IN3060/4060 – Semantic Technologies – Spring 2021

Lecture 8: Model Semantics & Reasoning

Jieying Chen

5th March 2021

Department of Informatics

University of Oslo

Oblig 5

- Published today
- First delivery due 19th of March
- Final delivery 2 weeks after feedback
- Extra question for IN4060 students
- “Real” semantics of RDF and RDFS
- Foundations book: Section 3.2
- Still OK to ignore some complications, see oblig text
- We provide an excerpt of Sect. 3.2 with unimportant parts removed.
- Go to group sessions!

Today's Plan

1. Repetition: RDF semantics
2. Literal Semantics
3. Blank Node Semantics
4. Properties of Entailment by Model Semantics
5. Entailment and Derivability

Outline

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:subPropertyOf 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:subPropertyOf 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:subPropertyOf 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:

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

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

Interpretations for RDF

- To interpret the six kinds of triples, we need to know how to interpret

Interpretations for RDF

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

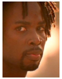
Interpretations for RDF

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

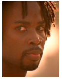
Interpretations for RDF

- To interpret 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.

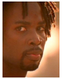
An example “intended” interpretation

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

An example “intended” interpretation

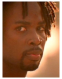
- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $romeo^{\mathcal{I}_1} =$  $juliet^{\mathcal{I}_1} =$ 

An example "intended" interpretation

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

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

An example "intended" interpretation

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

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

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

An example "non-intended" interpretation

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

An example "non-intended" interpretation

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

An example “non-intended” interpretation

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

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

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:

Validity in Interpretations

- 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

- 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

- 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 $i^{\mathcal{I}} \in C^{\mathcal{I}}$
  - $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

Validity in Interpretations

- 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 $i^{\mathcal{I}} \in C^{\mathcal{I}}$
  - $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
  - $\mathcal{I} \models r \sqsubseteq s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$

Validity in Interpretations

- 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 $i^{\mathcal{I}} \in C^{\mathcal{I}}$
  - $\mathcal{I} \models C \sqsubseteq D$ 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 $\mathrm{dom}\, r^{\mathcal{I}} \subseteq C^{\mathcal{I}}$

Validity in Interpretations

- 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 $i^{\mathcal{I}} \in C^{\mathcal{I}}$
  - $\mathcal{I} \models C \sqsubseteq D$ 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 $\mathrm{rg}\, r^{\mathcal{I}} \subseteq C^{\mathcal{I}}$

Validity in Interpretations

- 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 $i^{\mathcal{I}} \in C^{\mathcal{I}}$
  - $\mathcal{I} \models C \sqsubseteq D$ 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 $\mathrm{rg}\, r^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
- For a set of triples $\mathcal{A}$ (any of the six kinds)

Validity in Interpretations

- 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 $i^{\mathcal{I}} \in C^{\mathcal{I}}$
  - $\mathcal{I} \models C \sqsubseteq D$ 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 $\mathrm{rg}\, r^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
- For a set of triples $\mathcal{A}$ (any of the six kinds)
- $\mathcal{A}$ is valid in $\mathcal{I}$, written

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

Validity in Interpretations

- 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 $i^{\mathcal{I}} \in C^{\mathcal{I}}$
  - $\mathcal{I} \models C \sqsubseteq D$ 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 $\mathrm{rg}\, r^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
- For 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 $A \in \mathcal{A}$.

Validity Examples

- $\mathcal{I}_1 \models \mathit{loves}(\mathit{juliet}, \mathit{romeo})$ because

Validity 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 Examples

- $\mathcal{I}_1 \models \mathit{loves}(\mathit{juliet}, \mathit{romeo})$ because

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

- $\mathcal{I}_2 \not\models \mathit{Person}(\mathit{romeo})$ because

Validity Examples

- $\mathcal{I}_1 \models \mathit{loves}(\mathit{juliet}, \mathit{romeo})$ because

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

- $\mathcal{I}_2 \not\models \mathit{Person}(\mathit{romeo})$ because

  $\mathit{romeo}^{\mathcal{I}_2} = 17 \notin \mathit{Person}^{\mathcal{I}_2} = \{2, 4, 6, 8, 10, \ldots\}$

Validity Examples

- $\mathcal{I}_1 \models \mathit{loves}(\mathit{juliet}, \mathit{romeo})$ because

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

- $\mathcal{I}_2 \not\models \mathit{Person}(\mathit{romeo})$ because

  $\mathit{romeo}^{\mathcal{I}_2} = 17 \notin \mathit{Person}^{\mathcal{I}_2} = \{2, 4, 6, 8, 10, \ldots\}$

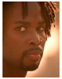
- $\mathcal{I}_1 \models \mathit{Lover} \sqsubseteq \mathit{Person}$ because

Validity Examples

- $\mathcal{I}_1 \models \textit{loves}(\textit{juliet}, \textit{romeo})$ because

  $\Big\langle$ ,  $\Big\rangle \in \textit{loves}^{\mathcal{I}_1} = \Big\{ \Big\langle$ ,  $\Big\rangle, \Big\langle$ ,  $\Big\rangle \Big\}$
- $\mathcal{I}_2 \not\models \textit{Person}(\textit{romeo})$ because

  $\textit{romeo}^{\mathcal{I}_2} = 17 \notin \textit{Person}^{\mathcal{I}_2} = \{2, 4, 6, 8, 10, \ldots\}$
- $\mathcal{I}_1 \models \textit{Lover} \sqsubseteq \textit{Person}$ because

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

Validity Examples

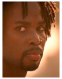
- $\mathcal{I}_1 \models \mathit{loves}(\mathit{juliet}, \mathit{romeo})$ because

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

- $\mathcal{I}_2 \not\models \mathit{Person}(\mathit{romeo})$ because

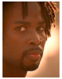
  $\mathit{romeo}^{\mathcal{I}_2} = 17 \notin \mathit{Person}^{\mathcal{I}_2} = \{2, 4, 6, 8, 10, \ldots\}$

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

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

- $\mathcal{I}_2 \not\models \mathit{Lover} \sqsubseteq \mathit{Person}$ because

Validity Examples

- $\mathcal{I}_1 \models loves(juliet, romeo)$ because
  $\langle$ ,  $\rangle \in loves^{\mathcal{I}_1} = \{\langle$ ,  $\rangle, \langle$ ,  $\rangle\}$
- $\mathcal{I}_2 \not\models Person(romeo)$ because
  $romeo^{\mathcal{I}_2} = 17 \notin Person^{\mathcal{I}_2} = \{2, 4, 6, 8, 10, \ldots\}$
- $\mathcal{I}_1 \models Lover \sqsubseteq Person$ 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 $Person^{\mathcal{I}_2} = \{2, 4, 6, 8, 10, \ldots\}$

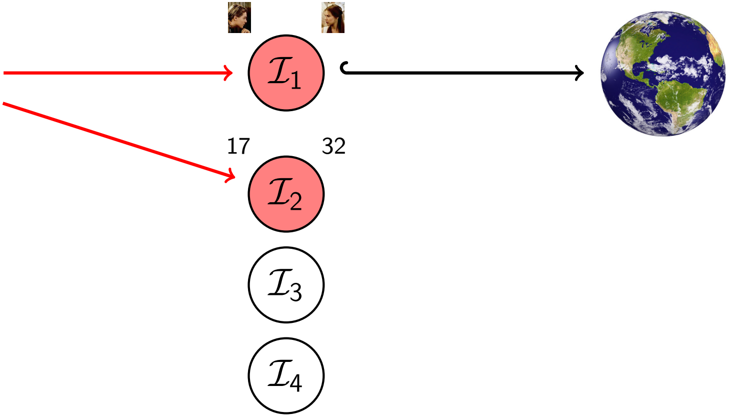
Finding out stuff about Romeo and Juliet

Statements

loves(*romeo*, *juliet*)
Lady(*juliet*)
Lady ⊑ *Person*
loves ⊑ *knows*
dom(*loves*, *Lover*)
rg(*loves*, *Beloved*)

Interpretations

The "Real World"



Finding out stuff about Romeo and Juliet

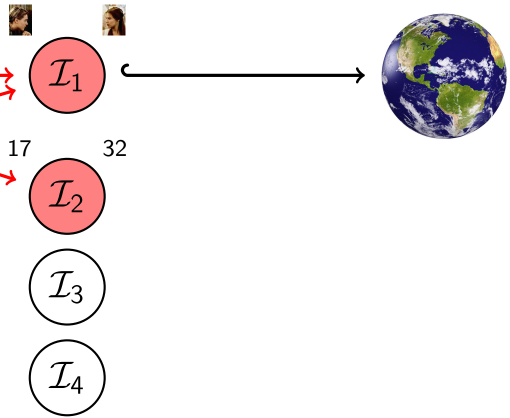
Statements

loves(*romeo*, *juliet*)
Lady(*juliet*)
Lady $\sqsubseteq$ *Person*
loves $\sqsubseteq$ *knows*
dom(*loves*, *Lover*)
rg(*loves*, *Beloved*)

loves(*juliet*, *romeo*)

Interpretations

The "Real World"



Finding out stuff about Romeo and Juliet

Statements

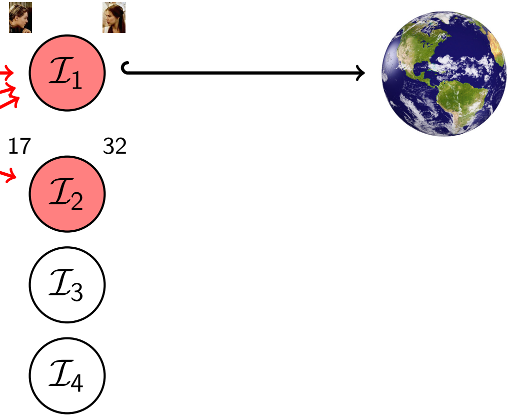
$\textit{loves}(\textit{romeo}, \textit{juliet})$
$\textit{Lady}(\textit{juliet})$
$\textit{Lady} \sqsubseteq \textit{Person}$
$\textit{loves} \sqsubseteq \textit{knows}$
$\mathbf{dom}(\textit{loves}, \textit{Lover})$
$\mathbf{rg}(\textit{loves}, \textit{Beloved})$

$\textit{loves}(\textit{juliet}, \textit{romeo})$

$\textit{Lover} \sqsubseteq \textit{Person}$

Interpretations

The “Real World”



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 $\mathcal{I} \models \mathcal{A}$
    - $\mathcal{I} \models T$.
- 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 $\mathcal{I} \models \mathcal{A}$
  - $\mathcal{I} \models T$.
- 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 $\mathcal{I} \models \mathcal{A}$
  - $\mathcal{I} \models T$.
- 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 $\mathcal{I} \models \mathcal{A}$
  - $\mathcal{I} \models T$.
- 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 $\mathcal{I} \models \mathcal{A}$
    - $\mathcal{I} \models T$.
- 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 $\mathcal{I} \models \mathcal{A}$
  - $\mathcal{I} \models T$.
- 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 $\mathit{juliet}^{\mathcal{I}} \in \mathit{Person}^{\mathcal{I}}$

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 $\mathcal{I} \models \mathcal{A}$
  - $\mathcal{I} \models T$.
- 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 $\mathit{juliet}^{\mathcal{I}} \in \mathit{Person}^{\mathcal{I}}$
- *Not* about T being (intuitively) true or not

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 $\mathcal{I} \models \mathcal{A}$
  - $\mathcal{I} \models T$.
- 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 $\mathit{juliet}^{\mathcal{I}} \in \mathit{Person}^{\mathcal{I}}$
- *Not* about T being (intuitively) true or not
- Only about whether T is a *consequence* of $\mathcal{A}$

Countermodels

- If $\mathcal{A} \not\models T$,. . .

Countermodels

- If $\mathcal{A} \not\models T$,...
- then there is an $\mathcal{I}$ with

Countermodels

- If $\mathcal{A} \not\models T, \ldots$
- then there is an $\mathcal{I}$ with
  - $\mathcal{I} \models \mathcal{A}$

Countermodels

- If $\mathcal{A} \not\models T, \ldots$
- then there is an $\mathcal{I}$ with
  - $\mathcal{I} \models \mathcal{A}$
  - $\mathcal{I} \not\models T$

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 $\mathcal{A} \not\models T$

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)

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:

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)

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)

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)

Outline

Motivating example

- Consider again the set of triples $\mathcal{A}$:

Motivating example

- Consider again the set of triples $\mathcal{A}$:

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

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

Motivating example

- Consider again the set of triples $\mathcal{A}$:

  $$\begin{array}{l} \mathit{loves}(\mathit{romeo}, \mathit{juliet}) \\ \mathit{Lady}(\mathit{juliet}) \end{array}$$

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

- We can now say something about if $\mathcal{A}$ is valid in an interpretation $\mathcal{I}$

Motivating example

- Consider again the set of triples $\mathcal{A}$:

  $$\begin{array}{l} \mathit{loves}(\mathit{romeo}, \mathit{juliet}) \\ \mathit{Lady}(\mathit{juliet}) \end{array}$$

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

- We can now say something about if $\mathcal{A}$ is valid in an interpretation $\mathcal{I}$
- Say we add the triple $T = \mathit{age}(\mathit{juliet}, \texttt{"13"})$

Motivating example

- Consider again the set of triples $\mathcal{A}$:

  $$\begin{array}{l} \mathit{loves}(\mathit{romeo}, \mathit{juliet}) \\ \mathit{Lady}(\mathit{juliet}) \end{array}$$

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

- We can now say something about if $\mathcal{A}$ is valid in an interpretation $\mathcal{I}$
- Say we add the triple $T = \mathit{age}(\mathit{juliet}, \texttt{"13"})$
- Is this new set of triples valid in any of our interpretations $\mathcal{I}_1$ or $\mathcal{I}_2$, why?

