IN3060/4060 – Semantic Technologies – Spring 2021

Lecture 7: RDF and RDFS semantics

Jieying Chen

26th February 2021

Department of Informatics

University of Oslo

Outline

Semantics—why do we need it?

A formal semantics for RDF and RDFS became necessary because

Semantics—why do we need it?

A formal semantics for RDF and RDFS became necessary because

1. the previous informal specification

Semantics—why do we need it?

A formal semantics for RDF and RDFS became necessary because

1. the previous informal specification
2. left plenty of room for interpretation of conclusions, whence

Semantics—why do we need it?

A formal semantics for RDF and RDFS became necessary because

1. the previous informal specification
2. left plenty of room for interpretation of conclusions, whence
3. triple stores sometimes answered queries differently, thereby

Semantics—why do we need it?

A formal semantics for RDF and RDFS became necessary because

1. the previous informal specification
2. left plenty of room for interpretation of conclusions, whence
3. triple stores sometimes answered queries differently, thereby
4. obstructing interoperability and interchangeability.

Semantics—why do we need it?

A formal semantics for RDF and RDFS became necessary because

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

Semantics—why do we need it?

A formal semantics for RDF and RDFS became necessary because

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

But RDF was supposed to be the **data liberation movement**

Example: What is the meaning of blank nodes?

Example: What is the meaning of blank nodes?

Names of people who co-starred with Johnny Depp

```
SELECT DISTINCT ?coStar WHERE {
    _:m dbo:starring [foaf:name "Johnny Depp"@en], [foaf:name ?coStar] .
}
```

Example: What is the meaning of blank nodes?

Names of people who co-starred with Johnny Depp

```
SELECT DISTINCT ?coStar WHERE {
    _:m dbo:starring [foaf:name "Johnny Depp"@en], [foaf:name ?coStar] .
}
```

SPARQL must

Example: What is the meaning of blank nodes?

Names of people who co-starred with Johnny Depp

```
SELECT DISTINCT ?coStar WHERE {
    _:m dbo:starring [foaf:name "Johnny Depp"@en], [foaf:name ?coStar] .
}
```

SPARQL must

- match the query to graph patterns

Example: What is the meaning of blank nodes?

Names of people who co-starred with Johnny Depp

```
SELECT DISTINCT ?coStar WHERE {
    _:m dbo:starring [foaf:name "Johnny Depp"@en], [foaf:name ?coStar] .
}
```

SPARQL must

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

Example: What is the meaning of blank nodes?

Names of people who co-starred with Johnny Depp

```
SELECT DISTINCT ?coStar WHERE {
    _:m dbo:starring [foaf:name "Johnny Depp"@en], [foaf:name ?coStar] .
}
```

SPARQL must

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

But,

- which values are to count?

Example: What is the meaning of blank nodes?

Names of people who co-starred with Johnny Depp

```
SELECT DISTINCT ?coStar WHERE {
    _:m dbo:starring [foaf:name "Johnny Depp"@en], [foaf:name ?coStar] .
}
```

SPARQL must

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

But,

- which values are to count?
- the problem becomes more acute under reasoning.

Example: What is the meaning of blank nodes?

Names of people who co-starred with Johnny Depp

```
SELECT DISTINCT ?coStar WHERE {
    _:m dbo:starring [foaf:name "Johnny Depp"@en], [foaf:name ?coStar] .
}
```

SPARQL must

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

But,

- which values are to count?
- the problem becomes more acute under reasoning.
- Should a value for `foaf:familyname` match a query for `foaf:name`?

Example: What is the meaning of blank nodes?

Names of people who co-starred with Johnny Depp

```
SELECT DISTINCT ?coStar WHERE {
    _:m dbo:starring [foaf:name "Johnny Depp"@en], [foaf:name ?coStar] .
}
```

SPARQL must

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

But,

- which values are to count?
- the problem becomes more acute under reasoning.
- Should a value for `foaf:familyname` match a query for `foaf:name`?
- Are blanks in SPARQL the same as blanks in RDF?

Example: What is the meaning of blank nodes?

Names of people who co-starred with Johnny Depp

```
SELECT DISTINCT ?coStar WHERE {
    _:m dbo:starring [foaf:name "Johnny Depp"@en], [foaf:name ?coStar] .
}
```

SPARQL must

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

But,

- which values are to count?
- the problem becomes more acute under reasoning.
- Should a value for `foaf:familyname` match a query for `foaf:name`?
- Are blanks in SPARQL the same as blanks in RDF?

Another look at the Semantic Web cake

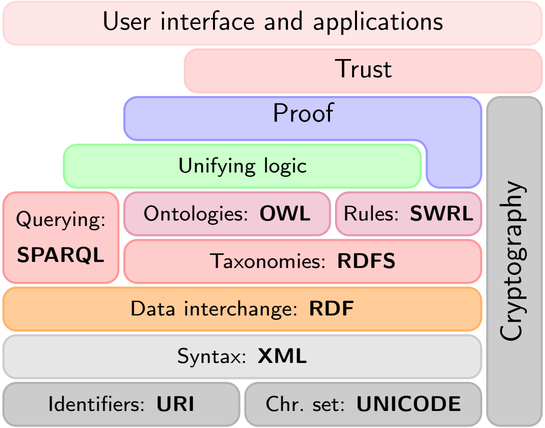


Figure: Semantic Web Stack

Absolute precisision required

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

Absolute precisision required

- RDF is to serve as the foundation of the entire Semantic Web stack.
- Can afford no ambiguity in interpreting RDF, SPARQL, etc.

Absolute precisision required

- RDF is to serve as the foundation of the entire Semantic Web stack.
- Can afford no ambiguity in interpreting RDF, SPARQL, etc.
- Two styles of semantics for e.g. programming languages

Absolute precisision required

- RDF is to serve as the foundation of the entire Semantic Web stack.
- Can afford no ambiguity in interpreting RDF, SPARQL, etc.
- Two styles of semantics for e.g. programming languages
  - Declarative (what does it mean)

Absolute precisision required

- RDF is to serve as the foundation of the entire Semantic Web stack.
- Can afford no ambiguity in interpreting RDF, SPARQL, etc.
- Two styles of semantics for e.g. programming languages
  - Declarative (what does it mean)
  - Operational (how is it computed)

Absolute precisision required

- RDF is to serve as the foundation of the entire Semantic Web stack.
- Can afford no ambiguity in interpreting RDF, SPARQL, etc.
- Two styles of semantics for e.g. programming languages
  - Declarative (what does it mean)
  - Operational (how is it computed)
- RDF represents information, not instructions

Absolute precisision required

- RDF is to serve as the foundation of the entire Semantic Web stack.
- Can afford no ambiguity in interpreting RDF, SPARQL, etc.
- Two styles of semantics for e.g. programming languages
  - Declarative (what does it mean)
  - Operational (how is it computed)
- RDF represents information, not instructions
  - Want a declarative style semantics

Absolute precisision required

- RDF is to serve as the foundation of the entire Semantic Web stack.
- Can afford no ambiguity in interpreting RDF, SPARQL, etc.
- Two styles of semantics for e.g. programming languages
  - Declarative (what does it mean)
  - Operational (how is it computed)
- RDF represents information, not instructions
  - Want a declarative style semantics
- We furnish RDF with a model semantics like a logic

Absolute precisision required

- RDF is to serve as the foundation of the entire Semantic Web stack.
- Can afford no ambiguity in interpreting RDF, SPARQL, etc.
- Two styles of semantics for e.g. programming languages
  - Declarative (what does it mean)
  - Operational (how is it computed)
- RDF represents information, not instructions
  - Want a declarative style semantics
- We furnish RDF with a model semantics like a logic
- Specifies how the different components should be interpreted

Absolute precisision required

- RDF is to serve as the foundation of the entire Semantic Web stack.
- Can afford no ambiguity in interpreting RDF, SPARQL, etc.
- Two styles of semantics for e.g. programming languages
  - Declarative (what does it mean)
  - Operational (how is it computed)
- RDF represents information, not instructions
  - Want a declarative style semantics
- We furnish RDF with a model semantics like a logic
- Specifies how the different components should be interpreted
- And what entailment should be taken to mean.

