

Semantic Technologies: Introduction

Bouvet Fagdag

Martin Giese (modified by Dag Hovland)

17 Feb 2023

Who wrote this and why is it in English?

- ▶ Martin Giese, Professor at University of Oslo, Norway, Teaching a course on Semantic Technologies since 2010
- ▶ Dag Hovland stole it and modified for Fagdag

Tutorial Structure

12:00 Introduction

- ▶ General intro, 30 mins
- ▶ Break

13:00 RDF – representing information

- ▶ Lecture, 30 mins
- ▶ Break, solve exercises, 15 mins
- ▶ Solutions to exercises, 10 mins

14:00 OWL – modeling

- ▶ 30 mins lecture
- ▶ Break, solve exercises, 15 mins
- ▶ Solutions to exercises, 10 mins

15:00 SPARQL – querying

- ▶ Intro 15 minutes
- ▶ Exercises, 30 minutes

Used Software, Tutorial Web Page

- ▶ You will need:
 - ▶ A text editor for writing RDF
 - ▶ A web browser to play with some web services
 - ▶ Protégé for modelling

Just Another Database?

- ▶ Semantic Technologies are *Technologies!*

Just Another Database?

- ▶ Semantic Technologies are *Technologies!*
- ▶ Something that looks like a “data format,” called RDF...

Just Another Database?

- ▶ Semantic Technologies are *Technologies!*
- ▶ Something that looks like a “data format,” called RDF...
- ▶ ...where data looks like a graph, instead of a table

Just Another Database?

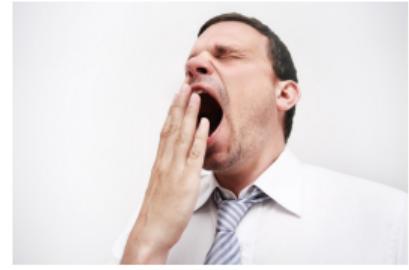
- ▶ Semantic Technologies are *Technologies!*
- ▶ Something that looks like a “data format,” called RDF...
- ▶ ...where data looks like a graph, instead of a table
- ▶ “Triple stores” where you can store those graphs

Just Another Database?

- ▶ Semantic Technologies are *Technologies!*
- ▶ Something that looks like a “data format,” called RDF...
- ▶ ...where data looks like a graph, instead of a table
- ▶ “Triple stores” where you can store those graphs
- ▶ A query language to query those graphs

Just Another Database?

- ▶ Semantic Technologies are *Technologies!*
- ▶ Something that looks like a “data format,” called RDF...
- ▶ ...where data looks like a graph, instead of a table
- ▶ “Triple stores” where you can store those graphs
- ▶ A query language to query those graphs
- ▶ Sounds like databases reinvented :-/



Just Another Database?

- ▶ Semantic Technologies are *Technologies!*
- ▶ Something that looks like a “data format,” called RDF...
- ▶ ...where data looks like a graph, instead of a table
- ▶ “Triple stores” where you can store those graphs
- ▶ A query language to query those graphs
- ▶ Sounds like databases reinvented :-/
- ▶ BONUS: a bunch of W3C standardisation mumbojumbo.



Just Another Database?

- ▶ Semantic Technologies are *Technologies!*
- ▶ Something that looks like a “data format,” called RDF...
- ▶ ...where data looks like a graph, instead of a table
- ▶ “Triple stores” where you can store those graphs
- ▶ A query language to query those graphs
- ▶ Sounds like databases reinvented :-/
- ▶ BONUS: a bunch of W3C standardisation mumbojumbo.
 - ▶ HTTP, XML, XSD, WGs, “recommendations”



Just Another Database?

- ▶ Semantic Technologies are *Technologies!*
- ▶ Something that looks like a “data format,” called RDF...
- ▶ ...where data looks like a graph, instead of a table
- ▶ “Triple stores” where you can store those graphs
- ▶ A query language to query those graphs
- ▶ Sounds like databases reinvented :-/
- ▶ BONUS: a bunch of W3C standardisation mumbojumbo.
 - ▶ HTTP, XML, XSD, WGs, “recommendations”
- ▶ BONUS: a bunch of crackpots!



Just Another Database?

- ▶ Semantic Technologies are *Technologies!*
- ▶ Something that looks like a “data format,” called RDF...
- ▶ ...where data looks like a graph, instead of a table
- ▶ “Triple stores” where you can store those graphs
- ▶ A query language to query those graphs
- ▶ Sounds like databases reinvented :-/
- ▶ BONUS: a bunch of W3C standardisation mumbojumbo.
 - ▶ HTTP, XML, XSD, WGs, “recommendations”
- ▶ BONUS: a bunch of crackpots!
 - ▶ Logicians!? Philosophers!!??



Just Another Database – from Hell!?

- ▶ Semantic Technologies are *Technologies!*
- ▶ Something that looks like a “data format,” called RDF...
- ▶ ...where data looks like a graph, instead of a table
- ▶ “Triple stores” where you can store those graphs
- ▶ A query language to query those graphs
- ▶ Sounds like databases reinvented :-/
- ▶ BONUS: a bunch of W3C standardisation mumbojumbo.
 - ▶ HTTP, XML, XSD, WGs, “recommendations”
- ▶ BONUS: a bunch of crackpots!
 - ▶ Logicians!? Philosophers!!??



It's not about the graphs! It's not about the data!

- Semantic Technologies are all about **Knowledge Representation**



It's not about the graphs! It's not about the data!

- ▶ Semantic Technologies are all about **Knowledge Representation**
 - ▶ I'll tell you more about Knowledge



It's not about the graphs! It's not about the data!

- ▶ Semantic Technologies are all about **Knowledge Representation**
 - ▶ I'll tell you more about Knowledge
 - ▶ I'll tell you more about representing it



It's not about the graphs! It's not about the data!

- ▶ Semantic Technologies are all about **Knowledge Representation**
 - ▶ I'll tell you more about Knowledge
 - ▶ I'll tell you more about representing it
- ▶ *Graphs* happen to be a nice way of representing knowledge



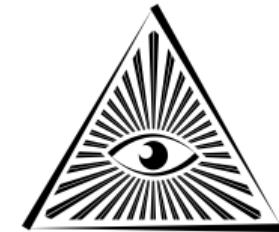
It's not about the graphs! It's not about the data!

- ▶ Semantic Technologies are all about **Knowledge Representation**
 - ▶ I'll tell you more about Knowledge
 - ▶ I'll tell you more about representing it
- ▶ *Graphs* happen to be a nice way of representing knowledge
 - ▶ part of the reason for the success of graph DBs?



It's not about the graphs! It's not about the data!

- ▶ Semantic Technologies are all about **Knowledge Representation**
 - ▶ I'll tell you more about Knowledge
 - ▶ I'll tell you more about representing it
- ▶ *Graphs* happen to be a nice way of representing knowledge
 - ▶ part of the reason for the success of graph DBs?
- ▶ Semantic Technologies are all about **Data Integration** and **Interoperability**



It's not about the graphs! It's not about the data!

- ▶ Semantic Technologies are all about **Knowledge Representation**
 - ▶ I'll tell you more about Knowledge
 - ▶ I'll tell you more about representing it
- ▶ *Graphs* happen to be a nice way of representing knowledge
 - ▶ part of the reason for the success of graph DBs?
- ▶ Semantic Technologies are all about **Data Integration** and **Interoperability**
 - ▶ Global identifiers are good (URIs?)



It's not about the graphs! It's not about the data!

- ▶ Semantic Technologies are all about **Knowledge Representation**
 - ▶ I'll tell you more about Knowledge
 - ▶ I'll tell you more about representing it
- ▶ *Graphs* happen to be a nice way of representing knowledge
 - ▶ part of the reason for the success of graph DBs?
- ▶ Semantic Technologies are all about **Data Integration** and **Interoperability**
 - ▶ Global identifiers are good (URIs?)
 - ▶ Need to address and move data sets (HTTP?)



It's not about the graphs! It's not about the data!

- ▶ Semantic Technologies are all about **Knowledge Representation**
 - ▶ I'll tell you more about Knowledge
 - ▶ I'll tell you more about representing it
- ▶ *Graphs* happen to be a nice way of representing knowledge
 - ▶ part of the reason for the success of graph DBs?
- ▶ Semantic Technologies are all about **Data Integration** and **Interoperability**
 - ▶ Global identifiers are good (URIs?)
 - ▶ Need to address and move data sets (HTTP?)
 - ▶ Need agreed standards (W3C?)



It's not about the graphs! It's not about the data!

- ▶ Semantic Technologies are all about **Knowledge Representation**
 - ▶ I'll tell you more about Knowledge
 - ▶ I'll tell you more about representing it
- ▶ *Graphs* happen to be a nice way of representing knowledge
 - ▶ part of the reason for the success of graph DBs?
- ▶ Semantic Technologies are all about **Data Integration** and **Interoperability**
 - ▶ Global identifiers are good (URIs?)
 - ▶ Need to address and move data sets (HTTP?)
 - ▶ Need agreed standards (W3C?)
- ▶ Semantic Technologies **Liberate Business Logic from Applications**



It's not about the graphs! It's not about the data!

- ▶ Semantic Technologies are all about **Knowledge Representation**
 - ▶ I'll tell you more about Knowledge
 - ▶ I'll tell you more about representing it
- ▶ *Graphs* happen to be a nice way of representing knowledge
 - ▶ part of the reason for the success of graph DBs?
- ▶ Semantic Technologies are all about **Data Integration** and **Interoperability**
 - ▶ Global identifiers are good (URIs?)
 - ▶ Need to address and move data sets (HTTP?)
 - ▶ Need agreed standards (W3C?)
- ▶ Semantic Technologies **Liberate Business Logic from Applications**
 - ▶ Need sw tools that works with Knowledge Representations (logic?)



It's not about the graphs! It's not about the data!