Simplifying Literals

- Literals can only occur as *objects* of triples

Simplifying Literals

- Literals can only occur as *objects* of triples
- Have datatype, can be with or without language tag

Simplifying Literals

- Literals can only occur as *objects* of triples
- Have datatype, can be with or without language tag
- The same predicate can be used with literals and resources:

Simplifying Literals

- Literals can only occur as *objects* of triples
- Have datatype, can be with or without language tag
- The same predicate can be used with literals and resources:

```
ex:me ex:likes dbpedia:Berlin .
```

Simplifying Literals

- Literals can only occur as *objects* of triples
- Have datatype, can be with or without language tag
- The same predicate can be used with literals and resources:

```
ex:me ex:likes dbpedia:Berlin .
ex:me ex:likes "some string" .
```

Simplifying Literals

- Literals can only occur as *objects* of triples
- Have datatype, can be with or without language tag
- The same predicate can be used with literals and resources:

```
ex:me ex:likes dbpedia:Berlin .
ex:me ex:likes "some string" .
```

- We simplify things by:

Simplifying Literals

- Literals can only occur as *objects* of triples
- Have datatype, can be with or without language tag
- The same predicate can be used with literals and resources:

```
ex:me ex:likes dbpedia:Berlin .
ex:me ex:likes "some string" .
```

- We simplify things by:
  - considering only string literals without language tag, and

Simplifying Literals

- Literals can only occur as *objects* of triples
- Have datatype, can be with or without language tag
- The same predicate can be used with literals and resources:

```
ex:me ex:likes dbpedia:Berlin .
ex:me ex:likes "some string" .
```

- We simplify things by:
  - considering only string literals without language tag, and
  - allowing either resource objects *or* literal objects for any predicate

Simplifying Literals

- Literals can only occur as *objects* of triples
- Have datatype, can be with or without language tag
- The same predicate can be used with literals and resources:

```
ex:me ex:likes dbpedia:Berlin .
ex:me ex:likes "some string" .
```

- We simplify things by:
  - considering only string literals without language tag, and
  - allowing either resource objects *or* literal objects for any predicate
- Five types of resources:

Simplifying Literals

- Literals can only occur as *objects* of triples
- Have datatype, can be with or without language tag
- The same predicate can be used with literals and resources:

```
ex:me ex:likes dbpedia:Berlin .
ex:me ex:likes "some string" .
```

- We simplify things by:
  - considering only string literals without language tag, and
  - allowing either resource objects *or* literal objects for any predicate
- Five types of resources:
  - *Object Properties* like `foaf:knows`

Simplifying Literals

- Literals can only occur as *objects* of triples
- Have datatype, can be with or without language tag
- The same predicate can be used with literals and resources:

```
ex:me ex:likes dbpedia:Berlin .
ex:me ex:likes "some string" .
```

- We simplify things by:
  - considering only string literals without language tag, and
  - allowing either resource objects *or* literal objects for any predicate
- Five types of resources:
  - *Object Properties* like `foaf:knows`
  - *Datatype Properties* like `dc:title`, `foaf:name`

Simplifying Literals

- Literals can only occur as *objects* of triples
- Have datatype, can be with or without language tag
- The same predicate can be used with literals and resources:

```
ex:me ex:likes dbpedia:Berlin .
ex:me ex:likes "some string" .
```

- We simplify things by:
  - considering only string literals without language tag, and
  - allowing either resource objects *or* literal objects for any predicate
- Five types of resources:
  - *Object Properties* like `foaf:knows`
  - *Datatype Properties* like `dc:title`, `foaf:name`
  - *Classes* like `foaf:Person`

Simplifying Literals

- Literals can only occur as *objects* of triples
- Have datatype, can be with or without language tag
- The same predicate can be used with literals and resources:

```
ex:me ex:likes dbpedia:Berlin .
ex:me ex:likes "some string" .
```

- We simplify things by:
  - considering only string literals without language tag, and
  - allowing either resource objects *or* literal objects for any predicate
- Five types of resources:
  - *Object Properties* like `foaf:knows`
  - *Datatype Properties* like `dc:title`, `foaf:name`
  - *Classes* like `foaf:Person`
  - *Built-ins*, a fixed set including `rdf:type`, `rdfs:domain`, etc.

Simplifying Literals

- Literals can only occur as *objects* of triples
- Have datatype, can be with or without language tag
- The same predicate can be used with literals and resources:

```
ex:me ex:likes dbpedia:Berlin .
ex:me ex:likes "some string" .
```

- We simplify things by:
  - considering only string literals without language tag, and
  - allowing either resource objects *or* literal objects for any predicate
- Five types of resources:
  - *Object Properties* like `foaf:knows`
  - *Datatype Properties* like `dc:title`, `foaf:name`
  - *Classes* like `foaf:Person`
  - *Built-ins*, a fixed set including `rdf:type`, `rdfs:domain`, etc.
  - *Individuals* (all the rest, "usual" resources)

Simplifying Literals

- Literals can only occur as *objects* of triples
- Have datatype, can be with or without language tag
- The same predicate can be used with literals and resources:

```
ex:me ex:likes dbpedia:Berlin .
ex:me ex:likes "some string" .
```

- We simplify things by:
  - considering only string literals without language tag, and
  - allowing either resource objects *or* literal objects for any predicate
- Five types of resources:
  - *Object Properties* like `foaf:knows`
  - *Datatype Properties* like `dc:title`, `foaf:name`
  - *Classes* like `foaf:Person`
  - *Built-ins*, a fixed set including `rdf:type`, `rdfs:domain`, etc.
  - *Individuals* (all the rest, “usual” resources)
- Why? – simpler, object/datatype split is in OWL

Allowed triples

Allow only triples using object properties and datatype properties as intended

Triples	Abbreviation
`indi o-prop indi .`	$r(i_1, i_2)$
`indi d-prop "lit" .`	$a(i, l)$
`indi rdf:type class .`	$C(i_1)$
`class rdfs:subClassOf class .`	$C \sqsubseteq D$
`o-prop rdfs:subPropertyOf o-prop .`	$r \sqsubseteq s$
`d-prop rdfs:subPropertyOf d-prop .`	$a \sqsubseteq b$
`o-prop rdfs:domain class .`	$\mathrm{dom}(r, C)$
`o-prop rdfs:range class .`	$\mathrm{rg}(r, C)$

Interpretation with Literals

- Let Λ be the set of all literal values, i.e. all strings

Interpretation with Literals

- Let Λ be the set of all literal values, i.e. all strings
  - Chosen once and for all, same for all interpretations

Interpretation with Literals

- Let Λ be the set of all literal values, i.e. all strings
  - Chosen once and for all, same for all interpretations
- A *DL-interpretation* $\mathcal{I}$ consists of

Interpretation with Literals

- Let Λ be the set of all literal values, i.e. all strings
  - Chosen once and for all, same for all interpretations
- A *DL-interpretation* $\mathcal{I}$ consists of
  - A set $\Delta^{\mathcal{I}}$, called the *domain* of $\mathcal{I}$

Interpretation with Literals

- Let Λ be the set of all literal values, i.e. all strings
  - Chosen once and for all, same for all interpretations
- A *DL-interpretation* $\mathcal{I}$ consists of
  - A set $\Delta^{\mathcal{I}}$, called the *domain* of $\mathcal{I}$
  - Interpretations $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ as before

Interpretation with Literals

- Let Λ be the set of all literal values, i.e. all strings
  - Chosen once and for all, same for all interpretations
- A *DL-interpretation* $\mathcal{I}$ consists of
  - A set $\Delta^{\mathcal{I}}$, called the *domain* of $\mathcal{I}$
  - Interpretations $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ as before
  - For each datatype property URI a, a relation $a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Lambda$

Interpretation with Literals

- Let Λ be the set of all literal values, i.e. all strings
  - Chosen once and for all, same for all interpretations
- A *DL-interpretation* $\mathcal{I}$ consists of
  - A set $\Delta^{\mathcal{I}}$, called the *domain* of $\mathcal{I}$
  - Interpretations $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ as before
  - For each datatype property URI a, a relation $a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Lambda$
- Semantics:

Interpretation with Literals

- Let Λ be the set of all literal values, i.e. all strings
  - Chosen once and for all, same for all interpretations
- A *DL-interpretation* $\mathcal{I}$ consists of
  - A set $\Delta^{\mathcal{I}}$, called the *domain* of $\mathcal{I}$
  - Interpretations $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ as before
  - For each datatype property URI a, a relation $a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Lambda$
- Semantics:
  - $\mathcal{I} \models r(i_1, i_2)$ iff $\langle i_1^{\mathcal{I}}, i_2^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$ for object property r

Interpretation with Literals

- Let Λ be the set of all literal values, i.e. all strings
  - Chosen once and for all, same for all interpretations
- A *DL-interpretation* $\mathcal{I}$ consists of
  - A set $\Delta^{\mathcal{I}}$, called the *domain* of $\mathcal{I}$
  - Interpretations $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ as before
  - For each datatype property URI a, a relation $a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Lambda$
- Semantics:
  - $\mathcal{I} \models r(i_1, i_2)$ iff $\langle i_1^{\mathcal{I}}, i_2^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$ for object property r
  - $\mathcal{I} \models a(i, l)$ iff $\langle i^{\mathcal{I}}, l \rangle \in a^{\mathcal{I}}$ for datatype property a

Interpretation with Literals

- Let Λ be the set of all literal values, i.e. all strings
  - Chosen once and for all, same for all interpretations
- A *DL-interpretation* $\mathcal{I}$ consists of
  - A set $\Delta^{\mathcal{I}}$, called the *domain* of $\mathcal{I}$
  - Interpretations $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ as before
  - For each datatype property URI a, a relation $a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Lambda$
- Semantics:
  - $\mathcal{I} \models r(i_1, i_2)$ iff $\langle i_1^{\mathcal{I}}, i_2^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$ for object property r
  - $\mathcal{I} \models a(i, l)$ iff $\langle i^{\mathcal{I}}, l \rangle \in a^{\mathcal{I}}$ for datatype property a
  - $\mathcal{I} \models r \sqsubseteq s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ for object properties r, s

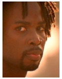
Interpretation with Literals

- Let Λ be the set of all literal values, i.e. all strings
  - Chosen once and for all, same for all interpretations
- A *DL-interpretation* $\mathcal{I}$ consists of
  - A set $\Delta^{\mathcal{I}}$, called the *domain* of $\mathcal{I}$
  - Interpretations $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ as before
  - For each datatype property URI a, a relation $a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Lambda$
- Semantics:
  - $\mathcal{I} \models r(i_1, i_2)$ iff $\langle i_1^{\mathcal{I}}, i_2^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$ for object property r
  - $\mathcal{I} \models a(i, l)$ iff $\langle i^{\mathcal{I}}, l \rangle \in a^{\mathcal{I}}$ for datatype property a
  - $\mathcal{I} \models r \sqsubseteq s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ for object properties r, s
  - $\mathcal{I} \models a \sqsubseteq b$ iff $a^{\mathcal{I}} \subseteq b^{\mathcal{I}}$ for datatype properties a, b

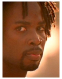
Interpretation with Literals

- Let Λ be the set of all literal values, i.e. all strings
  - Chosen once and for all, same for all interpretations
- A *DL-interpretation* $\mathcal{I}$ consists of
  - A set $\Delta^{\mathcal{I}}$, called the *domain* of $\mathcal{I}$
  - Interpretations $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ as before
  - For each datatype property URI a, a relation $a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Lambda$
- Semantics:
  - $\mathcal{I} \models r(i_1, i_2)$ iff $\langle i_1^{\mathcal{I}}, i_2^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$ for object property r
  - $\mathcal{I} \models a(i, l)$ iff $\langle i^{\mathcal{I}}, l \rangle \in a^{\mathcal{I}}$ for datatype property a
  - $\mathcal{I} \models r \sqsubseteq s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ for object properties r, s
  - $\mathcal{I} \models a \sqsubseteq b$ iff $a^{\mathcal{I}} \subseteq b^{\mathcal{I}}$ for datatype properties a, b
- Note: Literals l are in Λ, don't need to be interpreted.

Example: Interpretation with a Datatype Property

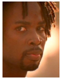
- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$

Example: Interpretation with a Datatype Property

- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $\mathit{loves}^{\mathcal{I}_1} = \{ \langle$ ,  $\rangle, \langle$ ,  $\rangle \}$

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

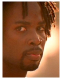
Example: Interpretation with a Datatype Property

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

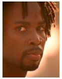
  $knows^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$
- $age^{\mathcal{I}_1} = \{\langle$ , `"16"` $\rangle, \langle$ , `"almost 14"` $\rangle, \langle$ , `"13"` $\rangle\}$

Outline

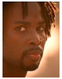
Motivating example

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

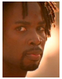
Motivating example

- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $\mathit{loves}^{\mathcal{I}_1} = \{ \langle$ ,  $\rangle, \langle$ ,  $\rangle \}$ $\qquad \mathit{knows}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$

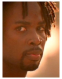
Motivating example

- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $\mathit{loves}^{\mathcal{I}_1} = \{\langle$ ,  $\rangle, \langle$ ,  $\rangle\}$ $\qquad \mathit{knows}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$
- $\mathit{age}^{\mathcal{I}_1} = \{\langle$ , $\texttt{"16"}\rangle, \langle$ , $\texttt{"almost 14"}\rangle, \langle$ , $\texttt{"13"}\rangle, \}$

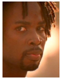
Motivating example

- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $\mathit{loves}^{\mathcal{I}_1} = \{ \langle$ ,  $\rangle, \langle$ ,  $\rangle \}$ $\qquad \mathit{knows}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$
- $\mathit{age}^{\mathcal{I}_1} = \{ \langle$ , `"16"` $\rangle, \langle$ , `"almost 14"` $\rangle, \langle$ , `"13"` $\rangle, \}$
- Let b_1 and b_2 be blank nodes

Motivating example

- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $\mathit{loves}^{\mathcal{I}_1} = \{\langle$ ,  $\rangle, \langle$ ,  $\rangle\}$ $\quad \mathit{knows}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$
- $\mathit{age}^{\mathcal{I}_1} = \{\langle$ , $\texttt{"16"}\rangle, \langle$ , $\texttt{"almost 14"}\rangle, \langle$ , $\texttt{"13"}\rangle, \}$
- Let b_1 and b_2 be blank nodes
- $\mathcal{A} = \{\mathit{age}(b_1, \texttt{"16"}), \mathit{loves}(b_1, b_2), \mathit{age}(b_2, \texttt{"13"})\}$

Motivating example

- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $loves^{\mathcal{I}_1} = \{ \langle$ ,  $\rangle, \langle$ ,  $\rangle \}$ $\quad knows^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$
- $age^{\mathcal{I}_1} = \{ \langle$ , $\texttt{"16"} \rangle, \langle$ , $\texttt{"almost 14"} \rangle, \langle$ , $\texttt{"13"} \rangle, \}$
- Let b_1 and b_2 be blank nodes
- $\mathcal{A} = \{age(b_1, \texttt{"16"}), loves(b_1, b_2), age(b_2, \texttt{"13"})\}$
- Is $\mathcal{A}$ valid in $\mathcal{I}_1$? why?

Blank Nodes

- Remember: Blank nodes are just like resources...