Outline

Formal semantics

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

Formal semantics

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

Formal semantics

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

Formal semantics

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

Formal semantics

- The study of how to model the meaning of a logical calculus.
- A logical calculus consists of:
  - A finite set of symbols,
  - a grammar, which specifies the formulae,
  - a set of axioms and inference rules from which we construct proofs.

Formal semantics

- The study of how to model the meaning of a logical calculus.
- A logical calculus consists of:
  - A finite set of symbols,
  - a grammar, which specifies the formulae,
  - a set of axioms and inference rules from which we construct proofs.
- A logical calculus can be defined apart from any interpretation.

Formal semantics

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

Formal semantics

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

Formal semantics

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

Derivations

A proof typically looks something like this:

Derivations

A proof typically looks something like this:

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

Derivations

A proof typically looks something like this:

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

Where each line represents an application of an inference rule.

Derivations

A proof typically looks something like this:

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

Where each line represents an application of an inference rule.

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

Derivations

A proof typically looks something like this:

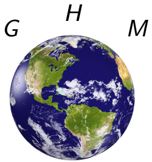
$$\dfrac{\dfrac{P \vdash Q, P \qquad Q, P \vdash Q}{P \to Q, P \vdash Q} \qquad \dfrac{R \vdash Q, P \qquad Q, R \vdash Q}{P \to Q, R \vdash Q}}{\dfrac{P \to Q, P \lor R \vdash Q}{P \to Q \vdash (P \lor R) \to Q}}$$

Where each line represents an application of an inference rule.

- How do we know that the inference rules are well-chosen?
- Which manipulations derive conclusions that hold in the real world?

Finding out stuff about the World

The "Real World"



G: Aristotle was Greek

H: Aristotle was human

M: Aristotle was mortal

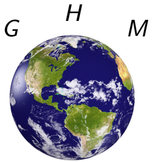
Finding out stuff about the World

Statements

$G \rightarrow H$

$H \rightarrow M$

The "Real World"



G: Aristotle was Greek
H: Aristotle was human
M: Aristotle was mortal

Finding out stuff about the World

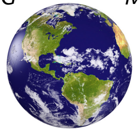
Statements

$G \rightarrow H$

$H \rightarrow M$

The "Real World"

G H M



G: Aristotle was Greek

H: Aristotle was human

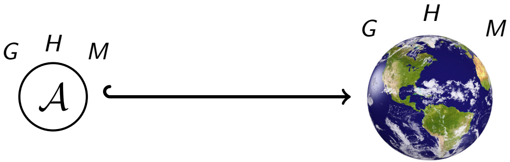
M: Aristotle was mortal

Finding out stuff about the World

Statements

Abstract to G, H, M

The "Real World"



$G \to H$

$H \to M$

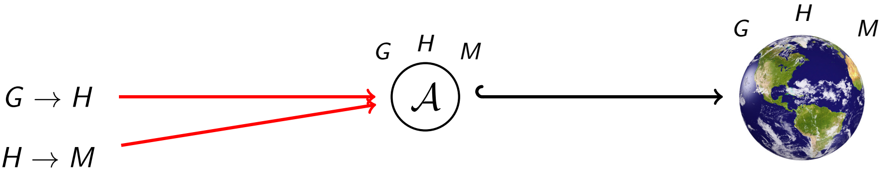
G: Aristotle was Greek

H: Aristotle was human

M: Aristotle was mortal

Finding out stuff about the World

Statements | Abstract to G, H, M | The "Real World"

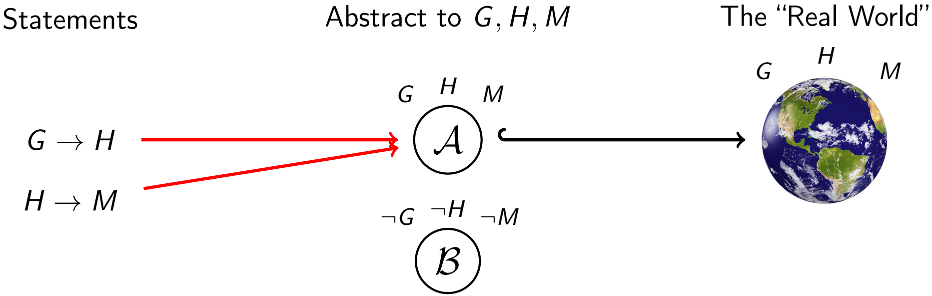


G: Aristotle was Greek

H: Aristotle was human

M: Aristotle was mortal

Finding out stuff about the World

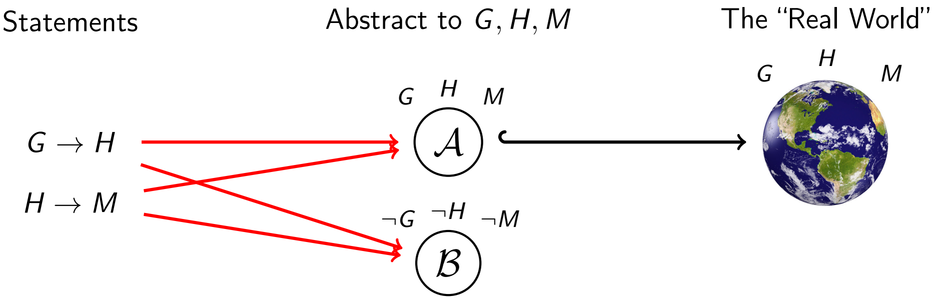


G: Aristotle was Greek

H: Aristotle was human

M: Aristotle was mortal

Finding out stuff about the World

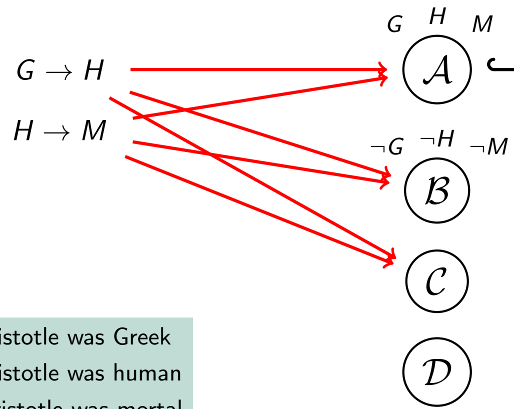


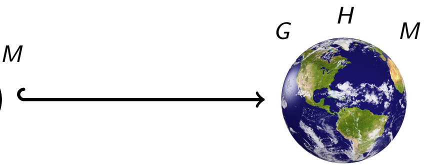
G: Aristotle was Greek
H: Aristotle was human
M: Aristotle was mortal

$\mathcal{A}$: intended model
$\mathcal{B}\ldots$: unintended models

Finding out stuff about the World

Statements | Abstract to G, H, M | The "Real World"