- ▶ Semantic Technologies are all about **Knowledge Representation**
 - ▶ I'll tell you more about Knowledge
 - ▶ I'll tell you more about representing it
- ▶ *Graphs* happen to be a nice way of representing knowledge
 - ▶ part of the reason for the success of graph DBs?
- ▶ Semantic Technologies are all about **Data Integration** and **Interoperability**
 - ▶ Global identifiers are good (URIs?)
 - ▶ Need to address and move data sets (HTTP?)
 - ▶ Need agreed standards (W3C?)
- ▶ Semantic Technologies **Liberate Business Logic from Applications**
 - ▶ Need sw tools that works with Knowledge Representations (logic?)
 - ▶ Need to agree on what it all *means* (philosophy?)



Data integration, engineering and knowledge

- ▶ Data usually lacks the context in which it was made

Data integration, engineering and knowledge

- ▶ Data usually lacks the context in which it was made
 - ▶ what column names mean

Data integration, engineering and knowledge

- ▶ Data usually lacks the context in which it was made
 - ▶ what column names mean
 - ▶ conditions for validity

Data integration, engineering and knowledge

- ▶ Data usually lacks the context in which it was made
 - ▶ what column names mean
 - ▶ conditions for validity
- ▶ Knowledge is the context

Data integration, engineering and knowledge

- ▶ Data usually lacks the context in which it was made
 - ▶ what column names mean
 - ▶ conditions for validity
- ▶ Knowledge is the context
 - ▶ How is this measurement made in this stage

Data integration, engineering and knowledge

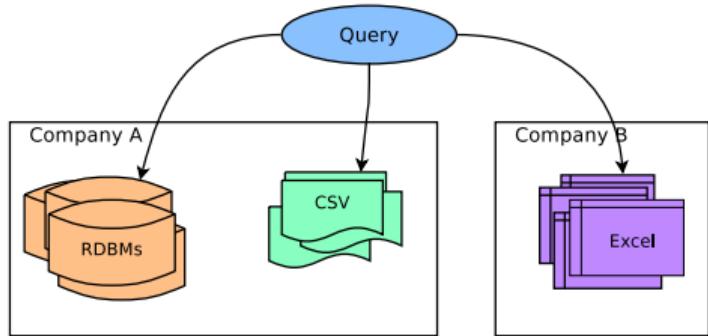
- ▶ Data usually lacks the context in which it was made
 - ▶ what column names mean
 - ▶ conditions for validity
- ▶ Knowledge is the context
 - ▶ How is this measurement made in this stage
 - ▶ What does this abbreviation mean in this filetype?

Data integration, engineering and knowledge

- ▶ Data usually lacks the context in which it was made
 - ▶ what column names mean
 - ▶ conditions for validity
- ▶ Knowledge is the context
 - ▶ How is this measurement made in this stage
 - ▶ What does this abbreviation mean in this filetype?
- ▶ Explicit formulation of context is the promise of semantic technology

Data integration, engineering and knowledge

- ▶ Data usually lacks the context in which it was made
 - ▶ what column names mean
 - ▶ conditions for validity
- ▶ Knowledge is the context
 - ▶ How is this measurement made in this stage
 - ▶ What does this abbreviation mean in this filetype?
- ▶ Explicit formulation of context is the promise of semantic technology



Data vs. Information

Data

MyBurgers	17	06-2018
-----------	----	---------



Information

“*MyBurgers sold 17 cheeseburgers in June 2018.*”

- ▶ (assume we understand all words)
- ▶ Sentence or statement
- ▶ That is, it is *true* or *false*
- ▶ a.k.a. Knowledge

The Many Shapes of Data

	...	Oct	Nov	Dec	...	Species	Month	Nr. sold
...	
Cats	...	15	33	18	...	Cat	Dec	18
Dogs	...	12	17	88	...	Dog	Nov	17
Skunks	...	1	0	0	...	Skunk	Oct	1
...			

The Many Shapes of Data

	...	Oct	Nov	Dec	...	Species	Month	Nr. sold
...
Cats	...	15	33	18	...	Cat	Dec	18
Dogs	...	12	17	88	...	Dog	Nov	17
Skunks	...	1	0	0	...	Skunk	Oct	1
...

- ▶ Data: lots of numbers

The Many Shapes of Data

	...	Oct	Nov	Dec	...	Species	Month	Nr. sold
...	
Cats	...	15	33	18	...	Cat	Dec	18
Dogs	...	12	17	88	...	Dog	Nov	17
Skunks	...	1	0	0	...	Skunk	Oct	1
...			

- ▶ Data: lots of numbers
- ▶ Information: “17 dogs sold in November.”

The Many Shapes of Data

	...	Oct	Nov	Dec	...	Species	Month	Nr. sold
...	
Cats	...	15	33	18	...	Cat	Dec	18
Dogs	...	12	17	88	...	Dog	Nov	17
Skunks	...	1	0	0	...	Skunk	Oct	1
...	

- ▶ Data: lots of numbers
- ▶ Information: “17 dogs sold in November.”
- ▶ Could be less obvious: 17 lines with Dog/November

Joining Data and Information

	...	Oct	Nov	Dec	...	Product	Year	# sold
...	
Cats	...	15	33	18	...	Hamburger	2013	13
Dogs	...	12	17	88	...	Cheeseburger	2014	71
Skunks	...	1	0	0	...	Baconburger	2015	23
...	

Joining Data and Information

	...	Oct	Nov	Dec	...	Product	Year	# sold
...	
Cats	...	15	33	18	...	Hamburger	2013	13
Dogs	...	12	17	88	...	Cheeseburger	2014	71
Skunks	...	1	0	0	...	Baconburger	2015	23
...	

- ▶ Integrating two tables — difficult

Joining Data and Information

	...	Oct	Nov	Dec	...	Product	Year	# sold
...	
Cats	...	15	33	18	...	Hamburger	2013	13
Dogs	...	12	17	88	...	Cheeseburger	2014	71
Skunks	...	1	0	0	...	Baconburger	2015	23
...	

- ▶ Integrating two tables — difficult
- ▶ Integrating Information

Joining Data and Information

	...	Oct	Nov	Dec	...	Product	Year	# sold
...	
Cats	...	15	33	18	...	Hamburger	2013	13
Dogs	...	12	17	88	...	Cheeseburger	2014	71
Skunks	...	1	0	0	...	Baconburger	2015	23
...	

- ▶ Integrating two tables — difficult
- ▶ Integrating Information
 - ▶ “17 dogs sold in November,” and

Joining Data and Information

	...	Oct	Nov	Dec	...	Product	Year	# sold
...	
Cats	...	15	33	18	...	Hamburger	2013	13
Dogs	...	12	17	88	...	Cheeseburger	2014	71
Skunks	...	1	0	0	...	Baconburger	2015	23
...	

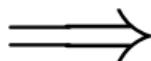
- ▶ Integrating two tables — difficult
- ▶ Integrating Information
 - ▶ “17 dogs sold in November,” **and**
 - ▶ “71 cheeseburgers sold in 2014.”

Knowledge Representation

Real thing



(picture)

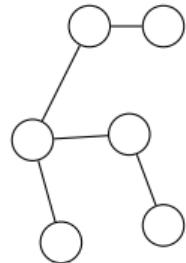
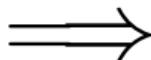


01100010
11011001
01011011
10100011
00100...

Representation

- ▶ Data (i.e. bits) representing the picture
- ▶ Operation: Increase brightness

"*MyBurgers* sold 17 cheeseburgers in June 2018."
(knowledge)



- ▶ Data structure, representing knowledge
- ▶ Operation: Reasoning, "*MyBurger* is a Burger Restaurant"

Let's go to the cinema!

► Lagunen, 20:00...

Let's go to the cinema!

- ▶ Lagunen, 20:00...
- ▶ “Let's go to see *Avatar 2* now!”



Let's go to the cinema!

- ▶ Lagunen, 20:00...
- ▶ "Let's go to see *Avatar 2* now!"
- ▶ Need to find out which cinema plays the movie tonight, e.g. on <http://www.google.no/movies>



Let's go to the cinema!

- ▶ Lagunen, 20:00...
- ▶ "Let's go to see *Avatar 2* now!"
- ▶ Need to find out which cinema plays the movie tonight, e.g. on <http://www.google.no/movies>
- ▶ Need to find out where those cinemas are



Let's go to the cinema!

- ▶ Lagunen, 20:00...
- ▶ "Let's go to see *Avatar 2* now!"
- ▶ Need to find out which cinema plays the movie tonight, e.g. on <http://www.google.no/movies>
- ▶ Need to find out where those cinemas are
- ▶ Need to find out which of those cinemas we can reach on time using public transport, e.g. on <http://skyss.no/>



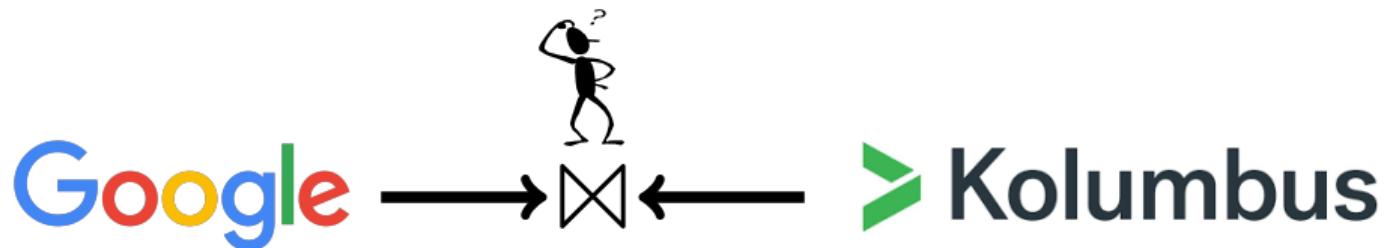
Let's go to the cinema!