Blank Nodes

- Remember: Blank nodes are just like resources...
- ...but without a “global” URI.

Blank Nodes

- Remember: Blank nodes are just like resources...
- ... but without a "global" URI.
- Blank node has a local "blank node identifier" instead.

Blank Nodes

- Remember: Blank nodes are just like resources. . .
- . . . but without a "global" URI.
- Blank node has a local "blank node identifier" instead.

- A blank node *can* be used in several triples. . .

Blank Nodes

- Remember: Blank nodes are just like resources. . .
- . . . but without a "global" URI.
- Blank node has a local "blank node identifier" instead.

- A blank node *can* be used in several triples. . .
- . . . but they have to be in the same "file" or "data set"

Blank Nodes

- Remember: Blank nodes are just like resources. . .
- . . . but without a "global" URI.
- Blank node has a local "blank node identifier" instead.

- A blank node *can* be used in several triples. . .
- . . . but they have to be in the same "file" or "data set"
- Semantics of blank nodes require looking at a set of triples

Blank Nodes

- Remember: Blank nodes are just like resources. . .
- . . . but without a "global" URI.
- Blank node has a local "blank node identifier" instead.

- A blank node *can* be used in several triples. . .
- . . . but they have to be in the same "file" or "data set"
- Semantics of blank nodes require looking at a set of triples

- But we still need to interpret single triples.

Blank Nodes

- Remember: Blank nodes are just like resources...
- ...but without a "global" URI.
- Blank node has a local "blank node identifier" instead.

- A blank node *can* be used in several triples...
- ...but they have to be in the same "file" or "data set"
- Semantics of blank nodes require looking at a set of triples

- But we still need to interpret single triples.
- Solution: pass in blank node interpretation, deal with sets later!

Blank Node Valuations

- Given an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$...

Blank Node Valuations

- Given an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$...
  - A *blank node valuation* β...

Blank Node Valuations

- Given an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$...
  - A *blank node valuation* β...
  - ...gives a domain element or literal value $\beta(b) \in \Delta^{\mathcal{I}} \cup \Lambda$...

Blank Node Valuations

- Given an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$...
  - A *blank node valuation* β...
  - ...gives a domain element or literal value $\beta(b) \in \Delta^{\mathcal{I}} \cup \Lambda$...
  - ...for every blank node ID b

Blank Node Valuations

- Given an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$...
  - A *blank node valuation* β...
  - ...gives a domain element or literal value $\beta(b) \in \Delta^{\mathcal{I}} \cup \Lambda$...
  - ...for every blank node ID b
- Now define $\cdot^{\mathcal{I},\beta}$

Blank Node Valuations

- Given an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$...
  - A *blank node valuation* β...
  - ...gives a domain element or literal value $\beta(b) \in \Delta^{\mathcal{I}} \cup \Lambda$...
  - ...for every blank node ID b
- Now define $\cdot^{\mathcal{I},\beta}$
  - $i^{\mathcal{I},\beta} = i^{\mathcal{I}}$ for individual URIs i

Blank Node Valuations

- Given an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$...
  - A *blank node valuation* β...
  - ...gives a domain element or literal value $\beta(b) \in \Delta^{\mathcal{I}} \cup \Lambda$...
  - ...for every blank node ID b
- Now define $\cdot^{\mathcal{I},\beta}$
  - $i^{\mathcal{I},\beta} = i^{\mathcal{I}}$ for individual URIs i
  - $l^{\mathcal{I},\beta} = l$ for literals l

Blank Node Valuations

- Given an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$...
  - A *blank node valuation* β...
  - ...gives a domain element or literal value $\beta(b) \in \Delta^{\mathcal{I}} \cup \Lambda$...
  - ...for every blank node ID b
- Now define $\cdot^{\mathcal{I},\beta}$
  - $i^{\mathcal{I},\beta} = i^{\mathcal{I}}$ for individual URIs i
  - $l^{\mathcal{I},\beta} = l$ for literals l
  - $b^{\mathcal{I},\beta} = \beta(b)$ for blank node IDs b

Blank Node Valuations

- Given an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}\ldots$
  - A *blank node valuation* $\beta\ldots$
  - $\ldots$gives a domain element or literal value $\beta(b) \in \Delta^{\mathcal{I}} \cup \Lambda\ldots$
  - $\ldots$for every blank node ID b
- Now define $\cdot^{\mathcal{I},\beta}$
  - $i^{\mathcal{I},\beta} = i^{\mathcal{I}}$ for individual URIs i
  - $l^{\mathcal{I},\beta} = l$ for literals l
  - $b^{\mathcal{I},\beta} = \beta(b)$ for blank node IDs b
- Interpretation:

Blank Node Valuations

- Given an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}\ldots$
  - A *blank node valuation* $\beta\ldots$
  - …gives a domain element or literal value $\beta(b) \in \Delta^{\mathcal{I}} \cup \Lambda\ldots$
  - …for every blank node ID b
- Now define $\cdot^{\mathcal{I},\beta}$
  - $i^{\mathcal{I},\beta} = i^{\mathcal{I}}$ for individual URIs i
  - $l^{\mathcal{I},\beta} = l$ for literals l
  - $b^{\mathcal{I},\beta} = \beta(b)$ for blank node IDs b
- Interpretation:
  - $\mathcal{I}, \beta \models r(x, y)$ iff $\langle x^{\mathcal{I},\beta}, y^{\mathcal{I},\beta} \rangle \in r^{\mathcal{I}}\ldots$

Blank Node Valuations

- Given an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$...
  - A *blank node valuation* β...
  - ...gives a domain element or literal value $\beta(b) \in \Delta^{\mathcal{I}} \cup \Lambda$...
  - ...for every blank node ID b
- Now define $\cdot^{\mathcal{I},\beta}$
  - $i^{\mathcal{I},\beta} = i^{\mathcal{I}}$ for individual URIs i
  - $l^{\mathcal{I},\beta} = l$ for literals l
  - $b^{\mathcal{I},\beta} = \beta(b)$ for blank node IDs b
- Interpretation:
  - $\mathcal{I}, \beta \models r(x, y)$ iff $\langle x^{\mathcal{I},\beta}, y^{\mathcal{I},\beta} \rangle \in r^{\mathcal{I}}$...
  - ...for any legal combination of URIs/literals/blank nodes x, y

Blank Node Valuations

- Given an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$...
  - A *blank node valuation* β...
  - ...gives a domain element or literal value $\beta(b) \in \Delta^{\mathcal{I}} \cup \Lambda$...
  - ...for every blank node ID b
- Now define $\cdot^{\mathcal{I},\beta}$
  - $i^{\mathcal{I},\beta} = i^{\mathcal{I}}$ for individual URIs i
  - $l^{\mathcal{I},\beta} = l$ for literals l
  - $b^{\mathcal{I},\beta} = \beta(b)$ for blank node IDs b
- Interpretation:
  - $\mathcal{I}, \beta \models r(x, y)$ iff $\langle x^{\mathcal{I},\beta}, y^{\mathcal{I},\beta} \rangle \in r^{\mathcal{I}}$...
  - ...for any legal combination of URIs/literals/blank nodes x, y
  - ...and object/datatype property r

Blank Node Valuations

- Given an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}\ldots$
  - A *blank node valuation* $\beta\ldots$
  - …gives a domain element or literal value $\beta(b) \in \Delta^{\mathcal{I}} \cup \Lambda\ldots$
  - …for every blank node ID b
- Now define $\cdot^{\mathcal{I},\beta}$
  - $i^{\mathcal{I},\beta} = i^{\mathcal{I}}$ for individual URIs i
  - $l^{\mathcal{I},\beta} = l$ for literals l
  - $b^{\mathcal{I},\beta} = \beta(b)$ for blank node IDs b
- Interpretation:
  - $\mathcal{I}, \beta \models r(x, y)$ iff $\langle x^{\mathcal{I},\beta}, y^{\mathcal{I},\beta}\rangle \in r^{\mathcal{I}}\ldots$
  - …for any legal combination of URIs/literals/blank nodes x, y
  - …and object/datatype property r
  - $\mathcal{I}, \beta \models C(x)$ iff $x^{\mathcal{I},\beta} \in C^{\mathcal{I},\beta}$

Blank Node Valuations

- Given an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}\ldots$
  - A *blank node valuation* $\beta\ldots$
  - ...gives a domain element or literal value $\beta(b) \in \Delta^{\mathcal{I}} \cup \Lambda\ldots$
  - ...for every blank node ID b
- Now define $\cdot^{\mathcal{I},\beta}$
  - $i^{\mathcal{I},\beta} = i^{\mathcal{I}}$ for individual URIs i
  - $l^{\mathcal{I},\beta} = l$ for literals l
  - $b^{\mathcal{I},\beta} = \beta(b)$ for blank node IDs b
- Interpretation:
  - $\mathcal{I}, \beta \models r(x, y)$ iff $\langle x^{\mathcal{I},\beta}, y^{\mathcal{I},\beta} \rangle \in r^{\mathcal{I}}\ldots$
  - ...for any legal combination of URIs/literals/blank nodes x, y
  - ...and object/datatype property r
  - $\mathcal{I}, \beta \models C(x)$ iff $x^{\mathcal{I},\beta} \in C^{\mathcal{I},\beta}$
  - ...for any URI/blank node x

Sets of Triples with Blank Nodes

- Given a set $\mathcal{A}$ of triples with blank nodes...

Sets of Triples with Blank Nodes

- Given a set $\mathcal{A}$ of triples with blank nodes...
- $\mathcal{I}, \beta \models \mathcal{A}$ iff $\mathcal{I}, \beta \models A$ for all $A \in \mathcal{A}$

Sets of Triples with Blank Nodes

- Given a set $\mathcal{A}$ of triples with blank nodes...
- $\mathcal{I}, \beta \models \mathcal{A}$ iff $\mathcal{I}, \beta \models A$ for all $A \in \mathcal{A}$

- $\mathcal{A}$ is valid in $\mathcal{I}$

Sets of Triples with Blank Nodes

- Given a set $\mathcal{A}$ of triples with blank nodes...
- $\mathcal{I}, \beta \models \mathcal{A}$ iff $\mathcal{I}, \beta \models A$ for all $A \in \mathcal{A}$
- $\mathcal{A}$ is valid in $\mathcal{I}$

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

Sets of Triples with Blank Nodes

- Given a set $\mathcal{A}$ of triples with blank nodes...
- $\mathcal{I}, \beta \models \mathcal{A}$ iff $\mathcal{I}, \beta \models A$ for all $A \in \mathcal{A}$
- $\mathcal{A}$ is valid in $\mathcal{I}$

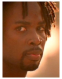
$$\mathcal{I} \models \mathcal{A}$$

  if there is a β such that $\mathcal{I}, \beta \models \mathcal{A}$

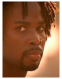
Sets of Triples with Blank Nodes

- Given a set $\mathcal{A}$ of triples with blank nodes...
- $\mathcal{I}, \beta \models \mathcal{A}$ iff $\mathcal{I}, \beta \models A$ for all $A \in \mathcal{A}$
- $\mathcal{A}$ is valid in $\mathcal{I}$
$$\mathcal{I} \models \mathcal{A}$$
if there is a β such that $\mathcal{I}, \beta \models \mathcal{A}$
- I.e. if there exists some valuation for the blank nodes that makes all triples true.

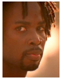
Example: Blank Node Semantics

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

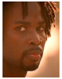
Example: Blank Node Semantics

- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $loves^{\mathcal{I}_1} = \{ \langle$ ,  $\rangle, \langle$ ,  $\rangle \}$ $\quad knows^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$

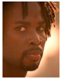
Example: Blank Node Semantics

- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $loves^{\mathcal{I}_1} = \{ \langle$ ,  $\rangle, \langle$ ,  $\rangle \}$ $\quad knows^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$
- $age^{\mathcal{I}_1} = \{ \langle$ , $\texttt{"16"} \rangle, \langle$ , $\texttt{"almost 14"} \rangle, \langle$ , $\texttt{"13"} \rangle, \}$

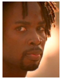
Example: Blank Node Semantics

- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $\mathit{loves}^{\mathcal{I}_1} = \{ \langle$ ,  $\rangle, \langle$ ,  $\rangle \}$ $\mathit{knows}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$
- $\mathit{age}^{\mathcal{I}_1} = \{ \langle$ , `"16"` $\rangle, \langle$ , `"almost 14"` $\rangle, \langle$ , `"13"` $\rangle, \}$
- Let b_1, b_2, b_3 be blank nodes

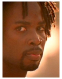
Example: Blank Node Semantics

- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $\mathit{loves}^{\mathcal{I}_1} = \{\langle$ ,  $\rangle, \langle$ ,  $\rangle\}$ $\quad \mathit{knows}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$
- $\mathit{age}^{\mathcal{I}_1} = \{\langle$ , $\texttt{"16"}\rangle, \langle$ , $\texttt{"almost 14"}\rangle, \langle$ , $\texttt{"13"}\rangle, \}$
- Let b_1, b_2, b_3 be blank nodes
- $\mathcal{A} = \{\mathit{age}(b_1, \texttt{"16"}), \mathit{knows}(b_1, b_2), \mathit{loves}(b_2, b_3), \mathit{age}(b_3, \texttt{"13"})\}$

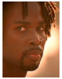
Example: Blank Node Semantics

- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $\mathit{loves}^{\mathcal{I}_1} = \{\langle$ ,  $\rangle, \langle$ ,  $\rangle\}$ $\quad \mathit{knows}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$
- $\mathit{age}^{\mathcal{I}_1} = \{\langle$ , $\texttt{"16"} \rangle, \langle$ , $\texttt{"almost 14"} \rangle, \langle$ , $\texttt{"13"} \rangle, \}$
- Let b_1, b_2, b_3 be blank nodes
- $\mathcal{A} = \{\mathit{age}(b_1, \texttt{"16"}), \mathit{knows}(b_1, b_2), \mathit{loves}(b_2, b_3), \mathit{age}(b_3, \texttt{"13"})\}$
- Valid in $\mathcal{I}_1$?

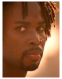
Example: Blank Node Semantics

- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $\mathit{loves}^{\mathcal{I}_1} = \{\langle$ ,  $\rangle, \langle$ ,  $\rangle\}$ $\mathit{knows}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$
- $\mathit{age}^{\mathcal{I}_1} = \{\langle$ , `"16"` $\rangle, \langle$ , `"almost 14"` $\rangle, \langle$ , `"13"` $\rangle, \}$
- Let b_1, b_2, b_3 be blank nodes
- $\mathcal{A} = \{\mathit{age}(b_1, \texttt{"16"}), \mathit{knows}(b_1, b_2), \mathit{loves}(b_2, b_3), \mathit{age}(b_3, \texttt{"13"})\}$
- Valid in $\mathcal{I}_1$?
- Pick $\beta(b_1) = \beta(b_2) =$ , $\beta(b_3) =$ .