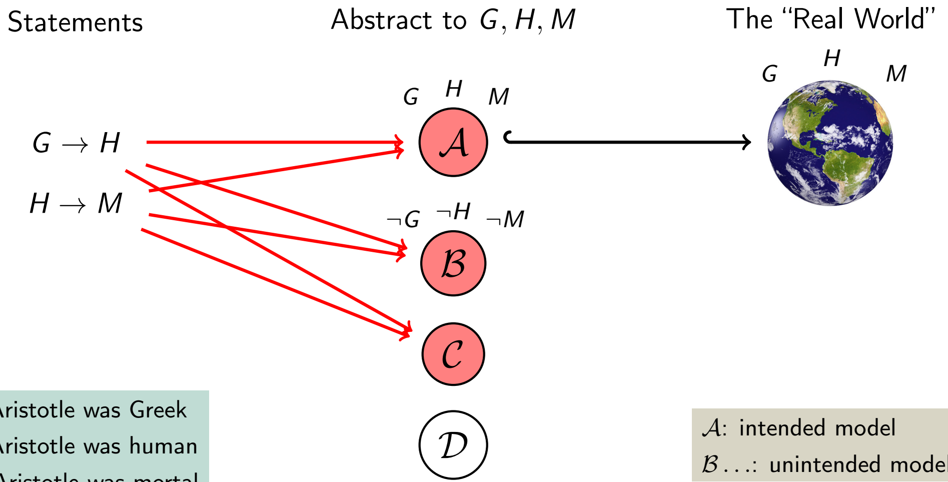




G: Aristotle was Greek
H: Aristotle was human
M: Aristotle was mortal

$\mathcal{A}$: intended model
$\mathcal{B}\ldots$: unintended models

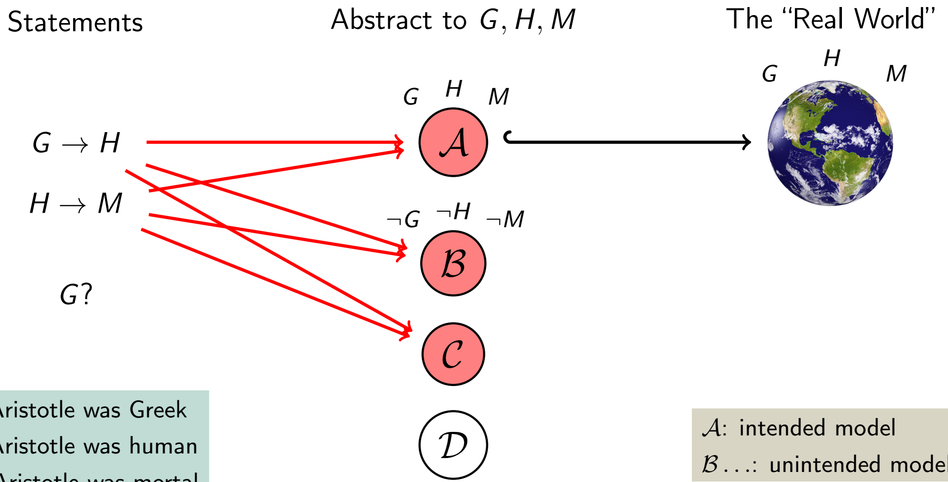
Finding out stuff about the World



G: Aristotle was Greek
H: Aristotle was human
M: Aristotle was mortal

$\mathcal{A}$: intended model
$\mathcal{B}\dots$: unintended models

Finding out stuff about the World

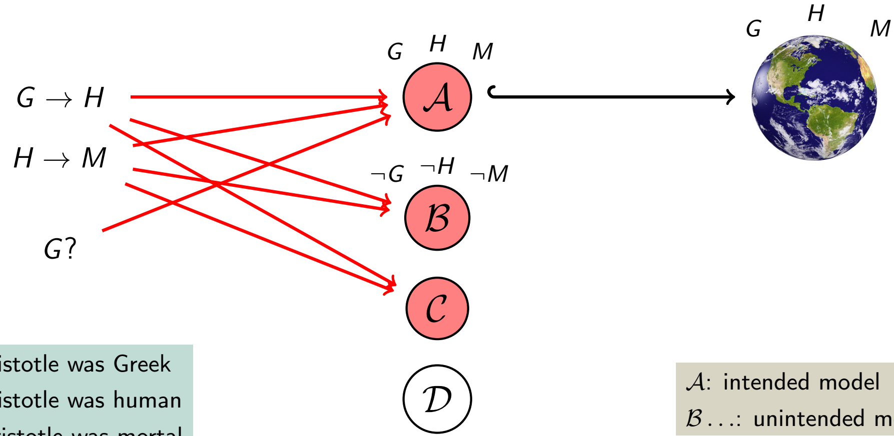


G: Aristotle was Greek
H: Aristotle was human
M: Aristotle was mortal

$\mathcal{A}$: intended model
$\mathcal{B} \ldots$: unintended models

Finding out stuff about the World

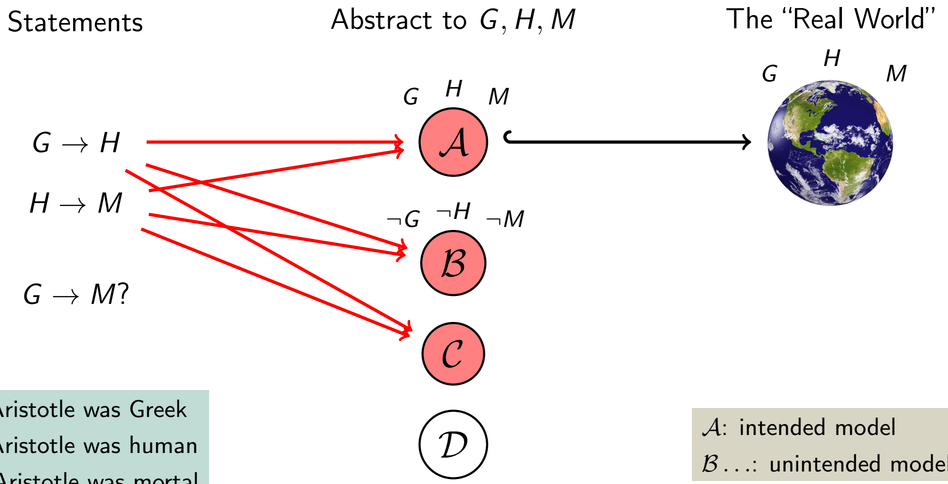
Statements | Abstract to G, H, M | The "Real World"