- ▶ Lagunen, 20:00...
- ▶ "Let's go to see *Avatar 2* now!"
- ▶ Need to find out which cinema plays the movie tonight, e.g. on <http://www.google.no/movies>
- ▶ Need to find out where those cinemas are
- ▶ Need to find out which of those cinemas we can reach on time using public transport, e.g. on <http://skyss.no/>
- ▶ Web user needs to combine information from different sites



Let's go to the cinema!

- ▶ Lagunen, 20:00...
- ▶ "Let's go to see *Avatar 2* now!"
- ▶ Need to find out which cinema plays the movie tonight, e.g. on <http://www.google.no/movies>
- ▶ Need to find out where those cinemas are
- ▶ Need to find out which of those cinemas we can reach on time using public transport, e.g. on <http://skyss.no/>
- ▶ Web user needs to combine information from different sites
- ▶ Essentially a database join!



The Solution?

- ▶ Wait for Google to produce a Cinema+Public Transport mashup?

The Solution?

- ▶ Wait for Google to produce a Cinema+Public Transport mashup?
- ▶ But what about

The Solution?

- ▶ Wait for Google to produce a Cinema+Public Transport mashup?
- ▶ But what about
 - ▶ Real estate + public transport?

The Solution?

- ▶ Wait for Google to produce a Cinema+Public Transport mashup?
- ▶ But what about
 - ▶ Real estate + public transport?
 - ▶ Plane schedules and pricing + weather information?

The Solution?

- ▶ Wait for Google to produce a Cinema+Public Transport mashup?
- ▶ But what about
 - ▶ Real estate + public transport?
 - ▶ Plane schedules and pricing + weather information?
 - ▶ Car rental + tourism?

The Solution?

- ▶ Wait for Google to produce a Cinema+Public Transport mashup?
- ▶ But what about
 - ▶ Real estate + public transport?
 - ▶ Plane schedules and pricing + weather information?
 - ▶ Car rental + tourism?
 - ▶ Public information + private information (preferences, calendar, location, etc.)

The Solution?

- ▶ Wait for Google to produce a Cinema+Public Transport mashup?
- ▶ But what about
 - ▶ Real estate + public transport?
 - ▶ Plane schedules and pricing + weather information?
 - ▶ Car rental + tourism?
 - ▶ Public information + private information (preferences, calendar, location, etc.)
 - ▶ Chronostratigraphic unit (+ field) + porosity

The Solution?

- ▶ Wait for Google to produce a Cinema+Public Transport mashup?
- ▶ But what about
 - ▶ Real estate + public transport?
 - ▶ Plane schedules and pricing + weather information?
 - ▶ Car rental + tourism?
 - ▶ Public information + private information (preferences, calendar, location, etc.)
 - ▶ Chronostratigraphic unit (+ field) + porosity
- ▶ Can hardly wait for a separate mashup for each useful combination!

The Solution?

- ▶ Wait for Google to produce a Cinema+Public Transport mashup?
- ▶ But what about
 - ▶ Real estate + public transport?
 - ▶ Plane schedules and pricing + weather information?
 - ▶ Car rental + tourism?
 - ▶ Public information + private information (preferences, calendar, location, etc.)
 - ▶ Chronostratigraphic unit (+ field) + porosity
- ▶ Can hardly wait for a separate mashup for each useful combination!
- ▶ Don't want to make them all ourselves?

The Solution?

- ▶ Wait for Google to produce a Cinema+Public Transport mashup?
- ▶ But what about
 - ▶ Real estate + public transport?
 - ▶ Plane schedules and pricing + weather information?
 - ▶ Car rental + tourism?
 - ▶ Public information + private information (preferences, calendar, location, etc.)
 - ▶ Chronostratigraphic unit (+ field) + porosity
- ▶ Can hardly wait for a separate mashup for each useful combination!
- ▶ Don't want to make them all ourselves?
- ▶ Can we automate this?

Machine-Processable Knowledge!

Imagine...

- ▶ All those data sources are published in a machine-readable format.

Machine-Processable Knowledge!

Imagine...

- ▶ All those data sources are published in a machine-readable format.
- ▶ The information published by different sources is *linked*

Machine-Processable Knowledge!

Imagine...

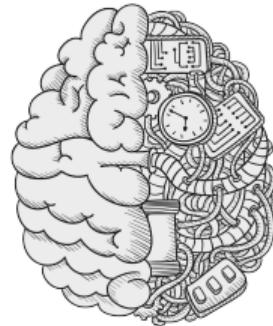
- ▶ All those data sources are published in a machine-readable format.
- ▶ The information published by different sources is *linked*
- ▶ Enough domain knowledge is available to machines to make use of the information

Machine-Processable Knowledge!

Imagine...

- ▶ All those data sources are published in a machine-readable format.
- ▶ The information published by different sources is *linked*
- ▶ Enough domain knowledge is available to machines to make use of the information

Integration of data sets comes within reach!



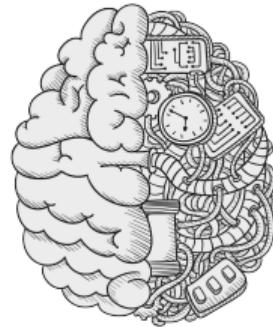
Machine-Processable Knowledge!

Imagine...

- ▶ All those data sources are published in a machine-readable format.
- ▶ The information published by different sources is *linked*
- ▶ Enough domain knowledge is available to machines to make use of the information

Integration of data sets comes within reach!

- ▶ Will rely on:



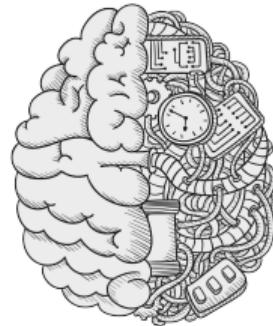
Machine-Processable Knowledge!

Imagine...

- ▶ All those data sources are published in a machine-readable format.
- ▶ The information published by different sources is *linked*
- ▶ Enough domain knowledge is available to machines to make use of the information

Integration of data sets comes within reach!

- ▶ Will rely on:
 - ▶ Domain Modelling



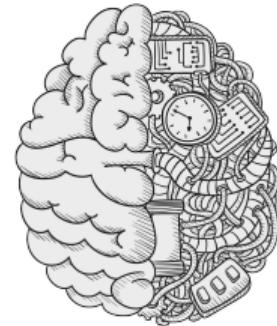
Machine-Processable Knowledge!

Imagine...

- ▶ All those data sources are published in a machine-readable format.
- ▶ The information published by different sources is *linked*
- ▶ Enough domain knowledge is available to machines to make use of the information

Integration of data sets comes within reach!

- ▶ Will rely on:
 - ▶ Domain Modelling
 - ▶ Reasoning with Knowledge



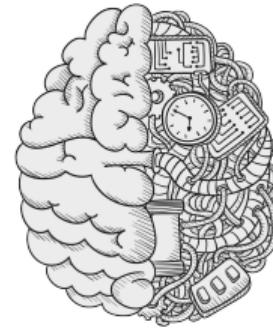
Machine-Processable Knowledge!

Imagine...

- ▶ All those data sources are published in a machine-readable format.
- ▶ The information published by different sources is *linked*
- ▶ Enough domain knowledge is available to machines to make use of the information

Integration of data sets comes within reach!

- ▶ Will rely on:
 - ▶ Domain Modelling
 - ▶ Reasoning with Knowledge
 - ▶ Moving knowledge representations over the 'net



Universals and Instances

Universals

- ▶ Human
- ▶ Burger shop
- ▶ Cheeseburger
- ▶ sells
- ▶ Sandstone

Instances

- ▶ Robert
- ▶ MyBurger
- ▶ cheeseburger123
- ▶ core sample 2 from 6507/10-1

Knowledge

- ▶ All humans are mammals.
- ▶ All Cheeseburgers are Hamburgers.
- ▶ Everything that sells a hamburger is a Burger Shop.

- ▶ Robert is a Human.
- ▶ Robert buys cb123.
- ▶ cb123 is a Cheeseburger.

Universals and Instances (cont.)

Knowledge

- ▶ All humans are mammals.
 - ▶ All Cheeseburgers are Hamburgers.
 - ▶ Everything that sells a hamburger is a Burger Shop.
 - ▶ Robert is a Human.
 - ▶ Robert buys cb123.
 - ▶ cb123 is a Cheeseburger.
-

- ▶ Conceptual model, Domain Model
- ▶ Ontology
- ▶ TBox
- ▶ Languages: **RDFS, OWL**
- ▶ Instance data
- ▶ Assertional knowledge
- ▶ ABox
- ▶ Language: **RDF**

Building Models

- A *model* is a simplified representation of certain aspects of the real world.