Example: Blank Node Semantics

- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $\mathit{loves}^{\mathcal{I}_1} = \{\langle$ ,  $\rangle, \langle$ ,  $\rangle\}$ $\quad \mathit{knows}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$
- $\mathit{age}^{\mathcal{I}_1} = \{\langle$ , `"16"` $\rangle, \langle$ , `"almost 14"` $\rangle, \langle$ , `"13"` $\rangle, \}$
- Let b_1, b_2, b_3 be blank nodes
- $\mathcal{A} = \{\mathit{age}(b_1, \texttt{"16"}), \mathit{knows}(b_1, b_2), \mathit{loves}(b_2, b_3), \mathit{age}(b_3, \texttt{"13"})\}$
- Valid in $\mathcal{I}_1$?
- Pick $\beta(b_1) = \beta(b_2) =$ , $\beta(b_3) =$ .
- Then $\mathcal{I}_1, \beta \models \mathcal{A}$

Example: Blank Node Semantics

- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $\mathit{loves}^{\mathcal{I}_1} = \{\langle$ ,  $\rangle, \langle$ ,  $\rangle\}$ $\quad \mathit{knows}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$
- $\mathit{age}^{\mathcal{I}_1} = \{\langle$ , $\texttt{"16"}\rangle, \langle$ , $\texttt{"almost 14"}\rangle, \langle$ , $\texttt{"13"}\rangle, \}$
- Let b_1, b_2, b_3 be blank nodes
- $\mathcal{A} = \{\mathit{age}(b_1, \texttt{"16"}), \mathit{knows}(b_1, b_2), \mathit{loves}(b_2, b_3), \mathit{age}(b_3, \texttt{"13"})\}$
- Valid in $\mathcal{I}_1$?
- Pick $\beta(b_1) = \beta(b_2) =$ , $\beta(b_3) =$ .
- Then $\mathcal{I}_1, \beta \models \mathcal{A}$
- So, yes, $\mathcal{I}_1 \models \mathcal{A}$.

Entailment with Blank Nodes

- Entailment is defined just like without blank nodes:

Entailment with Blank Nodes

- Entailment is defined just like without blank nodes:
  - Given sets of triples $\mathcal{A}$ and $\mathcal{B}$,

Entailment with Blank Nodes

- Entailment is defined just like without blank nodes:
  - Given sets of triples $\mathcal{A}$ and $\mathcal{B}$,
  - $\mathcal{A}$ entails $\mathcal{B}$, written $\mathcal{A} \models \mathcal{B}$

Entailment with Blank Nodes

- Entailment is defined just like without blank nodes:
  - Given sets of triples $\mathcal{A}$ and $\mathcal{B}$,
  - $\mathcal{A}$ entails $\mathcal{B}$, written $\mathcal{A} \models \mathcal{B}$
  - iff for any interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$, also $\mathcal{I} \models \mathcal{B}$.

Entailment with Blank Nodes

- Entailment is defined just like without blank nodes:
  - Given sets of triples $\mathcal{A}$ and $\mathcal{B}$,
  - $\mathcal{A}$ entails $\mathcal{B}$, written $\mathcal{A} \models \mathcal{B}$
  - iff for any interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$, also $\mathcal{I} \models \mathcal{B}$.
- This expands to: for any interpretation $\mathcal{I}$

Entailment with Blank Nodes

- Entailment is defined just like without blank nodes:
  - Given sets of triples $\mathcal{A}$ and $\mathcal{B}$,
  - $\mathcal{A}$ entails $\mathcal{B}$, written $\mathcal{A} \models \mathcal{B}$
  - iff for any interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$, also $\mathcal{I} \models \mathcal{B}$.
- This expands to: for any interpretation $\mathcal{I}$
  - such that there exists a β with $\mathcal{I}, \beta \models \mathcal{A}$

Entailment with Blank Nodes

- Entailment is defined just like without blank nodes:
  - Given sets of triples $\mathcal{A}$ and $\mathcal{B}$,
  - $\mathcal{A}$ entails $\mathcal{B}$, written $\mathcal{A} \models \mathcal{B}$
  - iff for any interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$, also $\mathcal{I} \models \mathcal{B}$.
- This expands to: for any interpretation $\mathcal{I}$
  - such that there exists a β with $\mathcal{I}, \beta \models \mathcal{A}$
  - there also exists a β such that $\mathcal{I}, \beta \models \mathcal{B}$

Entailment with Blank Nodes

- Entailment is defined just like without blank nodes:
  - Given sets of triples $\mathcal{A}$ and $\mathcal{B}$,
  - $\mathcal{A}$ entails $\mathcal{B}$, written $\mathcal{A} \models \mathcal{B}$
  - iff for any interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$, also $\mathcal{I} \models \mathcal{B}$.
- This expands to: for any interpretation $\mathcal{I}$
  - such that there exists a β_1 with $\mathcal{I}, \beta_1 \models \mathcal{A}$
  - there also exists a β_2 such that $\mathcal{I}, \beta_2 \models \mathcal{B}$
- Two different blank node valuations!

Entailment with Blank Nodes

- Entailment is defined just like without blank nodes:
  - Given sets of triples $\mathcal{A}$ and $\mathcal{B}$,
  - $\mathcal{A}$ entails $\mathcal{B}$, written $\mathcal{A} \models \mathcal{B}$
  - iff for any interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$, also $\mathcal{I} \models \mathcal{B}$.
- This expands to: for any interpretation $\mathcal{I}$
  - such that there exists a β_1 with $\mathcal{I}, \beta_1 \models \mathcal{A}$
  - there also exists a β_2 such that $\mathcal{I}, \beta_2 \models \mathcal{B}$
- Two different blank node valuations!
- Can evaluate the same blank node name differently in $\mathcal{A}$ and $\mathcal{B}$.

Entailment with Blank Nodes

- Entailment is defined just like without blank nodes:
  - Given sets of triples $\mathcal{A}$ and $\mathcal{B}$,
  - $\mathcal{A}$ entails $\mathcal{B}$, written $\mathcal{A} \models \mathcal{B}$
  - iff for any interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$, also $\mathcal{I} \models \mathcal{B}$.
- This expands to: for any interpretation $\mathcal{I}$
  - such that there exists a β_1 with $\mathcal{I}, \beta_1 \models \mathcal{A}$
  - there also exists a β_2 such that $\mathcal{I}, \beta_2 \models \mathcal{B}$
- Two different blank node valuations!
- Can evaluate the same blank node name differently in $\mathcal{A}$ and $\mathcal{B}$.
- Example:

Entailment with Blank Nodes

- Entailment is defined just like without blank nodes:
  - Given sets of triples $\mathcal{A}$ and $\mathcal{B}$,
  - $\mathcal{A}$ entails $\mathcal{B}$, written $\mathcal{A} \models \mathcal{B}$
  - iff for any interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$, also $\mathcal{I} \models \mathcal{B}$.
- This expands to: for any interpretation $\mathcal{I}$
  - such that there exists a β_1 with $\mathcal{I}, \beta_1 \models \mathcal{A}$
  - there also exists a β_2 such that $\mathcal{I}, \beta_2 \models \mathcal{B}$
- Two different blank node valuations!
- Can evaluate the same blank node name differently in $\mathcal{A}$ and $\mathcal{B}$.
- Example:

$$\{loves(b_1, juliet), knows(juliet, romeo), age(juliet, \texttt{"13"})\}$$

Entailment with Blank Nodes

- Entailment is defined just like without blank nodes:
  - Given sets of triples $\mathcal{A}$ and $\mathcal{B}$,
  - $\mathcal{A}$ entails $\mathcal{B}$, written $\mathcal{A} \models \mathcal{B}$
  - iff for any interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$, also $\mathcal{I} \models \mathcal{B}$.
- This expands to: for any interpretation $\mathcal{I}$
  - such that there exists a β_1 with $\mathcal{I}, \beta_1 \models \mathcal{A}$
  - there also exists a β_2 such that $\mathcal{I}, \beta_2 \models \mathcal{B}$
- Two different blank node valuations!
- Can evaluate the same blank node name differently in $\mathcal{A}$ and $\mathcal{B}$.
- Example:

$$\{\mathit{loves}(b_1, \mathit{juliet}), \mathit{knows}(\mathit{juliet}, \mathit{romeo}), \mathit{age}(\mathit{juliet}, \texttt{"13"})\}$$

$$\models \{\mathit{loves}(b_2, b_1), \mathit{knows}(b_1, \mathit{romeo})\}$$

Entailment with Blank Nodes

- Entailment is defined just like without blank nodes:
  - Given sets of triples $\mathcal{A}$ and $\mathcal{B}$,
  - $\mathcal{A}$ entails $\mathcal{B}$, written $\mathcal{A} \models \mathcal{B}$
  - iff for any interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$, also $\mathcal{I} \models \mathcal{B}$.
- This expands to: for any interpretation $\mathcal{I}$
  - such that there exists a β_1 with $\mathcal{I}, \beta_1 \models \mathcal{A}$
  - there also exists a β_2 such that $\mathcal{I}, \beta_2 \models \mathcal{B}$
- Two different blank node valuations!
- Can evaluate the same blank node name differently in $\mathcal{A}$ and $\mathcal{B}$.
- Example:

$$\{loves(b_1, juliet), knows(juliet, romeo), age(juliet, \texttt{"13"})\}$$
$$\models \{loves(b_2, b_1), knows(b_1, romeo)\}$$

- Simple entailment: entailment with blank nodes, but no RDFS semantics

Simple Entailment: Rules and Example

$$\frac{r(u,x)}{r(u,b_1)}\ \text{se1} \qquad \frac{r(u,x)}{r(b_1,x)}\ \text{se2}$$

Where b_1 is a blank node identifier, that either

- has not been used before in the graph, or
- has been used, but for the same URI/Literal/Blank node x resp. u.

Simple Entailment: Rules and Example

$$\frac{r(u,x)}{r(u,b_1)}\ \text{se1} \qquad \frac{r(u,x)}{r(b_1,x)}\ \text{se2}$$

Where b_1 is a blank node identifier, that either

- has not been used before in the graph, or
- has been used, but for the same URI/Literal/Blank node x resp. u.

$$\{loves(b_1, juliet), knows(juliet, romeo), age(juliet, \texttt{"13"})\}$$

Simple Entailment: Rules and Example

$$\frac{r(u,x)}{r(u,b_1)}\ \text{se1} \qquad \frac{r(u,x)}{r(b_1,x)}\ \text{se2}$$

Where b_1 is a blank node identifier, that either

- has not been used before in the graph, or
- has been used, but for the same URI/Literal/Blank node x resp. u.

$$\{\mathit{loves}(b_1, \mathit{juliet}), \mathit{knows}(\mathit{juliet}, \mathit{romeo}), \mathit{age}(\mathit{juliet}, \texttt{"13"})\}$$

$\models \{\mathit{loves}(b_2, b_3), \mathit{knows}(b_3, \mathit{romeo})\}$ renamed blank nodes in $\mathcal{B}$!

Simple Entailment: Rules and Example

$$\frac{r(u,x)}{r(u,b_1)}\ \text{se1} \qquad \frac{r(u,x)}{r(b_1,x)}\ \text{se2}$$

Where b_1 is a blank node identifier, that either

- has not been used before in the graph, or
- has been used, but for the same URI/Literal/Blank node x resp. u.

$$\{loves(b_1, juliet), knows(juliet, romeo), age(juliet, \texttt{"13"})\}$$

$loves(b_2, juliet)$ se2, $(b_2 \to b_1)$

$\models \{loves(b_2, b_3), knows(b_3, romeo)\}$ renamed blank nodes in $\mathcal{B}$!

Simple Entailment: Rules and Example

$$\frac{r(u,x)}{r(u,b_1)}\ \text{se1} \qquad \frac{r(u,x)}{r(b_1,x)}\ \text{se2}$$

Where b_1 is a blank node identifier, that either

- has not been used before in the graph, or
- has been used, but for the same URI/Literal/Blank node x resp. u.

$$\{loves(b_1, juliet), knows(juliet, romeo), age(juliet, \texttt{"13"})\}$$

$loves(b_2, juliet)$ se2, $(b_2 \to b_1)$

$loves(b_2, b_3)$ se1, $(b_3 \to juliet)$

$\models \{loves(b_2, b_3), knows(b_3, romeo)\}$ renamed blank nodes in $\mathcal{B}$!

Simple Entailment: Rules and Example

$$\frac{r(u, x)}{r(u, b_1)}\ \text{se1} \qquad \frac{r(u, x)}{r(b_1, x)}\ \text{se2}$$

Where b_1 is a blank node identifier, that either

- has not been used before in the graph, or
- has been used, but for the same URI/Literal/Blank node x resp. u.

$\{loves(b_1, juliet), knows(juliet, romeo), age(juliet, \texttt{"13"})\}$

$loves(b_2, juliet)$ se2, $(b_2 \to b_1)$

$loves(b_2, b_3)$ se1, $(b_3 \to juliet)$

$knows(b_3, romeo)$ se2, (reusing $b_3 \to juliet$)

$\models \{loves(b_2, b_3), knows(b_3, romeo)\}$ renamed blank nodes in $\mathcal{B}$!

Outline

Monotonicity

- Assume $\mathcal{A} \models \mathcal{B}$

Monotonicity

- Assume $\mathcal{A} \models \mathcal{B}$
- Now add information to $\mathcal{A}$, i.e. $\mathcal{A}' \supseteq \mathcal{A}$

Monotonicity

- Assume $\mathcal{A} \models \mathcal{B}$
- Now add information to $\mathcal{A}$, i.e. $\mathcal{A}' \supseteq \mathcal{A}$
- Then $\mathcal{B}$ is still entailed: $\mathcal{A}' \models \mathcal{B}$

Monotonicity

- Assume $\mathcal{A} \models \mathcal{B}$
- Now add information to $\mathcal{A}$, i.e. $\mathcal{A}' \supseteq \mathcal{A}$
- Then $\mathcal{B}$ is still entailed: $\mathcal{A}' \models \mathcal{B}$

- We say that RDF/RDFS entailment is *monotonic*

Monotonicity

- Assume $\mathcal{A} \models \mathcal{B}$
- Now add information to $\mathcal{A}$, i.e. $\mathcal{A}' \supseteq \mathcal{A}$
- Then $\mathcal{B}$ is still entailed: $\mathcal{A}' \models \mathcal{B}$
- We say that RDF/RDFS entailment is *monotonic*
- What would non-monotonic reasoning be like?