G: Aristotle was Greek
H: Aristotle was human
M: Aristotle was mortal

$\mathcal{A}$: intended model
$\mathcal{B}\ldots$: unintended models

Finding out stuff about the World

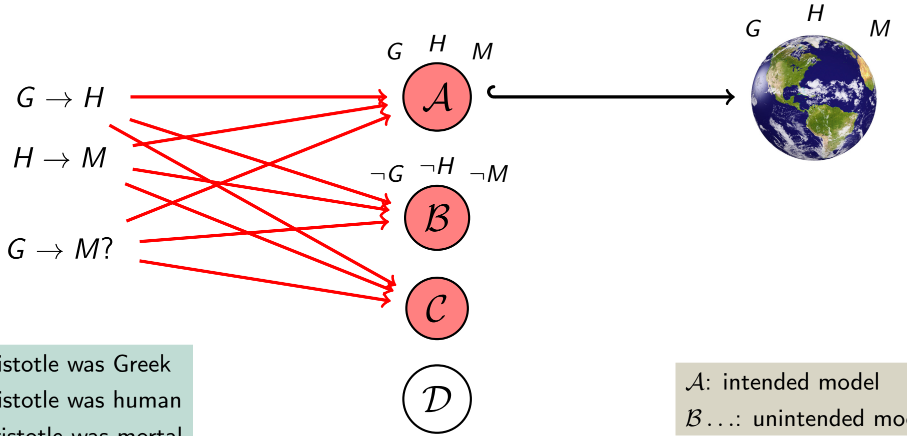


G: Aristotle was Greek
H: Aristotle was human
M: Aristotle was mortal

$\mathcal{A}$: intended model
$\mathcal{B}\ldots$: unintended models

Finding out stuff about the World

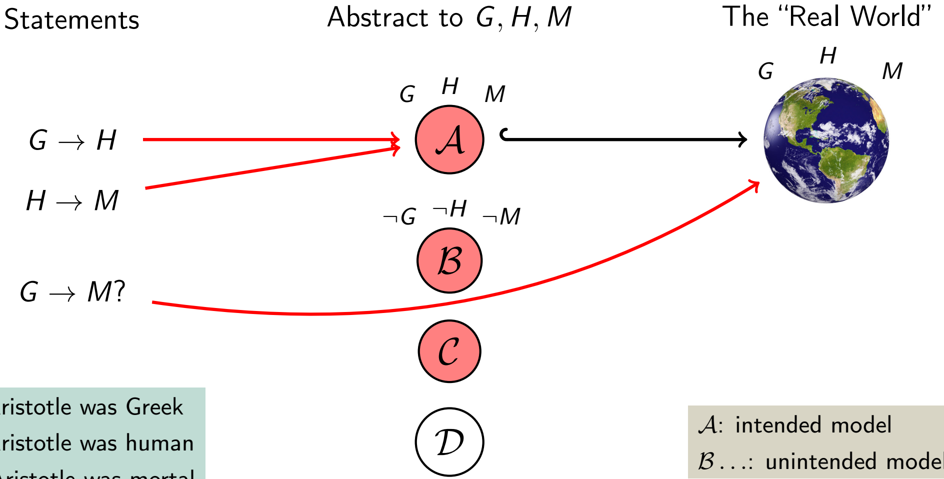
Statements | Abstract to G, H, M | The "Real World"



G: Aristotle was Greek
H: Aristotle was human
M: Aristotle was mortal

$\mathcal{A}$: intended model
$\mathcal{B}\ldots$: unintended models

Finding out stuff about the World



G: Aristotle was Greek
H: Aristotle was human
M: Aristotle was mortal

$\mathcal{A}$: intended model
$\mathcal{B}\ldots$: unintended models

Outline

Propositional Logic: Formulas

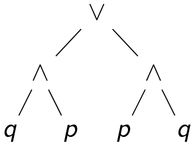
- Formulas are defined "by induction" or "recursively":

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

2 *if* A and B are formulas, *then*
  - $(A \wedge B)$ is also a formula (read: "A and B")
  - $(A \vee B)$ is also a formula (read: "A or B")
  - $\neg A$ is also a formula (read: "not A")

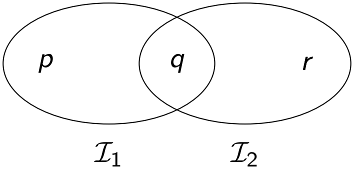
- Nothing else is. Only what rules [1] and [2] say is a formula.
- Examples of formulae: p $\quad (p \wedge \neg r)$ $\quad (q \wedge \neg q)$ $\quad ((p \vee \neg q) \wedge \neg p)$
- Formulas are just a kind of strings until now:
  - no meaning
  - but every formula can be "parsed" uniquely.

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



Interpretations

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



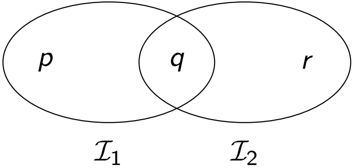
- But in $\mathcal{I}_2 = \{q, r\}$, p is false, but r is true.

Semantic Validity $\models$

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

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

- For instance



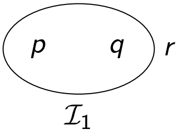
$$\mathcal{I}_1 \models p \qquad \mathcal{I}_2 \not\models p$$

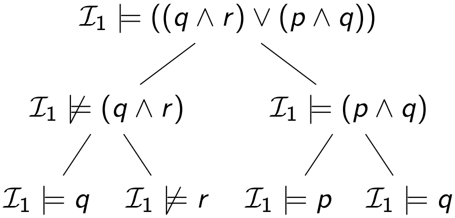
- In other words, for all letters p:

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

Validity of Compound Formulas

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





Truth Table

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

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

Tautologies

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

$$\models A$$

- $(p \lor \neg p)$ is a tautology
- True whatever p means:
  - The sky is blue or the sky is not blue.
  - P.N. will win the 50km in 2016 or P.N. will not win the 50km in 2016.
  - The slithy toves gyre or the slithy toves do not gyre.
- Possible to derive true statements mechanically...
- ...without understanding their meaning!
- ...e.g. using truth tables for small cases.

Entailment

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

  $\mathcal{I} \models B$

  for all interpretations $\mathcal{I}$ with $\mathcal{I} \models A$
- Also: "B is a logical consequence of A"
- Whenever A holds, also B holds
- For instance:

$$p \wedge q \models p$$

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

Question

Given the letters

P – Ola answers none of the questions correctly

Q – Ola fails the exam

Which of the following are tautologies of propositional logic?

1. Q
2. $\neg Q$
3. $P \rightarrow Q$
4. $Q \rightarrow (P \rightarrow Q)$

Outline

Taking the structure of triples into account

Unlike propositions, triples have parts, namely:

Taking the structure of triples into account

Unlike propositions, triples have parts, namely:

- subject
- predicate, and
- object

Taking the structure of triples into account

Unlike propositions, triples have parts, namely:

- subject
- predicate, and
- object

Less abstractly, these may be:

Taking the structure of triples into account

Unlike propositions, triples have parts, namely:

- subject
- predicate, and
- object

Less abstractly, these may be:

- URI references
- literal values, and
- blank nodes

Taking the structure of triples into account

Unlike propositions, triples have parts, namely:

- subject
- predicate, and
- object

Less abstractly, these may be:

- URI references
- literal values, and
- blank nodes

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

On what there is: Resources, Properties, Literals

On what there is: Resources, Properties, Literals

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

On what there is: Resources, Properties, Literals

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

Resources: All things described by RDF are called resources. Resources are identified by URIs

On what there is: Resources, Properties, Literals

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

Resources: All things described by RDF are called resources. Resources are identified by URIs

Properties: A property is a specific aspect, characteristic, attribute or relation used to describe a resource. Properties are also resources, and therefore identified by URIs.

On what there is: Resources, Properties, Literals

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

- **Resources:** All things described by RDF are called resources. Resources are identified by URIs
- **Properties:** A property is a specific aspect, characteristic, attribute or relation used to describe a resource. Properties are also resources, and therefore identified by URIs.
- **Literals:** A literal value is a concrete data item, such as an integer or a string.

On what there is: Resources, Properties, Literals

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

Resources: All things described by RDF are called resources. Resources are identified by URIs

Properties: A property is a specific aspect, characteristic, attribute or relation used to describe a resource. Properties are also resources, and therefore identified by URIs.

Literals: A literal value is a concrete data item, such as an integer or a string. String literals name themselves, i.e.