Building Models

- ▶ A *model* is a simplified representation of certain aspects of the real world.
- ▶ Made for



Building Models

- ▶ A *model* is a simplified representation of certain aspects of the real world.
- ▶ Made for
 - ▶ understanding



Building Models

- ▶ A *model* is a simplified representation of certain aspects of the real world.
- ▶ Made for
 - ▶ understanding
 - ▶ structuring



Building Models

- ▶ A *model* is a simplified representation of certain aspects of the real world.
- ▶ Made for
 - ▶ understanding
 - ▶ structuring
 - ▶ predicting



Building Models

- ▶ A *model* is a simplified representation of certain aspects of the real world.
- ▶ Made for
 - ▶ understanding
 - ▶ structuring
 - ▶ predicting
 - ▶ communicating



Building Models

- ▶ A *model* is a simplified representation of certain aspects of the real world.
- ▶ Made for
 - ▶ understanding
 - ▶ structuring
 - ▶ predicting
 - ▶ communicating
- ▶ Can be



Building Models

- ▶ A *model* is a simplified representation of certain aspects of the real world.
- ▶ Made for
 - ▶ understanding
 - ▶ structuring
 - ▶ predicting
 - ▶ communicating
- ▶ Can be
 - ▶ Taxonomies (e.g. species, genus, family, etc. in biology)



Building Models

- ▶ A *model* is a simplified representation of certain aspects of the real world.
- ▶ Made for
 - ▶ understanding
 - ▶ structuring
 - ▶ predicting
 - ▶ communicating
- ▶ Can be
 - ▶ Taxonomies (e.g. species, genus, family, etc. in biology)
 - ▶ Domain models, e.g. in UML



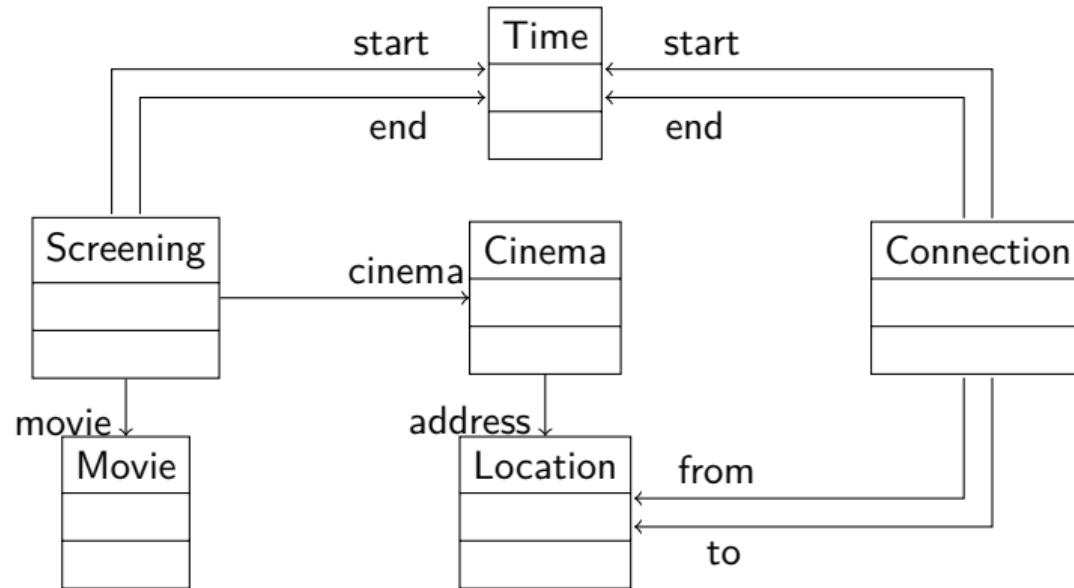
Building Models

- ▶ A *model* is a simplified representation of certain aspects of the real world.
- ▶ Made for
 - ▶ understanding
 - ▶ structuring
 - ▶ predicting
 - ▶ communicating
- ▶ Can be
 - ▶ Taxonomies (e.g. species, genus, family, etc. in biology)
 - ▶ Domain models, e.g. in UML
 - ▶ Numerical Models (Newtonian mechanics, Quantum mechanics)



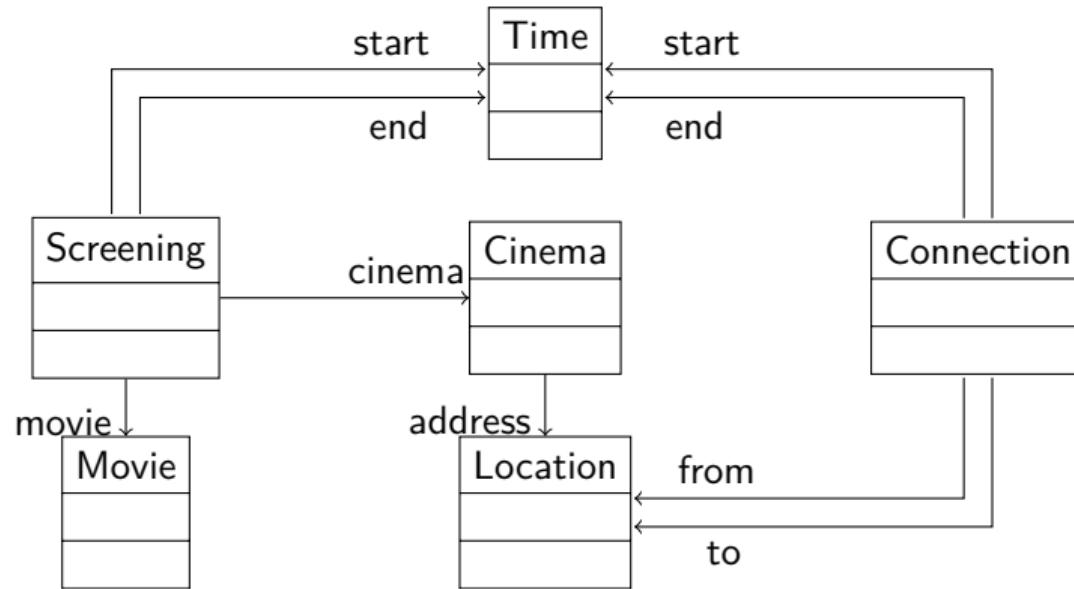
A Cinema Transport Model

An example of a UML domain model:



A Cinema Transport Model

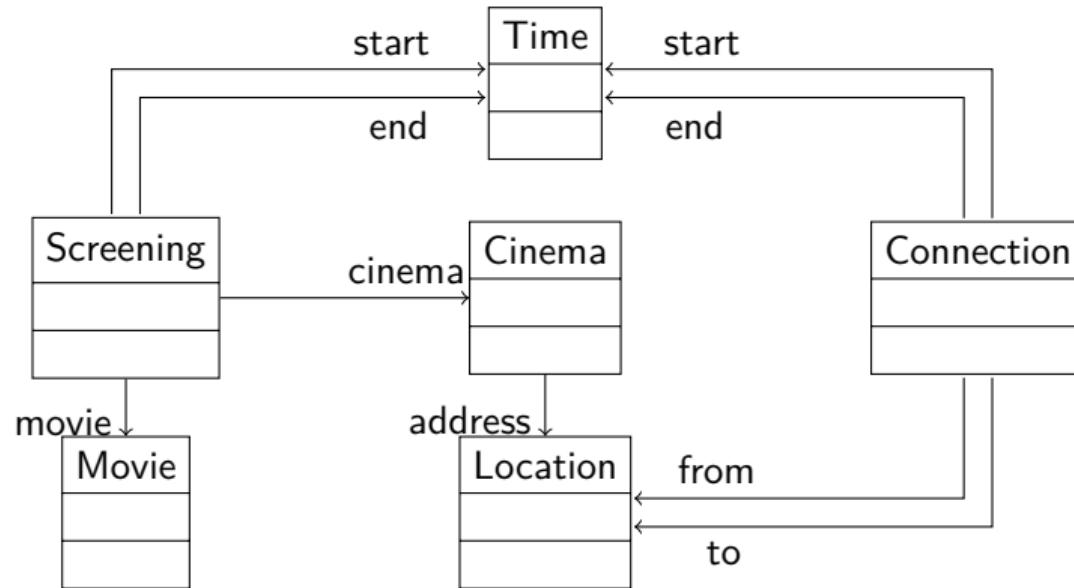
An example of a UML domain model:



- ▶ What is the vocabulary?

A Cinema Transport Model

An example of a UML domain model:



- What is the vocabulary?
- How is it connected?

A Query

What is it we want?

- ▶ Screening(s), movie(s , AV2)



A Query

What is it we want?

- ▶ Screening(s), movie(s , AV2)
- ▶ cinema(s, k), address(k, l)



A Query

What is it we want?

- ▶ Screening(s), movie(s , AV2)
- ▶ cinema(s, k), address(k, l)
- ▶ Connection(c), from($c, ETOILE$), to(c, l)



A Query

What is it we want?

- ▶ Screening(s), movie(s , AV2)
- ▶ cinema(s, k), address(k, l)
- ▶ Connection(c), from($c, ETOILE$), to(c, l)
- ▶ start($c, cStart$), before(20:00, $cStart$)



A Query

What is it we want?

- ▶ Screening(s), movie(s , AV2)
- ▶ cinema(s, k), address(k, l)
- ▶ Connection(c), from($c, ETOILE$), to(c, l)
- ▶ start($c, cStart$), before(20:00, $cStart$)
- ▶ end($c, cEnd$), start($s, sStart$), before($cEnd, sStart$)



A Query

What is it we want?

- ▶ Screening(s), movie(s , AV2)
- ▶ cinema(s, k), address(k, l)
- ▶ Connection(c), from($c, ETOILE$), to(c, l)
- ▶ start($c, cStart$), before(20:00, $cStart$)
- ▶ end($c, cEnd$), start($s, sStart$), before($cEnd, sStart$)



Find $s, k, l, c, cStart, cEnd, sStart$ satisfying this and we have the answer!

A Query

What is it we want?

- ▶ Screening(s), movie(s , AV2)
- ▶ cinema(s, k), address(k, l)
- ▶ Connection(c), from($c, ETOILE$), to(c, l)
- ▶ start($c, cStart$), before(20:00, $cStart$)
- ▶ end($c, cEnd$), start($s, sStart$), before($cEnd, sStart$)



Find $s, k, l, c, cStart, cEnd, sStart$ satisfying this and we have the answer!

- ▶ Maybe not the easiest way to ask, but it's a start.
- ▶ Need models to formulate queries.
- ▶ Need to connect models from different domains.

Conceptual Modelling – Challenges

- ▶ Choice of concepts to model



Conceptual Modelling – Challenges

- ▶ Choice of concepts to model
 - ▶ depending on the use-case



Conceptual Modelling – Challenges

- ▶ Choice of concepts to model
 - ▶ depending on the use-case
- ▶ Delimitation of concepts, e.g.:



Conceptual Modelling – Challenges

- ▶ Choice of concepts to model
 - ▶ depending on the use-case
- ▶ Delimitation of concepts, e.g.:
 - ▶ If it doesn't show any movies, is it a cinema?