Monotonicity

- Assume $\mathcal{A} \models \mathcal{B}$
- Now add information to $\mathcal{A}$, i.e. $\mathcal{A}' \supseteq \mathcal{A}$
- Then $\mathcal{B}$ is still entailed: $\mathcal{A}' \models \mathcal{B}$
- We say that RDF/RDFS entailment is *monotonic*
- What would non-monotonic reasoning be like?
  - $\{Bird \sqsubseteq CanFly, Bird(tweety)\} \models CanFly(tweety)$

Monotonicity

- Assume $\mathcal{A} \models \mathcal{B}$
- Now add information to $\mathcal{A}$, i.e. $\mathcal{A}' \supseteq \mathcal{A}$
- Then $\mathcal{B}$ is still entailed: $\mathcal{A}' \models \mathcal{B}$
- We say that RDF/RDFS entailment is *monotonic*
- What would non-monotonic reasoning be like?
  - $\{\mathit{Bird} \sqsubseteq \mathit{CanFly}, \mathit{Bird}(\mathit{tweety})\} \models \mathit{CanFly}(\mathit{tweety})$
  - $\{\ldots, \mathit{Penguin} \sqsubseteq \mathit{Bird}, \mathit{Penguin}(\mathit{tweety}), \mathit{Penguin} \sqsubseteq \neg\mathit{CanFly}\} \not\models \mathit{CanFly}(\mathit{tweety})$

Monotonicity

- Assume $\mathcal{A} \models \mathcal{B}$
- Now add information to $\mathcal{A}$, i.e. $\mathcal{A}' \supseteq \mathcal{A}$
- Then $\mathcal{B}$ is still entailed: $\mathcal{A}' \models \mathcal{B}$
- We say that RDF/RDFS entailment is *monotonic*
- What would non-monotonic reasoning be like?
  - $\{Bird \sqsubseteq CanFly, Bird(tweety)\} \models CanFly(tweety)$
  - $\{\ldots, Penguin \sqsubseteq Bird, Penguin(tweety), Penguin \sqsubseteq \neg CanFly\} \not\models CanFly(tweety)$
  - Interesting for human-style reasoning

Monotonicity

- Assume $\mathcal{A} \models \mathcal{B}$
- Now add information to $\mathcal{A}$, i.e. $\mathcal{A}' \supseteq \mathcal{A}$
- Then $\mathcal{B}$ is still entailed: $\mathcal{A}' \models \mathcal{B}$
- We say that RDF/RDFS entailment is *monotonic*
- What would non-monotonic reasoning be like?
  - $\{Bird \sqsubseteq CanFly, Bird(tweety)\} \models CanFly(tweety)$
  - $\{\ldots, Penguin \sqsubseteq Bird, Penguin(tweety), Penguin \sqsubseteq \neg CanFly\} \not\models CanFly(tweety)$
  - Interesting for human-style reasoning
  - Hard to combine with semantic web technologies

Expressive limitations of RDFS

Note that,

- RDFS cannot express inconsistencies, so any RDFS graph is consistent.

Expressive limitations of RDFS

Note that,

- RDFS cannot express inconsistencies, so any RDFS graph is consistent.
- RDFS has no notion of negation at all

Expressive limitations of RDFS

Note that,

- RDFS cannot express inconsistencies, so any RDFS graph is consistent.
- RDFS has no notion of negation at all
    - For instance, the two triples

Expressive limitations of RDFS

Note that,

- RDFS cannot express inconsistencies, so any RDFS graph is consistent.
- RDFS has no notion of negation at all
  - For instance, the two triples

```
ex:Joe rdf:type ex:Smoker .
```

Expressive limitations of RDFS

Note that,

- RDFS cannot express inconsistencies, so any RDFS graph is consistent.
- RDFS has no notion of negation at all
  - For instance, the two triples

```
ex:Joe rdf:type ex:Smoker .
ex:Joe rdf:type ex:NonSmoker .
```

Expressive limitations of RDFS

Note that,

- RDFS cannot express inconsistencies, so any RDFS graph is consistent.
- RDFS has no notion of negation at all
  - For instance, the two triples

    ```
    ex:Joe rdf:type ex:Smoker .
    ex:Joe rdf:type ex:NonSmoker .
    ```

    are not inconsistent.

Expressive limitations of RDFS

Note that,

- RDFS cannot express inconsistencies, so any RDFS graph is consistent.
- RDFS has no notion of negation at all
  - For instance, the two triples

    ```
    ex:Joe rdf:type ex:Smoker .
    ex:Joe rdf:type ex:NonSmoker .
    ```

    are not inconsistent.
    - (It is not possible to in RDFS to say that `ex:Smoker` and `ex:nonSmoker` are disjoint).

Therefore,

Expressive limitations of RDFS

Note that,

- RDFS cannot express inconsistencies, so any RDFS graph is consistent.
- RDFS has no notion of negation at all
  - For instance, the two triples

    ```
    ex:Joe rdf:type ex:Smoker .
    ex:Joe rdf:type ex:NonSmoker .
    ```

    are not inconsistent.
    - (It is not possible to in RDFS to say that `ex:Smoker` and `ex:nonSmoker` are disjoint).

Therefore,

- RDFS supports no reasoning services that require consistency-checking.

Expressive limitations of RDFS

Note that,

- RDFS cannot express inconsistencies, so any RDFS graph is consistent.
- RDFS has no notion of negation at all
  - For instance, the two triples

    ```
    ex:Joe rdf:type ex:Smoker .
    ex:Joe rdf:type ex:NonSmoker .
    ```

    are not inconsistent.
    - (It is not possible to in RDFS to say that `ex:Smoker` and `ex:nonSmoker` are disjoint).

Therefore,

- RDFS supports no reasoning services that require consistency-checking.
- If negation or consistency-checks are needed, one must turn to OWL.

Expressive limitations of RDFS

Note that,

- RDFS cannot express inconsistencies, so any RDFS graph is consistent.
- RDFS has no notion of negation at all
  - For instance, the two triples

    ```
    ex:Joe rdf:type ex:Smoker .
    ex:Joe rdf:type ex:NonSmoker .
    ```

    are not inconsistent.
    - (It is not possible to in RDFS to say that `ex:Smoker` and `ex:nonSmoker` are disjoint).

Therefore,

- RDFS supports no reasoning services that require consistency-checking.
- If negation or consistency-checks are needed, one must turn to OWL.
- More about that next week.

Entailment and SPARQL

- Given a knowledge base *KB* and a query `SELECT * WHERE {?x :p ?y.  ?y :q ?z.}`

Entailment and SPARQL

- Given a knowledge base KB and a query `SELECT * WHERE {?x :p ?y.  ?y :q ?z.}`
- The query means: find x, y, z with $p(x, y)$ and $q(y, z)$

Entailment and SPARQL

- Given a knowledge base KB and a query `SELECT * WHERE {?x :p ?y.  ?y :q ?z.}`
- The query means: find x, y, z with $p(x, y)$ and $q(y, z)$
- Semantics: find x, y, z with

$$KB \models \{p(x, y),\ q(y, z)\}$$

Entailment and SPARQL

- Given a knowledge base KB and a query `SELECT * WHERE {?x :p ?y.  ?y :q ?z.}`
- The query means: find x, y, z with $p(x, y)$ and $q(y, z)$
- Semantics: find x, y, z with

$$KB \models \{p(x,y),\ q(y,z)\}$$

- E.g. an answer

$$x \leftarrow \texttt{ex:a} \quad y \leftarrow \texttt{ifi:in3060} \quad z \leftarrow \texttt{"a"}$$

  means

$$KB \models \{p(a, \texttt{in3060}),\ q(\texttt{in3060}, \texttt{"a"})\}$$

Entailment and SPARQL

- Given a knowledge base KB and a query `SELECT * WHERE {?x :p ?y. ?y :q ?z.}`
- The query means: find x, y, z with $p(x, y)$ and $q(y, z)$
- Semantics: find x, y, z with
$$KB \models \{p(x, y),\ q(y, z)\}$$
- E.g. an answer
$$x \leftarrow \texttt{ex:a} \quad y \leftarrow \texttt{ifi:in3060} \quad z \leftarrow \texttt{"a"}$$
means
$$KB \models \{p(a, \texttt{in3060}),\ q(\texttt{in3060}, \texttt{"a"})\}$$
- Monotonicity:
$$KB \cup \{\cdots\} \models \{p(a, \texttt{in3060}),\ q(\texttt{in3060}, \texttt{"a"})\}$$

Entailment and SPARQL

- Given a knowledge base KB and a query `SELECT * WHERE {?x :p ?y.  ?y :q ?z.}`
- The query means: find x, y, z with $p(x, y)$ and $q(y, z)$
- Semantics: find x, y, z with

$$KB \models \{p(x, y),\ q(y, z)\}$$

- E.g. an answer

$$x \leftarrow \texttt{ex:a} \quad y \leftarrow \texttt{ifi:in3060} \quad z \leftarrow \texttt{"a"}$$

  means

$$KB \models \{p(a, \texttt{in3060}),\ q(\texttt{in3060}, \texttt{"a"})\}$$

- Monotonicity:

$$KB \cup \{\cdots\} \models \{p(a, \texttt{in3060}),\ q(\texttt{in3060}, \texttt{"a"})\}$$

- Answers remain valid with new information!

Database Lookup versus Entailment

- Knowledge base KB:

$$Person(harald) \qquad Person(haakon) \qquad isFatherOf(harald, haakon)$$

Database Lookup versus Entailment

- Knowledge base KB:

$$Person(harald) \qquad Person(haakon) \qquad isFatherOf(harald, haakon)$$

- Question: is there a person without a father?

Database Lookup versus Entailment

- Knowledge base KB:

$$Person(harald) \qquad Person(haakon) \qquad isFatherOf(harald, haakon)$$

- Question: is there a person without a father?
- Ask a database:

Database Lookup versus Entailment

- Knowledge base KB:

$$Person(harald) \qquad Person(haakon) \qquad isFatherOf(harald, haakon)$$

- Question: is there a person without a father?
- Ask a database:
  - Yes: *harald*

Database Lookup versus Entailment

- Knowledge base KB:

$$Person(harald) \qquad Person(haakon) \qquad isFatherOf(harald, haakon)$$

- Question: is there a person without a father?
- Ask a database:
  - Yes: *harald*
- ask a semantics based system

Database Lookup versus Entailment

- Knowledge base KB:

$$Person(harald) \qquad Person(haakon) \qquad isFatherOf(harald, haakon)$$

- Question: is there a person without a father?
- Ask a database:
  - Yes: *harald*
- ask a semantics based system
  - find x with $KB \models$ 'x *has no father*'

Database Lookup versus Entailment

- Knowledge base KB:

$$Person(harald) \qquad Person(haakon) \qquad isFatherOf(harald, haakon)$$

- Question: is there a person without a father?
- Ask a database:
  - Yes: *harald*
- ask a semantics based system
  - find x with $KB \models$ 'x *has no father*'
  - No answer: don't know

Database Lookup versus Entailment

- Knowledge base KB:

$$Person(harald) \qquad Person(haakon) \qquad isFatherOf(harald, haakon)$$

- Question: is there a person without a father?
- Ask a database:
  - Yes: *harald*
- ask a semantics based system
  - find x with $KB \models$ 'x *has no father*'
  - No answer: don't know
- Why?

Database Lookup versus Entailment

- Knowledge base KB:

$$Person(harald) \qquad Person(haakon) \qquad isFatherOf(harald, haakon)$$

- Question: is there a person without a father?
- Ask a database:
  - Yes: *harald*
- ask a semantics based system
  - find x with $KB \models$ 'x *has no father*'
  - No answer: don't know
- Why?
  - Monotonicity!

Database Lookup versus Entailment

- Knowledge base KB:

$$Person(harald) \qquad Person(haakon) \qquad isFatherOf(harald, haakon)$$

- Question: is there a person without a father?
- Ask a database:
  - Yes: *harald*
- ask a semantics based system
  - find x with $KB \models$ 'x *has no father*'
  - No answer: don't know
- Why?
  - Monotonicity!
  - $KB \cup \{isFatherOf(olav, harald)\} \models$ *harald does have a father*

Database Lookup versus Entailment

- Knowledge base KB:

$$Person(harald) \qquad Person(haakon) \qquad isFatherOf(harald, haakon)$$

- Question: is there a person without a father?
- Ask a database:
  - Yes: *harald*
- ask a semantics based system
  - find x with $KB \models$ '*x has no father*'
  - No answer: don't know
- Why?
  - Monotonicity!
  - $KB \cup \{isFatherOf(olav, harald)\} \models$ *harald does have a father*
  - In some models of KB, harald has a father, in others not.

Open World versus Closed World

- Closed World Assumption (CWA)

Open World versus Closed World

- Closed World Assumption (CWA)
  - If a thing is not listed in the knowledge base, it doesn't exist

Open World versus Closed World

- Closed World Assumption (CWA)
  - If a thing is not listed in the knowledge base, it doesn't exist
  - If a fact isn't stated (or derivable) it's false

Open World versus Closed World

- Closed World Assumption (CWA)
  - If a thing is not listed in the knowledge base, it doesn't exist
  - If a fact isn't stated (or derivable) it's false
  - Typical semantics for database systems

Open World versus Closed World

- Closed World Assumption (CWA)
  - If a thing is not listed in the knowledge base, it doesn't exist
  - If a fact isn't stated (or derivable) it's false
  - Typical semantics for database systems
- Open World Assumption (OWA)

Open World versus Closed World

- Closed World Assumption (CWA)
  - If a thing is not listed in the knowledge base, it doesn't exist
  - If a fact isn't stated (or derivable) it's false
  - Typical semantics for database systems
- Open World Assumption (OWA)
  - There might be things not mentioned in the knowledge base

Open World versus Closed World

- Closed World Assumption (CWA)
  - If a thing is not listed in the knowledge base, it doesn't exist
  - If a fact isn't stated (or derivable) it's false
  - Typical semantics for database systems
- Open World Assumption (OWA)
  - There might be things not mentioned in the knowledge base
  - There might be facts that are true, although they are not stated

Open World versus Closed World

- Closed World Assumption (CWA)
  - If a thing is not listed in the knowledge base, it doesn't exist
  - If a fact isn't stated (or derivable) it's false
  - Typical semantics for database systems
- Open World Assumption (OWA)
  - There might be things not mentioned in the knowledge base
  - There might be facts that are true, although they are not stated
  - Typical semantics for logic-based systems

Open World versus Closed World

- Closed World Assumption (CWA)
  - If a thing is not listed in the knowledge base, it doesn't exist
  - If a fact isn't stated (or derivable) it's false
  - Typical semantics for database systems
- Open World Assumption (OWA)
  - There might be things not mentioned in the knowledge base
  - There might be facts that are true, although they are not stated
  - Typical semantics for logic-based systems
- What is best for the Semantic Web?