On what there is: Resources, Properties, Literals

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

Resources: All things described by RDF are called resources. Resources are identified by URIs

Properties: A property is a specific aspect, characteristic, attribute or relation used to describe a resource. Properties are also resources, and therefore identified by URIs.

Literals: A literal value is a concrete data item, such as an integer or a string. String literals name themselves, i.e.

- "Julius Caesar" names the string "Julius Caesar"

On what there is: Resources, Properties, Literals

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

Resources: All things described by RDF are called resources. Resources are identified by URIs

Properties: A property is a specific aspect, characteristic, attribute or relation used to describe a resource. Properties are also resources, and therefore identified by URIs.

Literals: A literal value is a concrete data item, such as an integer or a string. String literals name themselves, i.e.

- "Julius Caesar" names the string "Julius Caesar"
- "42" names the string "42"

On what there is: Resources, Properties, Literals

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

Resources: All things described by RDF are called resources. Resources are identified by URIs

Properties: A property is a specific aspect, characteristic, attribute or relation used to describe a resource. Properties are also resources, and therefore identified by URIs.

Literals: A literal value is a concrete data item, such as an integer or a string. String literals name themselves, i.e.

- "Julius Caesar" names the string "Julius Caesar"
- "42" names the string "42"

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

Restricting RDF/RDFS

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

Restricting RDF/RDFS

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

Restricting RDF/RDFS

- We will simplify things by only looking at certain kinds of RDF graphs.
- No triples “about” properties, classes, etc., except RDFS
- Assume Resources are divided into four disjoint kinds:

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 kinds:
  - *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 kinds:
  - *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 kinds:
  - *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 kinds:
  - *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 kinds:
  - *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 kinds:
  - *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 kinds:
  - *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 kinds:
  - *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 kinds:
  - *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 kinds:
  - *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 kinds:
    - *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 kinds:
  - *Properties* like `foaf:knows`, `dc:title`
  - *Classes* like `foaf:Person`
  - *Built-ins*, a fixed set including `rdf:type`, `rdfs:domain`, etc.
  - *Individuals* (all the rest, "usual" resources)
- All triples have one of the forms:

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

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

- Forget blank nodes and literals for a while!

Short Forms

- Resources and Triples are no longer all alike

Short Forms

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

Short Forms

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

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

Short Forms

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

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

- This is called "Description Logic" (DL) Syntax

Short Forms

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

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

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

Example

- Triples:

Example

- Triples:

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

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

Example

- Triples:

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

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

- DL syntax, without namespaces:

Example

- Triples:

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

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

- DL syntax, without namespaces:

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

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

Interpretations for RDF

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

Interpretations for RDF

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

Interpretations for RDF

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

Interpretations for RDF

- To interpret propositional formulas, we need to know how to interpret
  - Letters
- To interpret the six kinds of triples, we need to know how to interpret
  - *Individual URIs* as real or imagined objects

Interpretations for RDF

- To interpret propositional formulas, we need to know how to interpret
  - Letters
- To interpret the six kinds of triples, we need to know how to interpret
  - *Individual URIs* as real or imagined objects
  - *Class URIs* as sets of such objects

Interpretations for RDF

- To interpret propositional formulas, we need to know how to interpret
  - Letters
- To interpret the six kinds of triples, we need to know how to interpret
  - *Individual URIs* as real or imagined objects
  - *Class URIs* as sets of such objects
  - *Property URIs* as relations between these objects

Interpretations for RDF

- To interpret propositional formulas, we need to know how to interpret
  - Letters
- To interpret the six kinds of triples, we need to know how to interpret
  - *Individual URIs* as real or imagined objects
  - *Class URIs* as sets of such objects
  - *Property URIs* as relations between these objects
- A *DL-interpretation* $\mathcal{I}$ consists of

Interpretations for RDF

- To interpret propositional formulas, we need to know how to interpret
  - Letters
- To interpret the six kinds of triples, we need to know how to interpret
  - *Individual URIs* as real or imagined objects
  - *Class URIs* as sets of such objects
  - *Property URIs* as relations between these objects
- A *DL-interpretation* $\mathcal{I}$ consists of
  - A set $\Delta^{\mathcal{I}}$, called the *domain* (sorry!) of $\mathcal{I}$

Interpretations for RDF

- To interpret propositional formulas, we need to know how to interpret
  - Letters
- To interpret the six kinds of triples, we need to know how to interpret
  - *Individual URIs* as real or imagined objects
  - *Class URIs* as sets of such objects
  - *Property URIs* as relations between these objects
- A *DL-interpretation* $\mathcal{I}$ consists of
  - A set $\Delta^{\mathcal{I}}$, called the *domain* (sorry!) of $\mathcal{I}$
  - For each individual URI i, an element $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$

Interpretations for RDF

- To interpret propositional formulas, we need to know how to interpret
  - Letters
- To interpret the six kinds of triples, we need to know how to interpret
  - *Individual URIs* as real or imagined objects
  - *Class URIs* as sets of such objects
  - *Property URIs* as relations between these objects
- A *DL-interpretation* $\mathcal{I}$ consists of
  - A set $\Delta^{\mathcal{I}}$, called the *domain* (sorry!) of $\mathcal{I}$
  - For each individual URI i, an element $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
  - For each class URI C, a subset $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$

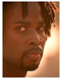
Interpretations for RDF

- To interpret propositional formulas, we need to know how to interpret
  - Letters
- To interpret the six kinds of triples, we need to know how to interpret
  - *Individual URIs* as real or imagined objects
  - *Class URIs* as sets of such objects
  - *Property URIs* as relations between these objects
- A *DL-interpretation* $\mathcal{I}$ consists of
  - A set $\Delta^{\mathcal{I}}$, called the *domain* (sorry!) of $\mathcal{I}$
  - For each individual URI i, an element $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
  - For each class URI C, a subset $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - For each property URI r, a relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

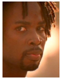
Interpretations for RDF

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

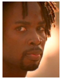
An example "intended" interpretation

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

An example “intended” interpretation

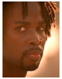
- $\Delta^{\mathcal{I}_1} = \{$ , ,  $\}$
- $\mathit{romeo}^{\mathcal{I}_1} =$  $\mathit{juliet}^{\mathcal{I}_1} =$ 

An example "intended" interpretation

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

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

An example "intended" interpretation

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

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

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

An example "non-intended" interpretation

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

An example “non-intended” interpretation

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

An example “non-intended” interpretation

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

An example "non-intended" interpretation

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

An example "non-intended" interpretation

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

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

An example “non-intended” interpretation

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

- Just because names (URIs) look familiar, they don’t need to denote what we think!
- In fact, there is *no way* of ensuring they denote only what we think!