Conceptual Modelling – Challenges

- ▶ Choice of concepts to model
 - ▶ depending on the use-case
- ▶ Delimitation of concepts, e.g.:
 - ▶ If it doesn't show any movies, is it a cinema?
 - ▶ Is it a mountain or a hill?



Conceptual Modelling – Challenges

- ▶ Choice of concepts to model
 - ▶ depending on the use-case
- ▶ Delimitation of concepts, e.g.:
 - ▶ If it doesn't show any movies, is it a cinema?
 - ▶ Is it a mountain or a hill?
 - ▶ Is the hole in the table part of the table?



Conceptual Modelling – Challenges

- ▶ Choice of concepts to model
 - ▶ depending on the use-case
- ▶ Delimitation of concepts, e.g.:
 - ▶ If it doesn't show any movies, is it a cinema?
 - ▶ Is it a mountain or a hill?
 - ▶ Is the hole in the table part of the table?
 - ▶ Solution: often dictated by



Conceptual Modelling – Challenges

- ▶ Choice of concepts to model
 - ▶ depending on the use-case
- ▶ Delimitation of concepts, e.g.:
 - ▶ If it doesn't show any movies, is it a cinema?
 - ▶ Is it a mountain or a hill?
 - ▶ Is the hole in the table part of the table?
 - ▶ Solution: often dictated by
 - ▶ laws and regulations



Conceptual Modelling – Challenges

- ▶ Choice of concepts to model
 - ▶ depending on the use-case
- ▶ Delimitation of concepts, e.g.:
 - ▶ If it doesn't show any movies, is it a cinema?
 - ▶ Is it a mountain or a hill?
 - ▶ Is the hole in the table part of the table?
 - ▶ Solution: often dictated by
 - ▶ laws and regulations
 - ▶ use-case



Conceptual Modelling – Challenges

- ▶ Choice of concepts to model
 - ▶ depending on the use-case
- ▶ Delimitation of concepts, e.g.:
 - ▶ If it doesn't show any movies, is it a cinema?
 - ▶ Is it a mountain or a hill?
 - ▶ Is the hole in the table part of the table?
 - ▶ Solution: often dictated by
 - ▶ laws and regulations
 - ▶ use-case
- ▶ Models that stay correct under changes and extensions



Conceptual Modelling – Challenges

- ▶ Choice of concepts to model
 - ▶ depending on the use-case
- ▶ Delimination of concepts, e.g.:
 - ▶ If it doesn't show any movies, is it a cinema?
 - ▶ Is it a mountain or a hill?
 - ▶ Is the hole in the table part of the table?
 - ▶ Solution: often dictated by
 - ▶ laws and regulations
 - ▶ use-case
- ▶ Models that stay correct under changes and extensions
 - ▶ E.g. “Employee is a subclass of Human”



Conceptual Modelling – Challenges

- ▶ Choice of concepts to model
 - ▶ depending on the use-case
- ▶ Delimination of concepts, e.g.:
 - ▶ If it doesn't show any movies, is it a cinema?
 - ▶ Is it a mountain or a hill?
 - ▶ Is the hole in the table part of the table?
 - ▶ Solution: often dictated by
 - ▶ laws and regulations
 - ▶ use-case
- ▶ Models that stay correct under changes and extensions
 - ▶ E.g. “Employee is a subclass of Human”
 - ▶ Only for some time, and some employer



Conceptual Modelling – Challenges

- ▶ Choice of concepts to model
 - ▶ depending on the use-case
- ▶ Delimination of concepts, e.g.:
 - ▶ If it doesn't show any movies, is it a cinema?
 - ▶ Is it a mountain or a hill?
 - ▶ Is the hole in the table part of the table?
 - ▶ Solution: often dictated by
 - ▶ laws and regulations
 - ▶ use-case
- ▶ Models that stay correct under changes and extensions
 - ▶ E.g. "Employee is a subclass of Human"
 - ▶ Only for some time, and some employer
 - ▶ Solution: Need to think beyond use case



Calculating

- ▶ What is calculation?

Calculating

- What is calculation?

$$\begin{array}{r} A \text{ owns } x \text{ Bs} \\ A \text{ gets another } y \text{ Bs} \\ \hline A \text{ now owns } (x + y) \text{ Bs} \end{array}$$

Calculating

- What is calculation?

$$\begin{array}{r} A \text{ owns } x \text{ Bs} \\ A \text{ gets another } y \text{ Bs} \\ \hline A \text{ now owns } (x + y) \text{ Bs} \end{array}$$

e.g.

$$\begin{array}{r} \text{Peter owns 1 apple} \\ \text{Peter gets another 4 apples} \\ \hline \text{Peter now owns 5 apples} \end{array}$$



Calculating

- ▶ What is calculation?

$$\begin{array}{r} A \text{ owns } x \text{ Bs} \\ A \text{ gets another } y \text{ Bs} \\ \hline A \text{ now owns } (x + y) \text{ Bs} \end{array}$$

e.g.

$$\begin{array}{r} \text{Peter owns 1 apple} \\ \text{Peter gets another 4 apples} \\ \hline \text{Peter now owns 5 apples} \end{array}$$



- ▶ Calculation is algorithmic manipulation of numbers...

Calculating

- ▶ What is calculation?

$$\begin{array}{r} A \text{ owns } x \text{ Bs} \\ A \text{ gets another } y \text{ Bs} \\ \hline A \text{ now owns } (x + y) \text{ Bs} \end{array}$$

e.g.

$$\begin{array}{r} \text{Peter owns 1 apple} \\ \text{Peter gets another 4 apples} \\ \hline \text{Peter now owns 5 apples} \end{array}$$



- ▶ Calculation is algorithmic manipulation of numbers...
- ▶ ...where the *meaning* of the numbers is not needed

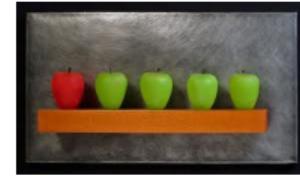
Calculating

- ▶ What is calculation?

$$\begin{array}{r} A \text{ owns } x \text{ Bs} \\ A \text{ gets another } y \text{ Bs} \\ \hline A \text{ now owns } (x + y) \text{ Bs} \end{array}$$

e.g.

$$\begin{array}{r} \text{Peter owns 1 apple} \\ \text{Peter gets another 4 apples} \\ \hline \text{Peter now owns 5 apples} \end{array}$$



- ▶ Calculation is algorithmic manipulation of numbers...
- ▶ ...where the *meaning* of the numbers is not needed
- ▶ Can calculate $1 + 4 = 5$ without knowing what is counted

Calculating

- ▶ What is calculation?

$$\begin{array}{r} A \text{ owns } x \text{ Bs} \\ A \text{ gets another } y \text{ Bs} \\ \hline A \text{ now owns } (x + y) \text{ Bs} \end{array}$$

e.g.

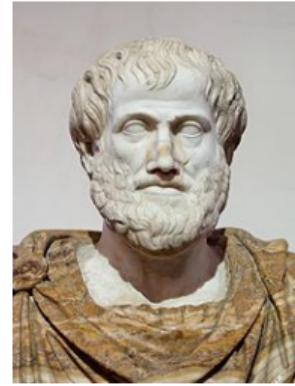
$$\begin{array}{r} \text{Peter owns 1 apple} \\ \text{Peter gets another 4 apples} \\ \hline \text{Peter now owns 5 apples} \end{array}$$



- ▶ Calculation is algorithmic manipulation of numbers...
- ▶ ...where the *meaning* of the numbers is not needed
- ▶ Can calculate $1 + 4 = 5$ without knowing what is counted
- ▶ Abstraction!

Calculating with Knowledge

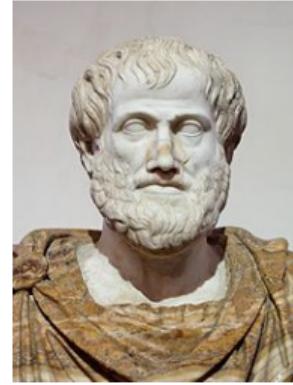
- ▶ Can be traced back to Aristotle (384–322 BC)



Calculating with Knowledge

- ▶ Can be traced back to Aristotle (384–322 BC)
- ▶ Modus Barbara:

$$\begin{array}{c} \text{All } A \text{ are } B \\ \text{All } B \text{ are } C \\ \hline \text{All } A \text{ are } C \end{array}$$



Calculating with Knowledge

- ▶ Can be traced back to Aristotle (384–322 BC)
- ▶ Modus Barbara:

All *A* are *B*

All *B* are *C*

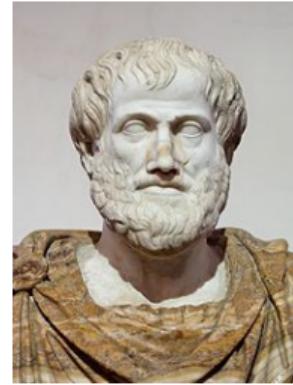
All *A* are *C*

e.g.

All Greeks are men

All men are mortal

All Greeks are mortal



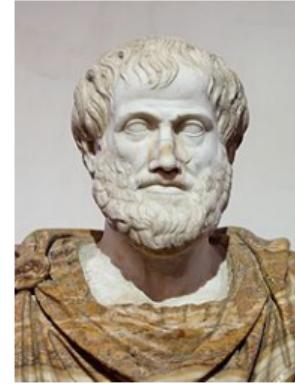
Calculating with Knowledge

- ▶ Can be traced back to Aristotle (384–322 BC)
- ▶ Modus Barbara:

$$\begin{array}{c} \text{All } A \text{ are } B \\ \text{All } B \text{ are } C \\ \hline \text{All } A \text{ are } C \end{array}$$

e.g.

$$\begin{array}{c} \text{All Greeks are men} \\ \text{All men are mortal} \\ \hline \text{All Greeks are mortal} \end{array}$$



- ▶ Algorithmic manipulation of *knowledge representations*...