Open World versus Closed World

- Closed World Assumption (CWA)
  - If a thing is not listed in the knowledge base, it doesn't exist
  - If a fact isn't stated (or derivable) it's false
  - Typical semantics for database systems
- Open World Assumption (OWA)
  - There might be things not mentioned in the knowledge base
  - There might be facts that are true, although they are not stated
  - Typical semantics for logic-based systems
- What is best for the Semantic Web?
  - Will never know all information sources

Open World versus Closed World

- Closed World Assumption (CWA)
  - If a thing is not listed in the knowledge base, it doesn't exist
  - If a fact isn't stated (or derivable) it's false
  - Typical semantics for database systems
- Open World Assumption (OWA)
  - There might be things not mentioned in the knowledge base
  - There might be facts that are true, although they are not stated
  - Typical semantics for logic-based systems
- What is best for the Semantic Web?
  - Will never know all information sources
  - Can "discover" new information by following links

Open World versus Closed World

- Closed World Assumption (CWA)
  - If a thing is not listed in the knowledge base, it doesn't exist
  - If a fact isn't stated (or derivable) it's false
  - Typical semantics for database systems
- Open World Assumption (OWA)
  - There might be things not mentioned in the knowledge base
  - There might be facts that are true, although they are not stated
  - Typical semantics for logic-based systems
- What is best for the Semantic Web?
  - Will never know all information sources
  - Can "discover" new information by following links
  - New information can be produced at any time

Open World versus Closed World

- Closed World Assumption (CWA)
  - If a thing is not listed in the knowledge base, it doesn't exist
  - If a fact isn't stated (or derivable) it's false
  - Typical semantics for database systems
- Open World Assumption (OWA)
  - There might be things not mentioned in the knowledge base
  - There might be facts that are true, although they are not stated
  - Typical semantics for logic-based systems
- What is best for the Semantic Web?
  - Will never know all information sources
  - Can "discover" new information by following links
  - New information can be produced at any time
  - Therefore: Open World Assumption

Consequences of the Open World Assumption

- Robust under missing information

Consequences of the Open World Assumption

- Robust under missing information
- Any answer given by

Consequences of the Open World Assumption

- Robust under missing information
- Any answer given by
  - Entailment

$$KB \models Person(juliet)$$

Consequences of the Open World Assumption

- Robust under missing information
- Any answer given by
  - Entailment

$$KB \models Person(juliet)$$

  - SPARQL query answering (entailment in disguise)

$$KB \models \{p(a, \texttt{in3060}),\ q(\texttt{in3060}, \text{“a”})\}$$

Consequences of the Open World Assumption

- Robust under missing information
- Any answer given by
  - Entailment

$$KB \models \mathit{Person}(\mathit{juliet})$$

  - SPARQL query answering (entailment in disguise)

$$KB \models \{p(a, \mathtt{in3060}),\ q(\mathtt{in3060}, \text{“a”})\}$$

remains valid when new information is added to KB

Consequences of the Open World Assumption

- Robust under missing information
- Any answer given by
  - Entailment

$$KB \models Person(juliet)$$

  - SPARQL query answering (entailment in disguise)

$$KB \models \{p(a, \texttt{in3060}),\ q(\texttt{in3060}, \text{“a”})\}$$

  remains valid when new information is added to KB
- Some things make no sense with this semantics

Consequences of the Open World Assumption

- Robust under missing information
- Any answer given by
  - Entailment

$$KB \models Person(juliet)$$

  - SPARQL query answering (entailment in disguise)

$$KB \models \{p(a, \texttt{in3060}),\ q(\texttt{in3060}, \text{“a”})\}$$

  remains valid when new information is added to KB
- Some things make no sense with this semantics
  - Queries with negation (“not”)

Consequences of the Open World Assumption

- Robust under missing information
- Any answer given by
  - Entailment

$$KB \models Person(juliet)$$

  - SPARQL query answering (entailment in disguise)

$$KB \models \{p(a, \texttt{in3060}),\ q(\texttt{in3060}, \text{“a”})\}$$

  remains valid when new information is added to KB
- Some things make no sense with this semantics
  - Queries with negation (“not”)
    - might be satisfied later on

Consequences of the Open World Assumption

- Robust under missing information
- Any answer given by
  - Entailment

$$KB \models Person(juliet)$$

  - SPARQL query answering (entailment in disguise)

$$KB \models \{p(a, \texttt{in3060}),\ q(\texttt{in3060}, \text{“a”})\}$$

  remains valid when new information is added to KB
- Some things make no sense with this semantics
  - Queries with negation (“not”)
    - might be satisfied later on
  - Queries with aggregation (counting, adding, . . .)

Consequences of the Open World Assumption

- Robust under missing information
- Any answer given by
  - Entailment

$$KB \models \mathit{Person}(\mathit{juliet})$$

  - SPARQL query answering (entailment in disguise)

$$KB \models \{p(a, \mathtt{in3060}),\ q(\mathtt{in3060}, \text{``a''})\}$$

  remains valid when new information is added to KB
- Some things make no sense with this semantics
  - Queries with negation (“not”)
    - might be satisfied later on
  - Queries with aggregation (counting, adding, . . .)
    - can change when more information comes

Outline

Two Kinds of Consequence?

- We now have two ways of describing logical consequence...

Two Kinds of Consequence?

- We now have two ways of describing logical consequence. . .

1. Using RDFS rules:

$$\frac{\texttt{:Lady rdfs:subClassOf :Person .} \qquad \texttt{:juliet a :Lady .}}{\texttt{:juliet a :Person .}}\ \text{rdfs9}$$

Two Kinds of Consequence?

- We now have two ways of describing logical consequence...

1. Using RDFS rules:

$$\frac{\texttt{:Lady rdfs:subClassOf :Person .} \qquad \texttt{:juliet a :Lady .}}{\texttt{:juliet a :Person .}}\ \text{rdfs9}$$

$$\frac{\mathit{Lady} \sqsubseteq \mathit{Person} \qquad \mathit{Lady}(\mathit{juliet})}{\mathit{Person}(\mathit{juliet})}\ \text{rdfs9}$$

Two Kinds of Consequence?

- We now have two ways of describing logical consequence. . .

1. Using RDFS rules:

$$\frac{\texttt{:Lady rdfs:subClassOf :Person .} \qquad \texttt{:juliet a :Lady .}}{\texttt{:juliet a :Person .}}\ \text{rdfs9}$$

$$\frac{\mathit{Lady} \sqsubseteq \mathit{Person} \qquad \mathit{Lady}(\mathit{juliet})}{\mathit{Person}(\mathit{juliet})}\ \text{rdfs9}$$

2. Using the model semantics

Two Kinds of Consequence?

- We now have two ways of describing logical consequence...

1. Using RDFS rules:

$$\frac{\texttt{:Lady rdfs:subClassOf :Person .} \qquad \texttt{:juliet a :Lady .}}{\texttt{:juliet a :Person .}}\ \text{rdfs9}$$

$$\frac{\mathit{Lady} \sqsubseteq \mathit{Person} \qquad \mathit{Lady}(\mathit{juliet})}{\mathit{Person}(\mathit{juliet})}\ \text{rdfs9}$$

2. Using the model semantics
   - If $\mathcal{I} \models \mathit{Lady} \sqsubseteq \mathit{Person}$ and $\mathcal{I} \models \mathit{Lady}(\mathit{juliet})$...

Two Kinds of Consequence?

- We now have two ways of describing logical consequence...

1. Using RDFS rules:

$$\frac{\texttt{:Lady rdfs:subClassOf :Person .} \qquad \texttt{:juliet a :Lady .}}{\texttt{:juliet a :Person .}}\text{rdfs9}$$

$$\frac{\mathit{Lady} \sqsubseteq \mathit{Person} \qquad \mathit{Lady}(\mathit{juliet})}{\mathit{Person}(\mathit{juliet})}\text{rdfs9}$$

2. Using the model semantics
   - If $\mathcal{I} \models \mathit{Lady} \sqsubseteq \mathit{Person}$ and $\mathcal{I} \models \mathit{Lady}(\mathit{juliet})$...
   - ...then $\mathit{Lady}^{\mathcal{I}} \subseteq \mathit{Person}^{\mathcal{I}}$ and $\mathit{juliet}^{\mathcal{I}} \in \mathit{Lady}^{\mathcal{I}}$...

Two Kinds of Consequence?

- We now have two ways of describing logical consequence...

1. Using RDFS rules:

$$\frac{\texttt{:Lady rdfs:subClassOf :Person .} \qquad \texttt{:juliet a :Lady .}}{\texttt{:juliet a :Person .}}\ \text{rdfs9}$$

$$\frac{\mathit{Lady} \sqsubseteq \mathit{Person} \qquad \mathit{Lady}(\mathit{juliet})}{\mathit{Person}(\mathit{juliet})}\ \text{rdfs9}$$

2. Using the model semantics
   - If $\mathcal{I} \models \mathit{Lady} \sqsubseteq \mathit{Person}$ and $\mathcal{I} \models \mathit{Lady}(\mathit{juliet})$...
   - ...then $\mathit{Lady}^{\mathcal{I}} \subseteq \mathit{Person}^{\mathcal{I}}$ and $\mathit{juliet}^{\mathcal{I}} \in \mathit{Lady}^{\mathcal{I}}$...
   - ...so by set theory, $\mathit{juliet}^{\mathcal{I}} \in \mathit{Person}^{\mathcal{I}}$...

Two Kinds of Consequence?

- We now have two ways of describing logical consequence...

1. Using RDFS rules:

$$\frac{\texttt{:Lady rdfs:subClassOf :Person .} \qquad \texttt{:juliet a :Lady .}}{\texttt{:juliet a :Person .}} \text{rdfs9}$$

$$\frac{\mathit{Lady} \sqsubseteq \mathit{Person} \qquad \mathit{Lady}(\mathit{juliet})}{\mathit{Person}(\mathit{juliet})} \text{rdfs9}$$

2. Using the model semantics
   - If $\mathcal{I} \models \mathit{Lady} \sqsubseteq \mathit{Person}$ and $\mathcal{I} \models \mathit{Lady}(\mathit{juliet})$...
   - ...then $\mathit{Lady}^{\mathcal{I}} \subseteq \mathit{Person}^{\mathcal{I}}$ and $\mathit{juliet}^{\mathcal{I}} \in \mathit{Lady}^{\mathcal{I}}$...
   - ...so by set theory, $\mathit{juliet}^{\mathcal{I}} \in \mathit{Person}^{\mathcal{I}}$...
   - ...and therefore $\mathcal{I} \models \mathit{Person}(\mathit{juliet})$.

Two Kinds of Consequence?

- We now have two ways of describing logical consequence. . .

1. Using RDFS rules:

$$\frac{\texttt{:Lady rdfs:subClassOf :Person .} \qquad \texttt{:juliet a :Lady .}}{\texttt{:juliet a :Person .}}\ \text{rdfs9}$$

$$\frac{\mathit{Lady} \sqsubseteq \mathit{Person} \qquad \mathit{Lady}(\mathit{juliet})}{\mathit{Person}(\mathit{juliet})}\ \text{rdfs9}$$

2. Using the model semantics
   - If $\mathcal{I} \models \mathit{Lady} \sqsubseteq \mathit{Person}$ and $\mathcal{I} \models \mathit{Lady}(\mathit{juliet})$. . .
   - . . . then $\mathit{Lady}^{\mathcal{I}} \subseteq \mathit{Person}^{\mathcal{I}}$ and $\mathit{juliet}^{\mathcal{I}} \in \mathit{Lady}^{\mathcal{I}}$. . .
   - . . . so by set theory, $\mathit{juliet}^{\mathcal{I}} \in \mathit{Person}^{\mathcal{I}}$. . .
   - . . . and therefore $\mathcal{I} \models \mathit{Person}(\mathit{juliet})$.

   - Together: $\{\mathit{Lady} \sqsubseteq \mathit{Person}, \mathit{Lady}(\mathit{juliet})\} \models \mathit{Person}(\mathit{juliet})$

Two Kinds of Consequence?

- We now have two ways of describing logical consequence...

1. Using RDFS rules:

$$\frac{\texttt{:Lady rdfs:subClassOf :Person .} \qquad \texttt{:juliet a :Lady .}}{\texttt{:juliet a :Person .}}\ \text{rdfs9}$$

$$\frac{\mathit{Lady} \sqsubseteq \mathit{Person} \qquad \mathit{Lady}(\mathit{juliet})}{\mathit{Person}(\mathit{juliet})}\ \text{rdfs9}$$

2. Using the model semantics
   - If $\mathcal{I} \models \mathit{Lady} \sqsubseteq \mathit{Person}$ and $\mathcal{I} \models \mathit{Lady}(\mathit{juliet})$...
   - ...then $\mathit{Lady}^{\mathcal{I}} \subseteq \mathit{Person}^{\mathcal{I}}$ and $\mathit{juliet}^{\mathcal{I}} \in \mathit{Lady}^{\mathcal{I}}$...
   - ...so by set theory, $\mathit{juliet}^{\mathcal{I}} \in \mathit{Person}^{\mathcal{I}}$...
   - ...and therefore $\mathcal{I} \models \mathit{Person}(\mathit{juliet})$.
   - Together: $\{\mathit{Lady} \sqsubseteq \mathit{Person}, \mathit{Lady}(\mathit{juliet})\} \models \mathit{Person}(\mathit{juliet})$

- What is the connection between these two?

Entailment and Derivability

- Actually, two different notions!

Entailment and Derivability

- Actually, two different notions!
- Entailment is defined using the model semantics.

