 \models \mathit{Person}(\mathit{romeo})$ 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 $\mathit{juliet}^{\mathcal{I}_2} = 32 \not< \mathit{romeo}^{\mathcal{I}_2} = 17$

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

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

Validity in Interpretations, cont. (RDFS)

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models C \subseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

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 $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$

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 $\mathrm{dom}\, r^{\mathcal{I}} \subseteq C^{\mathcal{I}}$

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 $\mathrm{rg}\, r^{\mathcal{I}} \subseteq C^{\mathcal{I}}$

Validity in Interpretations, cont. (RDFS)

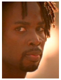
- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models C \subseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I} \models r \subseteq s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
- $\mathcal{I} \models \mathrm{dom}(r, C)$ iff $\mathrm{dom}\,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

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 $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
- $\mathcal{I} \models \text{dom}(r, C)$ iff $\text{dom}\, r^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
- $\mathcal{I} \models \text{rg}(r, C)$ iff $\text{rg}\, r^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
- Examples:
    - $\mathcal{I}_1 \models \mathit{Lover} \sqsubseteq \mathit{Person}$ because

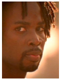
Validity in Interpretations, cont. (RDFS)

- 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:
  - $\mathcal{I}_1 \models Lover \sqsubseteq Person$ because

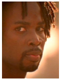
$Lover^{\mathcal{I}_1} = \{$ ,  $\} \subseteq Person^{\mathcal{I}_1} = \{$ , ,  $\}$

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

Validity in Interpretations, cont. (RDFS)

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I} \models r \sqsubseteq s$ 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:
  - $\mathcal{I}_1 \models Lover \sqsubseteq Person$ because
    $Lover^{\mathcal{I}_1} = \{$ ,  $\} \subseteq Person^{\mathcal{I}_1} = \{$ , ,  $\}$
  - $\mathcal{I}_2 \not\models Lover \sqsubseteq Person$ because
    $Lover^{\mathcal{I}_2} = \mathbb{N}$ and $Person^{\mathcal{I}_2} = \{2, 4, 6, 8, 10, \ldots\}$

Example: Range/Domain semantics

$$\mathcal{I}_2 \models \text{dom}(\textit{knows}, \textit{Beloved})$$

because. . .

Example: Range/Domain semantics

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

because. . .

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

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

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

Example: Range/Domain semantics

$$\mathcal{I}_2 \models \text{dom}(\textit{knows}, \textit{Beloved})$$

because...

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

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

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

Furthermore,

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

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 $\mathcal{I}$

Interpretation of Sets of Triples

- Given an interpretation $\mathcal{I}$
- And a set of triples $\mathcal{A}$ (any of the six kinds)

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

$$\mathcal{I} \models \mathcal{A}$$

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 $A \in \mathcal{A}$.

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 $\mathcal{A}$.

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:

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

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)

Entailment

- Given a set of triples $\mathcal{A}$ (any of the six kinds)
- And a further triple T (also any kind)

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 $\mathcal{A} \models T$

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

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 $\mathcal{I} \models \mathcal{A}$

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 $\mathcal{I} \models \mathcal{A}$
  - $\mathcal{I} \models T$.

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 $\mathcal{I} \models \mathcal{A}$

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:

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

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. . .

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 $\mathcal{I}$. . .

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 $\mathit{Lady}^{\mathcal{I}} \subseteq \mathit{Person}^{\mathcal{I}}$ ...

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:

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

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

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

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

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

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:

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

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

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:

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 although 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 although 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 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.

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 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 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 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 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 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 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 except for Jupiter, then there are 7 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 except for Jupiter, then there are 7 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 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.

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.

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

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

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,

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

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 address,

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.

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;

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,

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`,

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

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. 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

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,

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.

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