Calculating with Knowledge

- ▶ Can be traced back to Aristotle (384–322 BC)
- ▶ Modus Barbara:

All A are B

All B are C

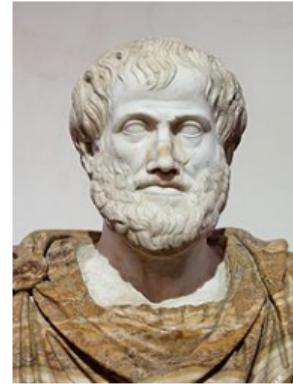
All A are C

e.g.

All Greeks are men

All men are mortal

All Greeks are mortal



- ▶ Algorithmic manipulation of *knowledge representations*...
- ▶ ...where the *meaning* of the words is not needed!

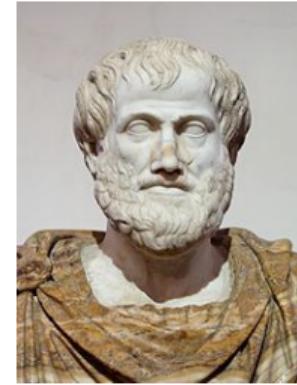
Calculating with Knowledge

- ▶ Can be traced back to Aristotle (384–322 BC)
- ▶ Modus Barbara:

$$\begin{array}{c} \text{All } A \text{ are } B \\ \text{All } B \text{ are } C \\ \hline \text{All } A \text{ are } C \end{array}$$

e.g.

$$\begin{array}{c} \text{All Greeks are men} \\ \text{All men are mortal} \\ \hline \text{All Greeks are mortal} \end{array}$$



- ▶ Algorithmic manipulation of *knowledge representations*...
- ▶ ...where the *meaning* of the words is not needed!
- ▶ Also an abstraction!

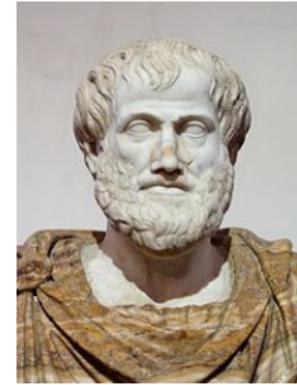
Calculating with Knowledge

- ▶ Can be traced back to Aristotle (384–322 BC)
- ▶ Modus Barbara:

$$\begin{array}{c} \text{All } A \text{ are } B \\ \text{All } B \text{ are } C \\ \hline \text{All } A \text{ are } C \end{array}$$

e.g.

$$\begin{array}{c} \text{All Greeks are men} \\ \text{All men are mortal} \\ \hline \text{All Greeks are mortal} \end{array}$$



- ▶ Algorithmic manipulation of *knowledge representations*...
- ▶ ...where the *meaning* of the words is not needed!
- ▶ Also an abstraction!
- ▶ The topic of *formal logic* and *computational logic*

Computing with Knowledge About Movies

- ▶ Query: find a *fun* event we can reach by public transport

Computing with Knowledge About Movies

- ▶ Query: find a *fun* event we can reach by public transport
- ▶ Knowledge base:

Computing with Knowledge About Movies

- ▶ Query: find a *fun* event we can reach by public transport
- ▶ Knowledge base:
 1. A movie screening is an event

Computing with Knowledge About Movies

- ▶ Query: find a *fun* event we can reach by public transport
- ▶ Knowledge base:
 1. A movie screening is an event
 2. A movie screening is fun if the movie being shown is not a documentary

Computing with Knowledge About Movies

- ▶ Query: find a *fun* event we can reach by public transport
- ▶ Knowledge base:
 1. A movie screening is an event
 2. A movie screening is fun if the movie being shown is not a documentary
 3. James Cameron does not direct documentaries

Computing with Knowledge About Movies

- ▶ Query: find a *fun* event we can reach by public transport
- ▶ Knowledge base:
 1. A movie screening is an event
 2. A movie screening is fun if the movie being shown is not a documentary
 3. James Cameron does not direct documentaries
 4. James Cameron directed *Avatar 2*

Computing with Knowledge About Movies

- ▶ Query: find a *fun* event we can reach by public transport
- ▶ Knowledge base:
 1. A movie screening is an event
 2. A movie screening is fun if the movie being shown is not a documentary
 3. James Cameron does not direct documentaries
 4. James Cameron directed *Avatar 2*
 5. There is a screening of *Avatar 2* at 19:00.

Computing with Knowledge About Movies

- ▶ Query: find a *fun* event we can reach by public transport
- ▶ Knowledge base:
 1. A movie screening is an event
 2. A movie screening is fun if the movie being shown is not a documentary
 3. James Cameron does not direct documentaries
 4. James Cameron directed *Avatar 2*
 5. There is a screening of *Avatar 2* at 19:00.

...

Computing with Knowledge About Movies

- ▶ Query: find a *fun* event we can reach by public transport
- ▶ Knowledge base:
 1. A movie screening is an event
 2. A movie screening is fun if the movie being shown is not a documentary
 3. James Cameron does not direct documentaries
 4. James Cameron directed *Avatar 2*
 5. There is a screening of *Avatar 2* at 19:00.

...
- ▶ Let us calculate...

Computing with Knowledge About Movies

- ▶ Query: find a *fun* event we can reach by public transport
- ▶ Knowledge base:
 1. A movie screening is an event
 2. A movie screening is fun if the movie being shown is not a documentary
 3. James Cameron does not direct documentaries
 4. James Cameron directed *Avatar 2*
 5. There is a screening of *Avatar 2* at 19:00.

...
- ▶ Let us calculate...
 6. From 3 and 4: *Avatar 2* is not a documentary

Computing with Knowledge About Movies

- ▶ Query: find a *fun* event we can reach by public transport
- ▶ Knowledge base:
 1. A movie screening is an event
 2. A movie screening is fun if the movie being shown is not a documentary
 3. James Cameron does not direct documentaries
 4. James Cameron directed *Avatar 2*
 5. There is a screening of *Avatar 2* at 19:00.

...
- ▶ Let us calculate...
 6. From 3 and 4: *Avatar 2* is not a documentary
 7. From 6 and 2: A screening of *Avatar 2* is fun

Computing with Knowledge About Movies

- ▶ Query: find a *fun* event we can reach by public transport
- ▶ Knowledge base:
 1. A movie screening is an event
 2. A movie screening is fun if the movie being shown is not a documentary
 3. James Cameron does not direct documentaries
 4. James Cameron directed *Avatar 2*
 5. There is a screening of *Avatar 2* at 19:00.

...
- ▶ Let us calculate...
 6. From 3 and 4: *Avatar 2* is not a documentary
 7. From 6 and 2: A screening of *Avatar 2* is fun
 8. From 1, 5, 7: there is a fun event at 19:00

Computing with Knowledge About Movies

- ▶ Query: find a *fun* event we can reach by public transport
- ▶ Knowledge base:
 1. A movie screening is an event
 2. A movie screening is fun if the movie being shown is not a documentary
 3. James Cameron does not direct documentaries
 4. James Cameron directed *Avatar 2*
 5. There is a screening of *Avatar 2* at 19:00.

...
- ▶ Let us calculate...
 6. From 3 and 4: *Avatar 2* is not a documentary
 7. From 6 and 2: A screening of *Avatar 2* is fun
 8. From 1, 5, 7: there is a fun event at 19:00

...

Computing with Knowledge About Movies

- ▶ Query: find a *fun* event we can reach by public transport
- ▶ Knowledge base:
 1. A movie screening is an event
 2. A movie screening is fun if the movie being shown is not a documentary
 3. James Cameron does not direct documentaries
 4. James Cameron directed *Avatar 2*
 5. There is a screening of *Avatar 2* at 19:00.

...
- ▶ Let us calculate...
 6. From 3 and 4: *Avatar 2* is not a documentary
 7. From 6 and 2: A screening of *Avatar 2* is fun
 8. From 1, 5, 7: there is a fun event at 19:00

...
- ▶ Computing with Knowledge connects the information to the questions!