Validity in Interpretations (RDF)

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

Validity in Interpretations (RDF)

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

Validity in Interpretations (RDF)

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

Validity in Interpretations (RDF)

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

Validity in Interpretations (RDF)

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

Validity in Interpretations (RDF)

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

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

Validity in Interpretations (RDF)

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

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

Validity in Interpretations (RDF)

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

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

    $\mathit{romeo}^{\mathcal{I}_1} =$  $\in \mathit{Person}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1}$

Validity in Interpretations (RDF)

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

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

    $\mathit{romeo}^{\mathcal{I}_1} =$  $\in \mathit{Person}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1}$
  - $\mathcal{I}_2 \not\models \mathit{loves}(\mathit{juliet}, \mathit{romeo})$ because

Validity in Interpretations (RDF)

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models r(i_1, i_2)$ iff $\langle i_1^{\mathcal{I}}, i_2^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$
- $\mathcal{I} \models C(i)$ iff $i^{\mathcal{I}} \in C^{\mathcal{I}}$
- Examples:
  - $\mathcal{I}_1 \models \mathit{loves}(\mathit{juliet}, \mathit{romeo})$ because
    $\Big\langle$  ,  $\Big\rangle \in \mathit{loves}^{\mathcal{I}_1} = \Big\{ \Big\langle$  ,  $\Big\rangle , \Big\langle$  ,  $\Big\rangle \Big\}$
  - $\mathcal{I}_1 \models \mathit{Person}(\mathit{romeo})$ because
    $\mathit{romeo}^{\mathcal{I}_1} =$  $\in \mathit{Person}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1}$
  - $\mathcal{I}_2 \not\models \mathit{loves}(\mathit{juliet}, \mathit{romeo})$ because
    $\mathit{loves}^{\mathcal{I}_2} = <$ and $\mathit{juliet}^{\mathcal{I}_2} = 32 \not< \mathit{romeo}^{\mathcal{I}_2} = 17$

Validity in Interpretations (RDF)

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models r(i_1, i_2)$ iff $\langle i_1^{\mathcal{I}}, i_2^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$
- $\mathcal{I} \models C(i)$ iff $i^{\mathcal{I}} \in C^{\mathcal{I}}$
- Examples:
  - $\mathcal{I}_1 \models loves(juliet, romeo)$ because
    $\Big\langle$ ,  $\Big\rangle \in loves^{\mathcal{I}_1} = \Big\{ \Big\langle$ ,  $\Big\rangle , \Big\langle$ ,  $\Big\rangle \Big\}$
  - $\mathcal{I}_1 \models Person(romeo)$ because
    $romeo^{\mathcal{I}_1} =$  $\in Person^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1}$
  - $\mathcal{I}_2 \not\models loves(juliet, romeo)$ because
    $loves^{\mathcal{I}_2} = <$ and $juliet^{\mathcal{I}_2} = 32 \not< romeo^{\mathcal{I}_2} = 17$
  - $\mathcal{I}_2 \not\models Person(romeo)$ because

Validity in Interpretations (RDF)

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models r(i_1, i_2)$ iff $\langle i_1^{\mathcal{I}}, i_2^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$
- $\mathcal{I} \models C(i)$ iff $i^{\mathcal{I}} \in C^{\mathcal{I}}$
- Examples:
  - $\mathcal{I}_1 \models loves(juliet, romeo)$ because
    $\langle$  ,  $\rangle \in loves^{\mathcal{I}_1} = \{ \langle$  ,  $\rangle , \langle$  ,  $\rangle \}$
  - $\mathcal{I}_1 \models Person(romeo)$ because
    $romeo^{\mathcal{I}_1} =$  $\in Person^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1}$
  - $\mathcal{I}_2 \not\models loves(juliet, romeo)$ because
    $loves^{\mathcal{I}_2} = <$ and $juliet^{\mathcal{I}_2} = 32 \not< romeo^{\mathcal{I}_2} = 17$
  - $\mathcal{I}_2 \not\models Person(romeo)$ because
  - $romeo^{\mathcal{I}_2} = 17 \notin Person^{\mathcal{I}_2} = \{2, 4, 6, 8, 10, \ldots\}$

Validity in Interpretations, cont. (RDFS)

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

Validity in Interpretations, cont. (RDFS)

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

Validity in Interpretations, cont. (RDFS)

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models C \sqsubseteq D$ 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, cont. (RDFS)

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I} \models r \sqsubseteq s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
- $\mathcal{I} \models \mathrm{dom}(r, C)$ iff for all $\langle x, y \rangle \in r^{\mathcal{I}}$, we have $x \in C^{\mathcal{I}}$

Validity in Interpretations, cont. (RDFS)

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I} \models r \sqsubseteq s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
- $\mathcal{I} \models \mathrm{dom}(r, C)$ iff for all $\langle x, y \rangle \in r^{\mathcal{I}}$, we have $x \in C^{\mathcal{I}}$
- $\mathcal{I} \models \mathrm{rg}(r, C)$ iff for all $\langle x, y \rangle \in r^{\mathcal{I}}$, we have $y \in C^{\mathcal{I}}$

Validity in Interpretations, cont. (RDFS)

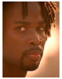
- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I} \models r \sqsubseteq s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
- $\mathcal{I} \models \mathrm{dom}(r, C)$ iff for all $\langle x, y \rangle \in r^{\mathcal{I}}$, we have $x \in C^{\mathcal{I}}$
- $\mathcal{I} \models \mathrm{rg}(r, C)$ iff for all $\langle x, y \rangle \in r^{\mathcal{I}}$, we have $y \in C^{\mathcal{I}}$
- Examples:

Validity in Interpretations, cont. (RDFS)

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I} \models r \sqsubseteq s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
- $\mathcal{I} \models \text{dom}(r, C)$ iff for all $\langle x, y \rangle \in r^{\mathcal{I}}$, we have $x \in C^{\mathcal{I}}$
- $\mathcal{I} \models \text{rg}(r, C)$ iff for all $\langle x, y \rangle \in r^{\mathcal{I}}$, we have $y \in C^{\mathcal{I}}$
- Examples:
  - $\mathcal{I}_1 \models \mathit{Lover} \sqsubseteq \mathit{Person}$ because

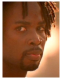
Validity in Interpretations, cont. (RDFS)

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models C \sqsubseteq D$ 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 for all $\langle x, y \rangle \in r^{\mathcal{I}}$, we have $x \in C^{\mathcal{I}}$
- $\mathcal{I} \models \mathrm{rg}(r, C)$ iff for all $\langle x, y \rangle \in r^{\mathcal{I}}$, we have $y \in C^{\mathcal{I}}$
- Examples:
  - $\mathcal{I}_1 \models \mathit{Lover} \sqsubseteq \mathit{Person}$ because

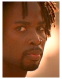
$\mathit{Lover}^{\mathcal{I}_1} = \{$ ,  $\} \subseteq \mathit{Person}^{\mathcal{I}_1} = \{$ , ,  $\}$

Validity in Interpretations, cont. (RDFS)

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I} \models r \sqsubseteq s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
- $\mathcal{I} \models \mathrm{dom}(r, C)$ iff for all $\langle x, y \rangle \in r^{\mathcal{I}}$, we have $x \in C^{\mathcal{I}}$
- $\mathcal{I} \models \mathrm{rg}(r, C)$ iff for all $\langle x, y \rangle \in r^{\mathcal{I}}$, we have $y \in C^{\mathcal{I}}$
- Examples:
  - $\mathcal{I}_1 \models \mathit{Lover} \sqsubseteq \mathit{Person}$ because

    $\mathit{Lover}^{\mathcal{I}_1} = \{$ ,  $\} \subseteq \mathit{Person}^{\mathcal{I}_1} = \{$ , ,  $\}$
  - $\mathcal{I}_2 \not\models \mathit{Lover} \sqsubseteq \mathit{Person}$ because