Entailment and Derivability

- Actually, two different notions!
- Entailment is defined using the model semantics.
- The rules say what can be *derived*

Entailment and Derivability

- Actually, two different notions!
- Entailment is defined using the model semantics.
- The rules say what can be *derived*
  - derivability

Entailment and Derivability

- Actually, two different notions!
- Entailment is defined using the model semantics.
- The rules say what can be *derived*
  - derivability
  - provability

Entailment and Derivability

- Actually, two different notions!
- Entailment is defined using the model semantics.
- The rules say what can be *derived*
  - derivability
  - provability
- Entailment

Entailment and Derivability

- Actually, two different notions!
- Entailment is defined using the model semantics.
- The rules say what can be *derived*
    - derivability
    - provability
- Entailment
    - is closely related to the *meaning* of things

Entailment and Derivability

- Actually, two different notions!
- Entailment is defined using the model semantics.
- The rules say what can be *derived*
  - derivability
  - provability
- Entailment
  - is closely related to the *meaning* of things
  - higher confidence in model semantics than in a bunch of rules

Entailment and Derivability

- Actually, two different notions!
- Entailment is defined using the model semantics.
- The rules say what can be *derived*
    - derivability
    - provability
- Entailment
    - is closely related to the *meaning* of things
    - higher confidence in model semantics than in a bunch of rules
    - *The* semantics given by the standard, rules are just "informative"

Entailment and Derivability

- Actually, two different notions!
- Entailment is defined using the model semantics.
- The rules say what can be *derived*
    - derivability
    - provability
- Entailment
    - is closely related to the *meaning* of things
    - higher confidence in model semantics than in a bunch of rules
    - *The* semantics given by the standard, rules are just "informative"
    - can't be directly checked mechanically (∞ many interpretations)

Entailment and Derivability

- Actually, two different notions!
- Entailment is defined using the model semantics.
- The rules say what can be *derived*
  - derivability
  - provability
- Entailment
  - is closely related to the *meaning* of things
  - higher confidence in model semantics than in a bunch of rules
  - *The* semantics given by the standard, rules are just "informative"
  - can't be directly checked mechanically (∞ many interpretations)
- Derivability

Entailment and Derivability

- Actually, two different notions!
- Entailment is defined using the model semantics.
- The rules say what can be *derived*
  - derivability
  - provability
- Entailment
  - is closely related to the *meaning* of things
  - higher confidence in model semantics than in a bunch of rules
  - *The* semantics given by the standard, rules are just "informative"
  - can't be directly checked mechanically (∞ many interpretations)
- Derivability
  - can be checked mechanically

Entailment and Derivability

- Actually, two different notions!
- Entailment is defined using the model semantics.
- The rules say what can be *derived*
  - derivability
  - provability
- Entailment
  - is closely related to the *meaning* of things
  - higher confidence in model semantics than in a bunch of rules
  - *The* semantics given by the standard, rules are just "informative"
  - can't be directly checked mechanically (∞ many interpretations)
- Derivability
  - can be checked mechanically
  - forward or backward chaining

Entailment and Derivability

- Actually, two different notions!
- Entailment is defined using the model semantics.
- The rules say what can be *derived*
  - derivability
  - provability
- Entailment
  - is closely related to the *meaning* of things
  - higher confidence in model semantics than in a bunch of rules
  - *The* semantics given by the standard, rules are just "informative"
  - can't be directly checked mechanically (∞ many interpretations)
- Derivability
  - can be checked mechanically
  - forward or backward chaining
- Want these notions to correspond:

Entailment and Derivability

- Actually, two different notions!
- Entailment is defined using the model semantics.
- The rules say what can be *derived*
  - derivability
  - provability
- Entailment
  - is closely related to the *meaning* of things
  - higher confidence in model semantics than in a bunch of rules
  - *The* semantics given by the standard, rules are just "informative"
  - can't be directly checked mechanically (∞ many interpretations)
- Derivability
  - can be checked mechanically
  - forward or backward chaining
- Want these notions to correspond:
  - $\mathcal{A} \models \mathcal{B}$ iff $\mathcal{B}$ can be derived from $\mathcal{A}$

Soundness

- Two directions:

Soundness

- Two directions:
  1. If $\mathcal{A} \models \mathcal{B}$ then $\mathcal{B}$ can be derived from $\mathcal{A}$

Soundness

- Two directions:
  1. If $\mathcal{A} \models \mathcal{B}$ then $\mathcal{B}$ can be derived from $\mathcal{A}$
  2. If $\mathcal{B}$ can be derived from $\mathcal{A}$ then $\mathcal{A} \models \mathcal{B}$

Soundness

- Two directions:
  1. If $\mathcal{A} \models \mathcal{B}$ then $\mathcal{B}$ can be derived from $\mathcal{A}$
  2. If $\mathcal{B}$ can be derived from $\mathcal{A}$ then $\mathcal{A} \models \mathcal{B}$
- Nr. 2 usually considered more important:

Soundness

- Two directions:
  1. If $\mathcal{A} \models \mathcal{B}$ then $\mathcal{B}$ can be derived from $\mathcal{A}$
  2. If $\mathcal{B}$ can be derived from $\mathcal{A}$ then $\mathcal{A} \models \mathcal{B}$
- Nr. 2 usually considered more important:
- If the calculus says that something is entailed then it is really entailed.

Soundness

- Two directions:
  1. If $\mathcal{A} \models \mathcal{B}$ then $\mathcal{B}$ can be derived from $\mathcal{A}$
  2. If $\mathcal{B}$ can be derived from $\mathcal{A}$ then $\mathcal{A} \models \mathcal{B}$
- Nr. 2 usually considered more important:
- If the calculus says that something is entailed then it is really entailed.
- The calculus gives no "wrong" answers.

Soundness

- Two directions:
  1. If $\mathcal{A} \models \mathcal{B}$ then $\mathcal{B}$ can be derived from $\mathcal{A}$
  2. If $\mathcal{B}$ can be derived from $\mathcal{A}$ then $\mathcal{A} \models \mathcal{B}$
- Nr. 2 usually considered more important:
- If the calculus says that something is entailed then it is really entailed.
- The calculus gives no “wrong” answers.
- This is known as *soundness*

Soundness

- Two directions:
  1. If $\mathcal{A} \models \mathcal{B}$ then $\mathcal{B}$ can be derived from $\mathcal{A}$
  2. If $\mathcal{B}$ can be derived from $\mathcal{A}$ then $\mathcal{A} \models \mathcal{B}$
- Nr. 2 usually considered more important:
- If the calculus says that something is entailed then it is really entailed.
- The calculus gives no "wrong" answers.
- This is known as *soundness*
- The calculus is said to be *sound* (w.r.t. the model semantics)

Showing Soundness

- Soundness of every rule has to be (manually) checked!

Showing Soundness

- Soundness of every rule has to be (manually) checked!
- E.g. rdfs11,

$$\frac{A \sqsubseteq B \qquad B \sqsubseteq C}{A \sqsubseteq C}\,\text{rdfs11}$$

Showing Soundness

- Soundness of every rule has to be (manually) checked!
- E.g. rdfs11,

$$\frac{A \sqsubseteq B \qquad B \sqsubseteq C}{A \sqsubseteq C}\ \text{rdfs11}$$

- Soundness means that

Showing Soundness

- Soundness of every rule has to be (manually) checked!
- E.g. rdfs11,

$$\frac{A \sqsubseteq B \qquad B \sqsubseteq C}{A \sqsubseteq C}\,\text{rdfs11}$$

- Soundness means that
  - For any choice of three classes A, B, C

Showing Soundness

- Soundness of every rule has to be (manually) checked!
- E.g. rdfs11,

$$\frac{A \sqsubseteq B \qquad B \sqsubseteq C}{A \sqsubseteq C}\ \text{rdfs11}$$

- Soundness means that
  - For any choice of three classes A, B, C
  - $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$

Showing Soundness

- Soundness of every rule has to be (manually) checked!
- E.g. rdfs11,

$$\frac{A \sqsubseteq B \qquad B \sqsubseteq C}{A \sqsubseteq C}\ \text{rdfs11}$$

- Soundness means that
  - For any choice of three classes A, B, C
  - $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$
- Proof:

Showing Soundness

- Soundness of every rule has to be (manually) checked!
- E.g. rdfs11,

$$\frac{A \sqsubseteq B \qquad B \sqsubseteq C}{A \sqsubseteq C}\ \text{rdfs11}$$

- Soundness means that
  - For any choice of three classes A, B, C
  - $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$
- Proof:
  - Let $\mathcal{I}$ be an arbitrary interpretation with $\mathcal{I} \models \{A \sqsubseteq B, B \sqsubseteq C\}$

Showing Soundness

- Soundness of every rule has to be (manually) checked!
- E.g. rdfs11,

$$\frac{A \sqsubseteq B \qquad B \sqsubseteq C}{A \sqsubseteq C}\text{ rdfs11}$$

- Soundness means that
  - For any choice of three classes A, B, C
  - $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$
- Proof:
  - Let $\mathcal{I}$ be an arbitrary interpretation with $\mathcal{I} \models \{A \sqsubseteq B, B \sqsubseteq C\}$
  - Then by model semantics, $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ and $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$

Showing Soundness

- Soundness of every rule has to be (manually) checked!
- E.g. rdfs11,

$$\frac{A \sqsubseteq B \qquad B \sqsubseteq C}{A \sqsubseteq C}\ \text{rdfs11}$$

- Soundness means that
  - For any choice of three classes A, B, C
  - $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$
- Proof:
  - Let $\mathcal{I}$ be an arbitrary interpretation with $\mathcal{I} \models \{A \sqsubseteq B, B \sqsubseteq C\}$
  - Then by model semantics, $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ and $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
  - By set theory, $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$

Showing Soundness

- Soundness of every rule has to be (manually) checked!
- E.g. rdfs11,

$$\frac{A \sqsubseteq B \qquad B \sqsubseteq C}{A \sqsubseteq C}\ \text{rdfs11}$$

- Soundness means that
  - For any choice of three classes A, B, C
  - $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$
- Proof:
  - Let $\mathcal{I}$ be an arbitrary interpretation with $\mathcal{I} \models \{A \sqsubseteq B, B \sqsubseteq C\}$
  - Then by model semantics, $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ and $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
  - By set theory, $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
  - By model semantics, $\mathcal{I} \models A \sqsubseteq C$

Showing Soundness

- Soundness of every rule has to be (manually) checked!
- E.g. rdfs11,

$$\frac{A \sqsubseteq B \qquad B \sqsubseteq C}{A \sqsubseteq C}\ \text{rdfs11}$$

- Soundness means that
  - For any choice of three classes A, B, C
  - $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$
- Proof:
  - Let $\mathcal{I}$ be an arbitrary interpretation with $\mathcal{I} \models \{A \sqsubseteq B, B \sqsubseteq C\}$
  - Then by model semantics, $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ and $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
  - By set theory, $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
  - By model semantics, $\mathcal{I} \models A \sqsubseteq C$
  - Q.E.D.

Showing Soundness

- Soundness of every rule has to be (manually) checked!
- E.g. rdfs11,

$$\frac{A \sqsubseteq B \qquad B \sqsubseteq C}{A \sqsubseteq C}\ \text{rdfs11}$$

- Soundness means that
  - For any choice of three classes A, B, C
  - $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$
- Proof:
  - Let $\mathcal{I}$ be an arbitrary interpretation with $\mathcal{I} \models \{A \sqsubseteq B, B \sqsubseteq C\}$
  - Then by model semantics, $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ and $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
  - By set theory, $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
  - By model semantics, $\mathcal{I} \models A \sqsubseteq C$
  - Q.E.D.
- This can be done similarly for all of the rules.

Showing Soundness

- Soundness of every rule has to be (manually) checked!
- E.g. rdfs11,

$$\frac{A \sqsubseteq B \qquad B \sqsubseteq C}{A \sqsubseteq C}\ \text{rdfs11}$$

- Soundness means that
  - For any choice of three classes A, B, C
  - $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$
- Proof:
  - Let $\mathcal{I}$ be an arbitrary interpretation with $\mathcal{I} \models \{A \sqsubseteq B, B \sqsubseteq C\}$
  - Then by model semantics, $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ and $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
  - By set theory, $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
  - By model semantics, $\mathcal{I} \models A \sqsubseteq C$
  - Q.E.D.
- This can be done similarly for all of the rules.
  - All given SE/RDF/RDFS rules are sound w.r.t. the model semantics!

Completeness

- Two directions:

Completeness

- Two directions:
  1. If $\mathcal{A} \models \mathcal{B}$ then $\mathcal{B}$ can be derived from $\mathcal{A}$

Completeness

- Two directions:
  1. If $\mathcal{A} \models \mathcal{B}$ then $\mathcal{B}$ can be derived from $\mathcal{A}$
  2. If $\mathcal{B}$ can be derived from $\mathcal{A}$ then $\mathcal{A} \models \mathcal{B}$

Completeness

- Two directions:
  1. If $\mathcal{A} \models \mathcal{B}$ then $\mathcal{B}$ can be derived from $\mathcal{A}$
  2. If $\mathcal{B}$ can be derived from $\mathcal{A}$ then $\mathcal{A} \models \mathcal{B}$
- Nr. 1 says that any entailment can be found using the rules.

Completeness

- Two directions:
  1. If $\mathcal{A} \models \mathcal{B}$ then $\mathcal{B}$ can be derived from $\mathcal{A}$
  2. If $\mathcal{B}$ can be derived from $\mathcal{A}$ then $\mathcal{A} \models \mathcal{B}$
- Nr. 1 says that any entailment can be found using the rules.
- I.e. we have "enough" rules.

Completeness

- Two directions:
  1. If $\mathcal{A} \models \mathcal{B}$ then $\mathcal{B}$ can be derived from $\mathcal{A}$
  2. If $\mathcal{B}$ can be derived from $\mathcal{A}$ then $\mathcal{A} \models \mathcal{B}$
- Nr. 1 says that any entailment can be found using the rules.
- I.e. we have "enough" rules.
- Can't be checked separately for each rule, only for whole rule set

Completeness

- Two directions:
  1. If $\mathcal{A} \models \mathcal{B}$ then $\mathcal{B}$ can be derived from $\mathcal{A}$
  2. If $\mathcal{B}$ can be derived from $\mathcal{A}$ then $\mathcal{A} \models \mathcal{B}$
- Nr. 1 says that any entailment can be found using the rules.
- I.e. we have "enough" rules.
- Can't be checked separately for each rule, only for whole rule set
- Proofs are more complicated than soundness

Simple Entailment: Completeness

- Simple entailment is entailment

Simple Entailment: Completeness

- Simple entailment is entailment
  - With blank nodes and literals

Simple Entailment: Completeness

- Simple entailment is entailment
  - With blank nodes and literals
  - but without RDFS

Simple Entailment: Completeness

- Simple entailment is entailment
  - With blank nodes and literals
  - but without RDFS
  - and without RDF axioms like `rdf:type rdf:Property .`

Simple Entailment: Completeness

- Simple entailment is entailment
  - With blank nodes and literals
  - but without RDFS
  - and without RDF axioms like `rdf:type rdf:Property .`
- se1 and se2 are complete for simple entailment, i.e.