Data Modelling – Schemas, Constraints

Constraints describe the *data*, i.e., information about data:

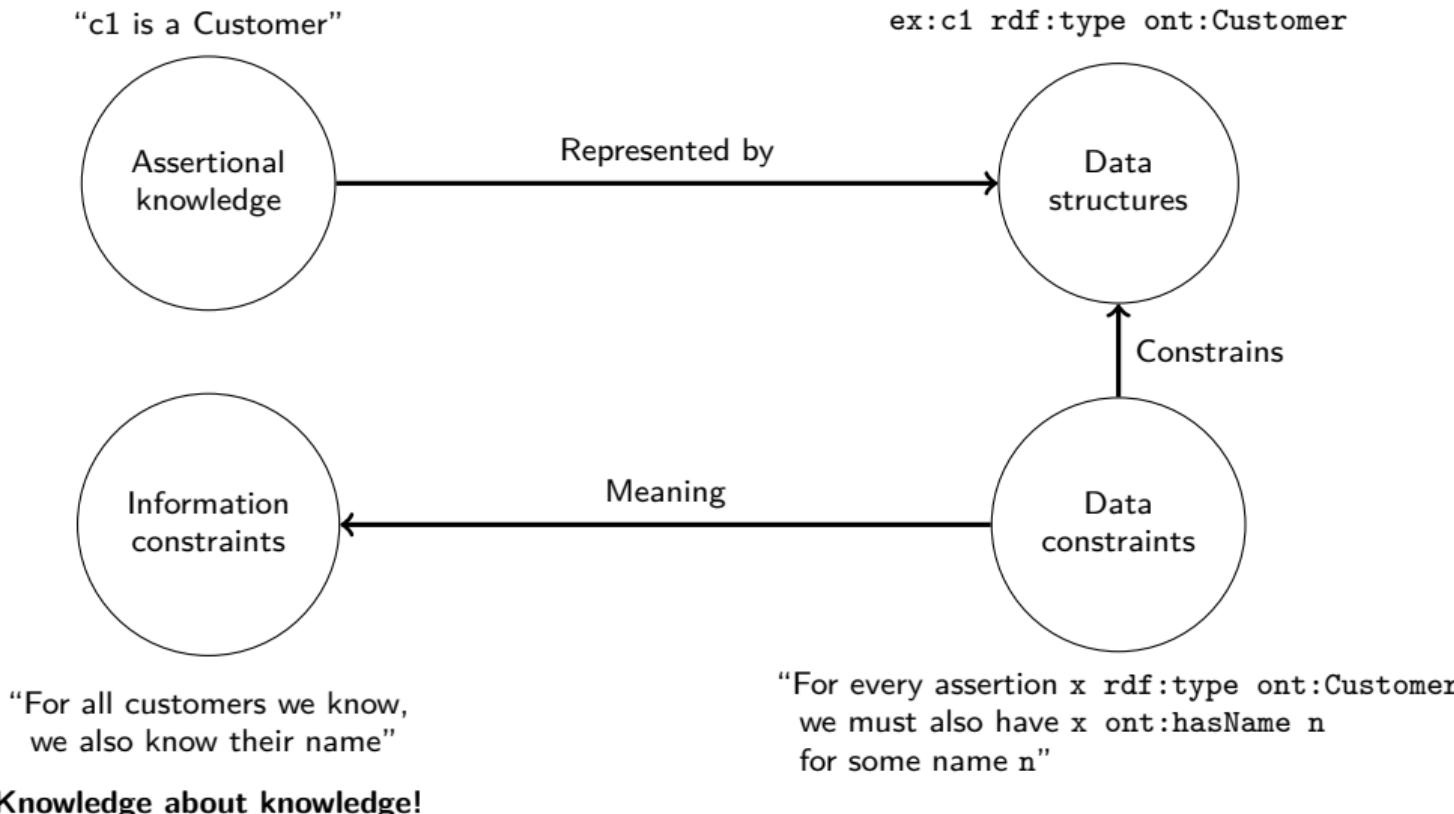
- ▶ For every row in the Customer table, the Name-column is non-NULL.
- ▶ The <name>-element is always in the <customer>-element.
- ▶ There is at most 1 <name>-element in <customer>.

which is different from

- ▶ Every customer has a name.

which is knowledge, i.e., information about customers.

Information Model



Ontology vs. Information Model

Ontology

- ▶ Knowledge about domain
- ▶ *Can do:* infer new knowledge
- ▶ Reuse across applications
- ▶ Languages: RDFS, OWL

Information model

- ▶ Knowledge about our knowledge of the domain
- ▶ *Can do:* check completeness of existing information
- ▶ Specific to use (one system or exchange)
- ▶ Languages: SHACL, ShEx

What about the Semantic Web?

- ▶ Much of the semantic technologies originally motivated by Tim Berners-Lee's "Semantic Web" vision



Tim Berners-Lee

What about the Semantic Web?

- ▶ Much of the semantic technologies originally motivated by Tim Berners-Lee's "Semantic Web" vision
- ▶ Web of Data:



Tim Berners-Lee

What about the Semantic Web?

- ▶ Much of the semantic technologies originally motivated by Tim Berners-Lee's "Semantic Web" vision
- ▶ Web of Data:
 - ▶ data distributed over the web



Tim Berners-Lee

What about the Semantic Web?

- ▶ Much of the semantic technologies originally motivated by Tim Berners-Lee's "Semantic Web" vision
- ▶ Web of Data:
 - ▶ data distributed over the web
 - ▶ anybody can publish any information about anything



Tim Berners-Lee

What about the Semantic Web?

- ▶ Much of the semantic technologies originally motivated by Tim Berners-Lee's "Semantic Web" vision
- ▶ Web of Data:
 - ▶ data distributed over the web
 - ▶ anybody can publish any information about anything
 - ▶ intelligent user agents cater to all your information needs



Tim Berners-Lee

What about the Semantic Web?

- ▶ Much of the semantic technologies originally motivated by Tim Berners-Lee's "Semantic Web" vision
- ▶ Web of Data:
 - ▶ data distributed over the web
 - ▶ anybody can publish any information about anything
 - ▶ intelligent user agents cater to all your information needs
 - ▶ the ultimate data integration and interoperability challenge!



Tim Berners-Lee

What about the Semantic Web?

- ▶ Much of the semantic technologies originally motivated by Tim Berners-Lee's "Semantic Web" vision
- ▶ Web of Data:
 - ▶ data distributed over the web
 - ▶ anybody can publish any information about anything
 - ▶ intelligent user agents cater to all your information needs
 - ▶ the ultimate data integration and interoperability challenge!
 - ▶ requires models, inference, global IDs, etc.



Tim Berners-Lee

What about the Semantic Web?

- ▶ Much of the semantic technologies originally motivated by Tim Berners-Lee's "Semantic Web" vision
- ▶ Web of Data:
 - ▶ data distributed over the web
 - ▶ anybody can publish any information about anything
 - ▶ intelligent user agents cater to all your information needs
 - ▶ the ultimate data integration and interoperability challenge!
 - ▶ requires models, inference, global IDs, etc.
 - ▶ many problems with discovery, trust,...



Tim Berners-Lee

What about the Semantic Web?

- ▶ Much of the semantic technologies originally motivated by Tim Berners-Lee's "Semantic Web" vision
- ▶ Web of Data:
 - ▶ data distributed over the web
 - ▶ anybody can publish any information about anything
 - ▶ intelligent user agents cater to all your information needs
 - ▶ the ultimate data integration and interoperability challenge!
 - ▶ requires models, inference, global IDs, etc.
 - ▶ many problems with discovery, trust,...
- ▶ Much more realistic in less open "enterprise" settings



Tim Berners-Lee

What about the Semantic Web?

- ▶ Much of the semantic technologies originally motivated by Tim Berners-Lee's "Semantic Web" vision
- ▶ Web of Data:
 - ▶ data distributed over the web
 - ▶ anybody can publish any information about anything
 - ▶ intelligent user agents cater to all your information needs
 - ▶ the ultimate data integration and interoperability challenge!
 - ▶ requires models, inference, global IDs, etc.
 - ▶ many problems with discovery, trust,...
- ▶ Much more realistic in less open "enterprise" settings
- ▶ Talk about "Semantic Technologies" or "Semantic Web Technologies"



Tim Berners-Lee

The “Home” of the Semantic Web

See the W3C pages for the Semantic Web effort:

<http://www.w3.org/2001/sw/>

For standards (RDF, OWL, etc.), see:

http://www.w3.org/2001/sw/wiki/Main_Page



This Tutorial

The aim of this tutorial is to give you a glimpse of...

- ▶ some basic semantic technologies (RDF, OWL)

This Tutorial

The aim of this tutorial is to give you a glimpse of...

- ▶ some basic semantic technologies (RDF, OWL)
- ▶ some underlying concepts like reasoning

This Tutorial

The aim of this tutorial is to give you a glimpse of...

- ▶ some basic semantic technologies (RDF, OWL)
- ▶ some underlying concepts like reasoning
- ▶ ...to give you some overview of the field

This Tutorial

The aim of this tutorial is to give you a glimpse of...

- ▶ some basic semantic technologies (RDF, OWL)
- ▶ some underlying concepts like reasoning
- ▶ ...to give you some overview of the field
- ▶ ...as an entry point for further exploration

This Tutorial

The aim of this tutorial is to give you a glimpse of...

- ▶ some basic semantic technologies (RDF, OWL)
- ▶ some underlying concepts like reasoning
- ▶ ...to give you some overview of the field
- ▶ ...as an entry point for further exploration

For more information:

This Tutorial

The aim of this tutorial is to give you a glimpse of...

- ▶ some basic semantic technologies (RDF, OWL)
- ▶ some underlying concepts like reasoning
- ▶ ...to give you some overview of the field
- ▶ ...as an entry point for further exploration

For more information:

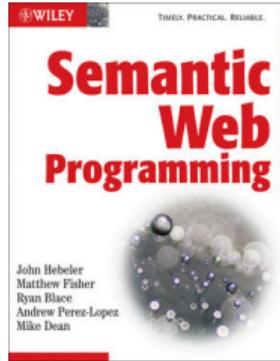
- ▶ the W3C web pages contain the standards

Further Reading

- ▶ For practical aspects:

Semantic Web Programming.

Hebler, Fisher, Blace, Perez-Lopez.
Wiley 2009

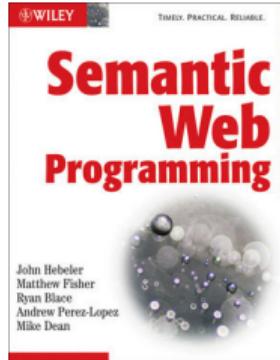


Further Reading

- ▶ For practical aspects:

Semantic Web Programming.

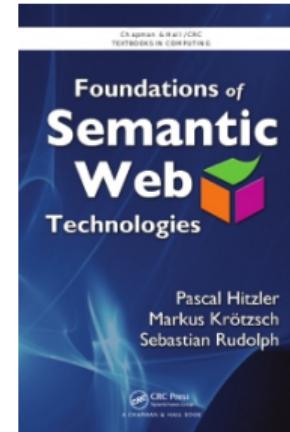
Hebler, Fisher, Blace, Perez-Lopez.
Wiley 2009



- ▶ For theoretical aspects:

Foundations of Semantic Web Technologies.