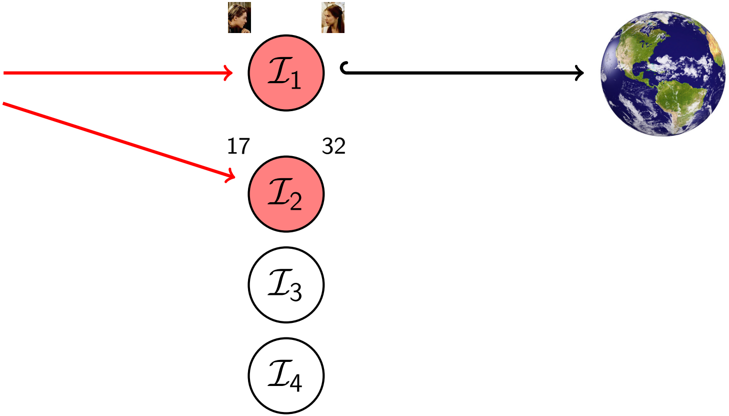
Validity in Interpretations, cont. (RDFS)

- Given an interpretation $\mathcal{I}$, define $\models$ as follows:
- $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I} \models r \sqsubseteq s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
- $\mathcal{I} \models \mathrm{dom}(r, C)$ iff for all $\langle x, y \rangle \in r^{\mathcal{I}}$, we have $x \in C^{\mathcal{I}}$
- $\mathcal{I} \models \mathrm{rg}(r, C)$ iff for all $\langle x, y \rangle \in r^{\mathcal{I}}$, we have $y \in C^{\mathcal{I}}$
- Examples:
  - $\mathcal{I}_1 \models \mathit{Lover} \sqsubseteq \mathit{Person}$ because
    $\mathit{Lover}^{\mathcal{I}_1} = \{$ ,  $\} \subseteq \mathit{Person}^{\mathcal{I}_1} = \{$ , ,  $\}$
  - $\mathcal{I}_2 \not\models \mathit{Lover} \sqsubseteq \mathit{Person}$ because
    $\mathit{Lover}^{\mathcal{I}_2} = \mathbb{N}$ and $\mathit{Person}^{\mathcal{I}_2} = \{2, 4, 6, 8, 10, \ldots\}$

Finding out stuff about Romeo and Juliet

Statements | **Interpretations** | **The "Real World"**

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



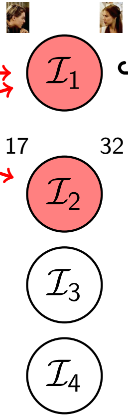
Finding out stuff about Romeo and Juliet

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

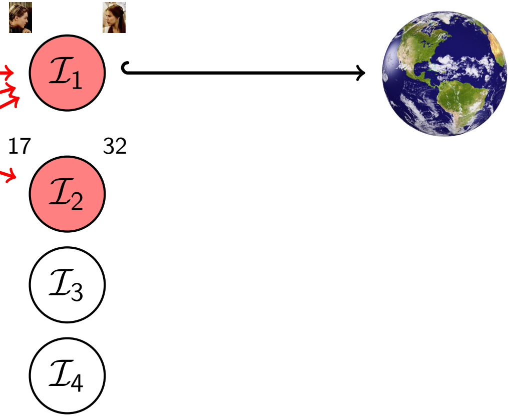
$\mathit{loves}(\mathit{romeo}, \mathit{juliet})$
$\mathit{Lady}(\mathit{juliet})$
$\mathit{Lady} \sqsubseteq \mathit{Person}$
$\mathit{loves} \sqsubseteq \mathit{knows}$
$\mathbf{dom}(\mathit{loves}, \mathit{Lover})$
$\mathbf{rg}(\mathit{loves}, \mathit{Beloved})$

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

$\mathit{Lover} \sqsubseteq \mathit{Person}$

Interpretations

The "Real World"



Example: Range/Domain semantics

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

because. . .

Example: Range/Domain semantics

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

because...

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

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

Example: Range/Domain semantics

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

because...

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

and for any x and y with

$$\langle x, y \rangle \in \mathit{knows}^{\mathcal{I}_2}, \quad \text{i.e.} \quad x \leq y,$$

Example: Range/Domain semantics

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

because...

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

and for any x and y with

$$\langle x, y\rangle \in \mathit{knows}^{\mathcal{I}_2}, \quad \text{i.e.} \quad x \leq y,$$

we also have

$$x \in \mathbb{N} \quad \text{i.e.} \quad x \in \mathit{Beloved}^{\mathcal{I}_2}$$

Interpretation of Sets of Triples

- Given an interpretation $\mathcal{I}$

Interpretation of Sets of Triples

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

Interpretation of Sets of Triples

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

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

Interpretation of Sets of Triples

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

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

- iff $\mathcal{I} \models A$ for all $A \in \mathcal{A}$.

Interpretation of Sets of Triples

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

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

- iff $\mathcal{I} \models A$ for all $A \in \mathcal{A}$.
- Then $\mathcal{I}$ is also called a model of $\mathcal{A}$.

Interpretation of Sets of Triples

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

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

- iff $\mathcal{I} \models A$ for all $A \in \mathcal{A}$.
- Then $\mathcal{I}$ is also called a model of $\mathcal{A}$.
- Examples:

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

Interpretation of Sets of Triples

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

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

- iff $\mathcal{I} \models A$ for all $A \in \mathcal{A}$.
- Then $\mathcal{I}$ is also called a model of $\mathcal{A}$.
- Examples:

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

- Then $\mathcal{I}_1 \models \mathcal{A}$ and $\mathcal{I}_2 \models \mathcal{A}$

Entailment

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

Entailment

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

Entailment

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

Entailment

- Given a set of triples $\mathcal{A}$ (any of the six kinds)
- And a further triple T (also any kind)
- T is entailed by $\mathcal{A}$, written $\mathcal{A} \models T$
- iff

Entailment

- Given a set of triples $\mathcal{A}$ (any of the six kinds)
- And a further triple T (also any kind)
- T is entailed by $\mathcal{A}$, written $\mathcal{A} \models T$
- iff
  - For any interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$

Entailment

- Given a set of triples $\mathcal{A}$ (any of the six kinds)
- And a further triple T (also any kind)
- T is entailed by $\mathcal{A}$, written $\mathcal{A} \models T$
- iff
  - For any interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$
  - $\mathcal{I} \models T$.

Entailment

- Given a set of triples $\mathcal{A}$ (any of the six kinds)
- And a further triple T (also any kind)
- T is entailed by $\mathcal{A}$, written $\mathcal{A} \models T$
- iff
    - For any interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$
    - $\mathcal{I} \models T$.
- $\mathcal{A} \models \mathcal{B}$ iff $\mathcal{I} \models \mathcal{B}$ for all $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$

Entailment

- Given a set of triples $\mathcal{A}$ (any of the six kinds)
- And a further triple T (also any kind)
- T is entailed by $\mathcal{A}$, written $\mathcal{A} \models T$
- iff
    - For any interpretation $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$
    - $\mathcal{I} \models T$.
- $\mathcal{A} \models \mathcal{B}$ iff $\mathcal{I} \models \mathcal{B}$ for all $\mathcal{I}$ with $\mathcal{I} \models \mathcal{A}$
- Example:

Entailment

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

Entailment

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

Entailment

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

Entailment

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

Entailment

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

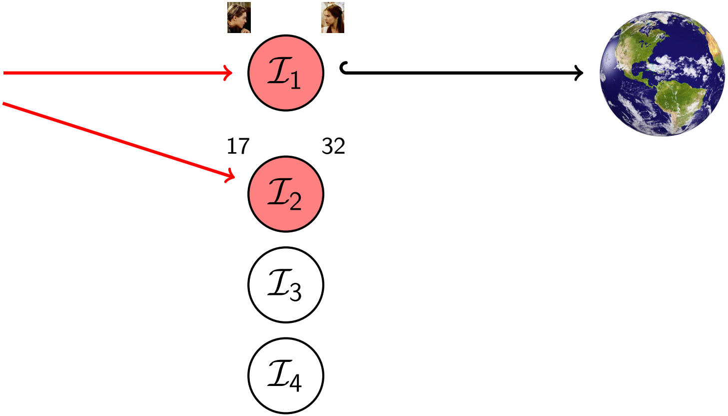
Finding out stuff about Romeo and Juliet

Statements

Interpretations

The “Real World”

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



Finding out stuff about Romeo and Juliet

Statements

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

Person(*juliet*)

Interpretations


$\mathcal{I}_1$

17 32

$\mathcal{I}_2$

$\mathcal{I}_3$

$\mathcal{I}_4$

The "Real World"

Countermodels

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

Countermodel Example

- $\mathcal{A}$ as before:

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

Countermodel Example

- $\mathcal{A}$ as before:

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

- Does $\mathcal{A} \models \mathit{Lover} \sqsubseteq \mathit{Beloved}$?