Simple Entailment: Completeness

- Simple entailment is entailment
  - With blank nodes and literals
  - but without RDFS
  - and without RDF axioms like `rdf:type rdf:Property .`
- se1 and se2 are complete for simple entailment, i.e.

  if $\mathcal{A}$ simply entails $\mathcal{B}$

Simple Entailment: Completeness

- Simple entailment is entailment
  - With blank nodes and literals
  - but without RDFS
  - and without RDF axioms like `rdf:type rdf:Property .`
- se1 and se2 are complete for simple entailment, i.e.

  if $\mathcal{A}$ simply entails $\mathcal{B}$

  then $\mathcal{A}$ can be extended with se1 and se2 to $\mathcal{A}'$ with $\mathcal{B} \subseteq \mathcal{A}'$.

Simple Entailment: Completeness

- Simple entailment is entailment
  - With blank nodes and literals
  - but without RDFS
  - and without RDF axioms like `rdf:type rdf:Property .`
- se1 and se2 are complete for simple entailment, i.e.

    if $\mathcal{A}$ simply entails $\mathcal{B}$

    then $\mathcal{A}$ can be extended with se1 and se2 to $\mathcal{A}'$ with $\mathcal{B} \subseteq \mathcal{A}'$.
- (requires blank node IDs in $\mathcal{A}$ and $\mathcal{B}$ to be disjoint)

Rules for (simplified) RDF/RDFS

- See Foundations book, Sect. 3.3

Rules for (simplified) RDF/RDFS

- See Foundations book, Sect. 3.3
- Many rules and axioms not needed for our “simplified” RDF/RDFS

Rules for (simplified) RDF/RDFS

- See Foundations book, Sect. 3.3
- Many rules and axioms not needed for our "simplified" RDF/RDFS
  - `rdfs:range rdfs:domain rdfs:Class ...`

Rules for (simplified) RDF/RDFS

- See Foundations book, Sect. 3.3
- Many rules and axioms not needed for our “simplified” RDF/RDFS
  - `rdfs:range rdfs:domain rdfs:Class ...`
- Important rules for us:

$$\frac{\mathrm{dom}(r, A) \qquad r(x, y)}{A(x)}\ \mathrm{rdfs2} \qquad\qquad \frac{\mathrm{rg}(r, B) \qquad r(x, y)}{B(y)}\ \mathrm{rdfs3}$$

Rules for (simplified) RDF/RDFS

- See Foundations book, Sect. 3.3
- Many rules and axioms not needed for our "simplified" RDF/RDFS
  - `rdfs:range rdfs:domain rdfs:Class ...`
- Important rules for us:

$$\frac{\mathrm{dom}(r, A) \qquad r(x, y)}{A(x)}\ \mathrm{rdfs2} \qquad\qquad \frac{\mathrm{rg}(r, B) \qquad r(x, y)}{B(y)}\ \mathrm{rdfs3}$$

$$\frac{r \sqsubseteq s \qquad s \sqsubseteq t}{r \sqsubseteq t}\ \mathrm{rdfs5} \qquad \frac{}{r \sqsubseteq r}\ \mathrm{rdfs6} \qquad \frac{r \sqsubseteq s \qquad r(x, y)}{s(x, y)}\ \mathrm{rdfs7}$$

Rules for (simplified) RDF/RDFS

- See Foundations book, Sect. 3.3
- Many rules and axioms not needed for our "simplified" RDF/RDFS
  - `rdfs:range rdfs:domain rdfs:Class ...`
- Important rules for us:

$$\frac{\mathrm{dom}(r, A) \qquad r(x, y)}{A(x)}\ \mathrm{rdfs2} \qquad\qquad \frac{\mathrm{rg}(r, B) \qquad r(x, y)}{B(y)}\ \mathrm{rdfs3}$$

$$\frac{r \sqsubseteq s \qquad s \sqsubseteq t}{r \sqsubseteq t}\ \mathrm{rdfs5} \qquad \frac{}{r \sqsubseteq r}\ \mathrm{rdfs6} \qquad \frac{r \sqsubseteq s \qquad r(x, y)}{s(x, y)}\ \mathrm{rdfs7}$$

$$\frac{A \sqsubseteq B \qquad A(x)}{B(x)}\ \mathrm{rdfs9} \qquad \frac{}{A \sqsubseteq A}\ \mathrm{rdfs10} \qquad \frac{A \sqsubseteq B \qquad B \sqsubseteq C}{A \sqsubseteq C}\ \mathrm{rdfs11}$$

Complete?

- These rules are *not* complete for our RDF/RDFS semantics

Complete?

- These rules are *not* complete for our RDF/RDFS semantics
- For instance

$$\{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\} \models \mathrm{rg}(\mathit{loves}, \mathit{Person})$$

Complete?

- These rules are *not* complete for our RDF/RDFS semantics
- For instance

$$\{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\} \models \mathrm{rg}(\mathit{loves}, \mathit{Person})$$

- Because for every interpretation $\mathcal{I}$,

Complete?

- These rules are *not* complete for our RDF/RDFS semantics
- For instance

$$\{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\} \models \mathrm{rg}(\mathit{loves}, \mathit{Person})$$

- Because for every interpretation $\mathcal{I}$,
  - if $\mathcal{I} \models \{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\}$

Complete?

- These rules are *not* complete for our RDF/RDFS semantics
- For instance

$$\{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\} \models \mathrm{rg}(\mathit{loves}, \mathit{Person})$$

- Because for every interpretation $\mathcal{I}$,
  - if $\mathcal{I} \models \{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\}$
  - then by semantics, for all $\langle x, y\rangle \in \mathit{loves}^{\mathcal{I}}$, $y \in \mathit{Beloved}^{\mathcal{I}}$; and $\mathit{Beloved}^{\mathcal{I}} \subseteq \mathit{Person}^{\mathcal{I}}$.

Complete?

- These rules are *not* complete for our RDF/RDFS semantics
- For instance

$$\{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\} \models \mathrm{rg}(\mathit{loves}, \mathit{Person})$$

- Because for every interpretation $\mathcal{I}$,
  - if $\mathcal{I} \models \{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\}$
  - then by semantics, for all $\langle x, y\rangle \in \mathit{loves}^{\mathcal{I}}$, $y \in \mathit{Beloved}^{\mathcal{I}}$; and $\mathit{Beloved}^{\mathcal{I}} \subseteq \mathit{Person}^{\mathcal{I}}$.
  - Therefore, by set theory, for all $\langle x, y\rangle \in \mathit{loves}^{\mathcal{I}}$, $y \in \mathit{Person}^{\mathcal{I}}$.

Complete?

- These rules are *not* complete for our RDF/RDFS semantics
- For instance

$$\{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\} \models \mathrm{rg}(\mathit{loves}, \mathit{Person})$$

- Because for every interpretation $\mathcal{I}$,
  - if $\mathcal{I} \models \{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\}$
  - then by semantics, for all $\langle x, y\rangle \in \mathit{loves}^{\mathcal{I}}$, $y \in \mathit{Beloved}^{\mathcal{I}}$; and $\mathit{Beloved}^{\mathcal{I}} \subseteq \mathit{Person}^{\mathcal{I}}$.
  - Therefore, by set theory, for all $\langle x, y\rangle \in \mathit{loves}^{\mathcal{I}}$, $y \in \mathit{Person}^{\mathcal{I}}$.
  - By semantics, $\mathcal{I} \models \mathrm{rg}(\mathit{loves}, \mathit{Person})$

Complete?

- These rules are *not* complete for our RDF/RDFS semantics
- For instance

$$\{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\} \models \mathrm{rg}(\mathit{loves}, \mathit{Person})$$

- Because for every interpretation $\mathcal{I}$,
  - if $\mathcal{I} \models \{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\}$
  - then by semantics, for all $\langle x, y\rangle \in \mathit{loves}^{\mathcal{I}}$, $y \in \mathit{Beloved}^{\mathcal{I}}$; and $\mathit{Beloved}^{\mathcal{I}} \subseteq \mathit{Person}^{\mathcal{I}}$.
  - Therefore, by set theory, for all $\langle x, y\rangle \in \mathit{loves}^{\mathcal{I}}$, $y \in \mathit{Person}^{\mathcal{I}}$.
  - By semantics, $\mathcal{I} \models \mathrm{rg}(\mathit{loves}, \mathit{Person})$
- But there is no way to derive this using the given rules

Complete?

- These rules are *not* complete for our RDF/RDFS semantics
- For instance

$$\{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\} \models \mathrm{rg}(\mathit{loves}, \mathit{Person})$$

- Because for every interpretation $\mathcal{I}$,
  - if $\mathcal{I} \models \{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\}$
  - then by semantics, for all $\langle x, y\rangle \in \mathit{loves}^{\mathcal{I}}$, $y \in \mathit{Beloved}^{\mathcal{I}}$; and $\mathit{Beloved}^{\mathcal{I}} \subseteq \mathit{Person}^{\mathcal{I}}$.
  - Therefore, by set theory, for all $\langle x, y\rangle \in \mathit{loves}^{\mathcal{I}}$, $y \in \mathit{Person}^{\mathcal{I}}$.
  - By semantics, $\mathcal{I} \models \mathrm{rg}(\mathit{loves}, \mathit{Person})$
- But there is no way to derive this using the given rules
  - There is no rule which allows to derive a range statement.

Complete?

- These rules are *not* complete for our RDF/RDFS semantics
- For instance

$$\{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\} \models \mathrm{rg}(\mathit{loves}, \mathit{Person})$$

- Because for every interpretation $\mathcal{I}$,
  - if $\mathcal{I} \models \{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\}$
  - then by semantics, for all $\langle x, y\rangle \in \mathit{loves}^{\mathcal{I}}$, $y \in \mathit{Beloved}^{\mathcal{I}}$; and $\mathit{Beloved}^{\mathcal{I}} \subseteq \mathit{Person}^{\mathcal{I}}$.
  - Therefore, by set theory, for all $\langle x, y\rangle \in \mathit{loves}^{\mathcal{I}}$, $y \in \mathit{Person}^{\mathcal{I}}$.
  - By semantics, $\mathcal{I} \models \mathrm{rg}(\mathit{loves}, \mathit{Person})$
- But there is no way to derive this using the given rules
  - There is no rule which allows to derive a range statement.
- We could now add rules to make the system complete

Complete?

- These rules are *not* complete for our RDF/RDFS semantics
- For instance

$$\{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\} \models \mathrm{rg}(\mathit{loves}, \mathit{Person})$$

- Because for every interpretation $\mathcal{I}$,
  - if $\mathcal{I} \models \{\mathrm{rg}(\mathit{loves}, \mathit{Beloved}), \mathit{Beloved} \sqsubseteq \mathit{Person}\}$
  - then by semantics, for all $\langle x, y\rangle \in \mathit{loves}^{\mathcal{I}}$, $y \in \mathit{Beloved}^{\mathcal{I}}$; and $\mathit{Beloved}^{\mathcal{I}} \subseteq \mathit{Person}^{\mathcal{I}}$.
  - Therefore, by set theory, for all $\langle x, y\rangle \in \mathit{loves}^{\mathcal{I}}$, $y \in \mathit{Person}^{\mathcal{I}}$.
  - By semantics, $\mathcal{I} \models \mathrm{rg}(\mathit{loves}, \mathit{Person})$
- But there is no way to derive this using the given rules
  - There is no rule which allows to derive a range statement.
- We could now add rules to make the system complete
- Won't bother to do that now. Will get completeness for OWL.

Outlook

- RDFS allows some simple modelling: "all ladies are persons"

Outlook

- RDFS allows some simple modelling: “all ladies are persons”
- The following lectures will be about OWL

Outlook

- RDFS allows some simple modelling: "all ladies are persons"
- The following lectures will be about OWL
- Will allow to say things like

Outlook

- RDFS allows some simple modelling: "all ladies are persons"
- The following lectures will be about OWL
- Will allow to say things like
    - Every car has a motor

Outlook

- RDFS allows some simple modelling: “all ladies are persons”
- The following lectures will be about OWL
- Will allow to say things like
    - Every car has a motor
    - Every car has at least three parts of type wheel

Outlook

- RDFS allows some simple modelling: "all ladies are persons"
- The following lectures will be about OWL
- Will allow to say things like
  - Every car has a motor
  - Every car has at least three parts of type wheel
  - A mother is a person who is female and has at least one child

Outlook

- RDFS allows some simple modelling: “all ladies are persons”
- The following lectures will be about OWL
- Will allow to say things like
  - Every car has a motor
  - Every car has at least three parts of type wheel
  - A mother is a person who is female and has at least one child
  - The friends of my friends are also my friends

Outlook

- RDFS allows some simple modelling: "all ladies are persons"
- The following lectures will be about OWL
- Will allow to say things like
  - Every car has a motor
  - Every car has at least three parts of type wheel
  - A mother is a person who is female and has at least one child
  - The friends of my friends are also my friends
  - A metropolis is a town with at least a million inhabitants

Outlook

- RDFS allows some simple modelling: “all ladies are persons”
- The following lectures will be about OWL
- Will allow to say things like
  - Every car has a motor
  - Every car has at least three parts of type wheel
  - A mother is a person who is female and has at least one child
  - The friends of my friends are also my friends
  - A metropolis is a town with at least a million inhabitants
  - . . . and many more

Outlook

- RDFS allows some simple modelling: “all ladies are persons”
- The following lectures will be about OWL
- Will allow to say things like
    - Every car has a motor
    - Every car has at least three parts of type wheel
    - A mother is a person who is female and has at least one child
    - The friends of my friends are also my friends
    - A metropolis is a town with at least a million inhabitants
    - . . . and many more
- Modeling will not be done by writing triples manually:

Outlook

- RDFS allows some simple modelling: "all ladies are persons"
- The following lectures will be about OWL
- Will allow to say things like
  - Every car has a motor
  - Every car has at least three parts of type wheel
  - A mother is a person who is female and has at least one child
  - The friends of my friends are also my friends
  - A metropolis is a town with at least a million inhabitants
  - . . . and many more
- Modeling will not be done by writing triples manually:
- Will use ontology editor Protégé.