Countermodel Example

- $\mathcal{A}$ as before:

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

- Does $\mathcal{A} \models \mathit{Lover} \sqsubseteq \mathit{Beloved}$?
- Holds in $\mathcal{I}_1$ and $\mathcal{I}_2$.

Countermodel Example

- $\mathcal{A}$ as before:

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

- Does $\mathcal{A} \models \mathit{Lover} \sqsubseteq \mathit{Beloved}$?
- Holds in $\mathcal{I}_1$ and $\mathcal{I}_2$.
- Try to find an interpretaion with $\Delta^{\mathcal{I}} = \{a, b\}$, $a \neq b$.

Countermodel Example

- $\mathcal{A}$ as before:

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

- Does $\mathcal{A} \models Lover \sqsubseteq Beloved$?
- Holds in $\mathcal{I}_1$ and $\mathcal{I}_2$.
- Try to find an interpretaion with $\Delta^{\mathcal{I}} = \{a, b\}$, $a \neq b$.
- Interpret $romeo^{\mathcal{I}} = a$ and $juliet^{\mathcal{I}} = b$

Countermodel Example

- $\mathcal{A}$ as before:

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

- Does $\mathcal{A} \models Lover \sqsubseteq Beloved$?
- Holds in $\mathcal{I}_1$ and $\mathcal{I}_2$.
- Try to find an interpretaion with $\Delta^{\mathcal{I}} = \{a, b\}$, $a \neq b$.
- Interpret $romeo^{\mathcal{I}} = a$ and $juliet^{\mathcal{I}} = b$
- Then $\langle a, b\rangle \in loves^{\mathcal{I}}$, $a \in Lover^{\mathcal{I}}$, $b \in Beloved^{\mathcal{I}}$.

Countermodel Example

- $\mathcal{A}$ as before:

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

- Does $\mathcal{A} \models Lover \sqsubseteq Beloved$?
- Holds in $\mathcal{I}_1$ and $\mathcal{I}_2$.
- Try to find an interpretaion with $\Delta^{\mathcal{I}} = \{a, b\}$, $a \neq b$.
- Interpret $romeo^{\mathcal{I}} = a$ and $juliet^{\mathcal{I}} = b$
- Then $\langle a, b\rangle \in loves^{\mathcal{I}}$, $a \in Lover^{\mathcal{I}}$, $b \in Beloved^{\mathcal{I}}$.
- Choose

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

Countermodel Example

- $\mathcal{A}$ as before:

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

- Does $\mathcal{A} \models Lover \sqsubseteq Beloved$?
- Holds in $\mathcal{I}_1$ and $\mathcal{I}_2$.
- Try to find an interpretaion with $\Delta^{\mathcal{I}} = \{a, b\}$, $a \neq b$.
- Interpret $romeo^{\mathcal{I}} = a$ and $juliet^{\mathcal{I}} = b$
- Then $\langle a, b\rangle \in loves^{\mathcal{I}}$, $a \in Lover^{\mathcal{I}}$, $b \in Beloved^{\mathcal{I}}$.
- Choose

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

- With $Lover^{\mathcal{I}} = \{a\}$ and $Beloved^{\mathcal{I}} = \{b\}$, to complete the counter-model while satisfying $\mathcal{I} \models \mathcal{A}$.

Countermodel Example

- $\mathcal{A}$ as before:

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

- Does $\mathcal{A} \models Lover \sqsubseteq Beloved$?
- Holds in $\mathcal{I}_1$ and $\mathcal{I}_2$.
- Try to find an interpretaion with $\Delta^{\mathcal{I}} = \{a, b\}$, $a \neq b$.
- Interpret $romeo^{\mathcal{I}} = a$ and $juliet^{\mathcal{I}} = b$
- Then $\langle a, b\rangle \in loves^{\mathcal{I}}$, $a \in Lover^{\mathcal{I}}$, $b \in Beloved^{\mathcal{I}}$.
- Choose

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

- With $Lover^{\mathcal{I}} = \{a\}$ and $Beloved^{\mathcal{I}} = \{b\}$, to complete the counter-model while satisfying $\mathcal{I} \models \mathcal{A}$.
- $\mathcal{I} \not\models Lover \sqsubseteq Beloved$!

Countermodels about Romeo and Juliet

Statements

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

Interpretations

$\mathcal{I}_1$

17 32

$\mathcal{I}_2$

a b

$\mathcal{I}_3$

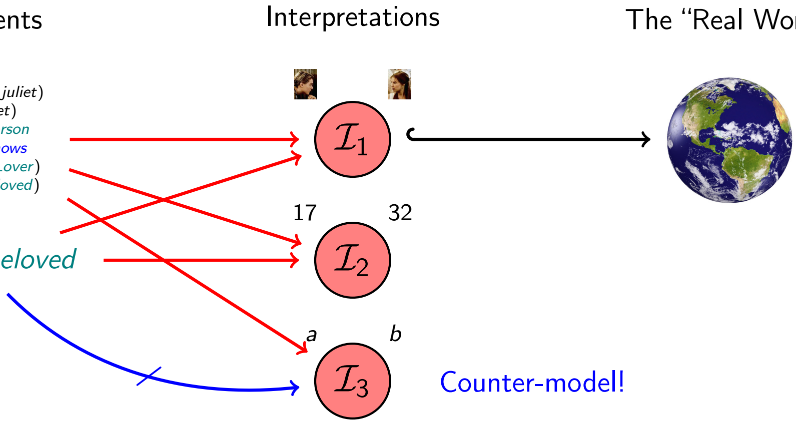
The "Real World"

Countermodels about Romeo and Juliet

Statements

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

$Lover \sqsubseteq Beloved$



Take aways

Take aways

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

Take aways

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

Take aways

1. Model-theoretic semantics yields an unambigous notion of entailment,
2. which is necessary in order to liberate data from applications.
3. Shown today: A simplified semantics for parts of RDF
    1. Only RDF/RDFS vocabulary to talk “about” predicates and classes
    2. Literals and blank nodes next time

Take aways

1. Model-theoretic semantics yields an unambigous notion of entailment,
2. which is necessary in order to liberate data from applications.
3. Shown today: A simplified semantics for parts of RDF
   1. Only RDF/RDFS vocabulary to talk "about" predicates and classes
   2. Literals and blank nodes next time

Supplementary reading on RDF and RDFS semantics:

- `http://www.w3.org/TR/rdf-mt/`
- Section 3.2 in Foundations of SW Technologies