Hitzler, Krötzsch, Rudolph.
CRC Press 2009



Downloads, Tutorial Web Page

Download and install software needed for the exercises:

- ▶ Protégé: ontology editor <https://protege.stanford.edu/>
- ▶ Text editor (visual studio code)
- ▶ RDFox triple store (no license needed)
<https://www.oxfordsemantic.tech/downloads>

The conceptual components of RDF

The RDF data model in a nutshell:



The conceptual components of RDF

The RDF data model in a nutshell:

- ▶ Information is represented as *triples*.



The conceptual components of RDF

The RDF data model in a nutshell:

- ▶ Information is represented as *triples*.
- ▶ Triples = subject-predicate-object patterns



The conceptual components of RDF

The RDF data model in a nutshell:

- ▶ Information is represented as *triples*.
 - ▶ Triples = subject-predicate-object patterns
- ▶ Things (in a broad sense) are labelled with *URLs*



The conceptual components of RDF

The RDF data model in a nutshell:

- ▶ Information is represented as *triples*.
 - ▶ Triples = subject-predicate-object patterns
- ▶ Things (in a broad sense) are labelled with *URLs*
 - ▶ URLs act as globally valid names



The conceptual components of RDF

The RDF data model in a nutshell:

- ▶ Information is represented as *triples*.
 - ▶ Triples = subject-predicate-object patterns
- ▶ Things (in a broad sense) are labelled with *URLs*
 - ▶ URLs act as globally valid names
- ▶ Sets of names are organized in *vocabularies*



The conceptual components of RDF

The RDF data model in a nutshell:

- ▶ Information is represented as *triples*.
 - ▶ Triples = subject-predicate-object patterns
- ▶ Things (in a broad sense) are labelled with *URLs*
 - ▶ URLs act as globally valid names
- ▶ Sets of names are organized in *vocabularies*
 - ▶ Vocabularies are demarcated by *namespaces*



The conceptual components of RDF

The RDF data model in a nutshell:

- ▶ Information is represented as *triples*.
 - ▶ Triples = subject-predicate-object patterns
- ▶ Things (in a broad sense) are labelled with *URLs*
 - ▶ URLs act as globally valid names
- ▶ Sets of names are organized in *vocabularies*
 - ▶ Vocabularies are demarcated by *namespaces*



We shall look at each in turn.

The conceptual components of RDF

The RDF data model in a nutshell:

- ▶ Information is represented as *triples*.
 - ▶ Triples = subject-predicate-object patterns
- ▶ Things (in a broad sense) are labelled with *URLs*
 - ▶ URLs act as globally valid names
- ▶ Sets of names are organized in *vocabularies*
 - ▶ Vocabularies are demarcated by *namespaces*



We shall look at each in turn.

Standard: <https://www.w3.org/TR/rdf11-concepts/>

The conceptual components of RDF

The RDF data model in a nutshell:

- ▶ Information is represented as *triples*.
 - ▶ Triples = subject-predicate-object patterns
- ▶ Things (in a broad sense) are labelled with *URLs*
 - ▶ URLs act as globally valid names
- ▶ Sets of names are organized in *vocabularies*
 - ▶ Vocabularies are demarcated by *namespaces*



We shall look at each in turn.

Standard: <https://www.w3.org/TR/rdf11-concepts/>

Primer: <https://www.w3.org/2007/02/turtle/primer/>

Triples

- ▶ All information in RDF is expressed as *triples*

Triples

- ▶ All information in RDF is expressed as *triples*
- ▶ A triple consists of

Triples

- ▶ All information in RDF is expressed as *triples*
- ▶ A triple consists of
 - ▶ A **subject**

Triples

- ▶ All information in RDF is expressed as *triples*
- ▶ A triple consists of
 - ▶ A **subject**
 - ▶ A **predicate**

Triples

- ▶ All information in RDF is expressed as *triples*
- ▶ A triple consists of
 - ▶ A **subject**
 - ▶ A **predicate**
 - ▶ An **object**

Triples

- ▶ All information in RDF is expressed as *triples*
- ▶ A triple consists of
 - ▶ A **subject**
 - ▶ A **predicate**
 - ▶ An **object**
- ▶ For instance

Triples

- ▶ All information in RDF is expressed as *triples*
- ▶ A triple consists of
 - ▶ A **subject**
 - ▶ A **predicate**
 - ▶ An **object**
- ▶ For instance
 - ▶ Norway **has capital** Oslo



Triples

- ▶ All information in RDF is expressed as *triples*

- ▶ A triple consists of

- ▶ A **subject**
- ▶ A **predicate**
- ▶ An **object**

- ▶ For instance

- ▶ Norway **has capital** Oslo
- ▶ Oslo **has mayor** Marianne Borgen



Triples

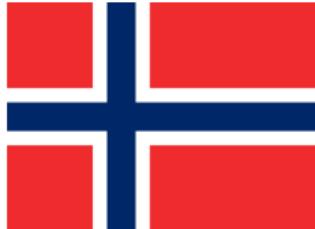
- ▶ All information in RDF is expressed as *triples*

- ▶ A triple consists of

- ▶ A **subject**
- ▶ A **predicate**
- ▶ An **object**

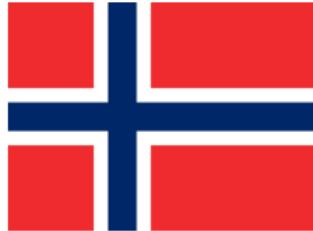
- ▶ For instance

- ▶ Norway **has capital** Oslo
- ▶ Oslo **has mayor** Marianne Borgen
- ▶ Marianne Borgen **has year of birth** 1951



Triples

- ▶ All information in RDF is expressed as *triples*
- ▶ A triple consists of
 - ▶ A **subject**
 - ▶ A **predicate**
 - ▶ An **object**
- ▶ For instance
 - ▶ Norway **has capital** Oslo
 - ▶ Oslo **has mayor** Marianne Borgen
 - ▶ Marianne Borgen **has year of birth** 1951
- ▶ Any other representation of information can be transformed to triples



Triples

- ▶ All information in RDF is expressed as *triples*
- ▶ A triple consists of
 - ▶ A **subject**
 - ▶ A **predicate**
 - ▶ An **object**
- ▶ For instance
 - ▶ Norway **has capital** Oslo
 - ▶ Oslo **has mayor** Marianne Borgen
 - ▶ Marianne Borgen **has year of birth** 1951
- ▶ Any other representation of information can be transformed to triples
- ▶ Another word for an RDF triple is a *statement*



Resource Identifiers

- ▶ RDF (Resource Description Framework) talks about *resources*

Resource Identifiers

- ▶ RDF (Resource Description Framework) talks about *resources*
 - ▶ Resources can be pretty much anything. People, types, relations...

Resource Identifiers

- ▶ RDF (Resource Description Framework) talks about *resources*
 - ▶ Resources can be pretty much anything. People, types, relations...
- ▶ Resources are identified by IRIs (Internationalized Resource Identifiers)

Resource Identifiers

- ▶ RDF (Resource Description Framework) talks about *resources*
 - ▶ Resources can be pretty much anything. People, types, relations...
- ▶ Resources are identified by IRIs (Internationalized Resource Identifiers)
- ▶ e.g. in dbpedia.org :

Resource Identifiers

- ▶ RDF (Resource Description Framework) talks about *resources*
 - ▶ Resources can be pretty much anything. People, types, relations...
- ▶ Resources are identified by IRIs (Internationalized Resource Identifiers)
- ▶ e.g. in dbpedia.org :
 - ▶ Norway: <http://dbpedia.org/resource/Norway>

Resource Identifiers

- ▶ RDF (Resource Description Framework) talks about *resources*
 - ▶ Resources can be pretty much anything. People, types, relations...
- ▶ Resources are identified by IRIs (Internationalized Resource Identifiers)
- ▶ e.g. in dbpedia.org :
 - ▶ Norway: <http://dbpedia.org/resource/Norway>
 - ▶ has capital: <http://dbpedia.org/ontology/capital>

Resource Identifiers

- ▶ RDF (Resource Description Framework) talks about *resources*
 - ▶ Resources can be pretty much anything. People, types, relations...
- ▶ Resources are identified by IRIs (Internationalized Resource Identifiers)
- ▶ e.g. in dbpedia.org :
 - ▶ Norway: <http://dbpedia.org/resource/Norway>
 - ▶ has capital: <http://dbpedia.org/ontology/capital>
 - ▶ Oslo: <http://dbpedia.org/resource/Oslo>

Resource Identifiers

- ▶ RDF (Resource Description Framework) talks about *resources*
 - ▶ Resources can be pretty much anything. People, types, relations...
- ▶ Resources are identified by IRIs (Internationalized Resource Identifiers)
- ▶ e.g. in dbpedia.org :
 - ▶ Norway: <http://dbpedia.org/resource/Norway>
 - ▶ has capital: <http://dbpedia.org/ontology/capital>
 - ▶ Oslo: <http://dbpedia.org/resource/Oslo>
 - ▶ has mayor: <http://dbpedia.org/ontology/leaderName>

Resource Identifiers

- ▶ RDF (Resource Description Framework) talks about *resources*
 - ▶ Resources can be pretty much anything. People, types, relations...
- ▶ Resources are identified by IRIs (Internationalized Resource Identifiers)
- ▶ e.g. in dbpedia.org :
 - ▶ Norway: <http://dbpedia.org/resource/Norway>
 - ▶ has capital: <http://dbpedia.org/ontology/capital>
 - ▶ Oslo: <http://dbpedia.org/resource/Oslo>
 - ▶ has mayor: <http://dbpedia.org/ontology/leaderName>
 - ▶ Marianne Borgen: http://dbpedia.org/resource/Marianne_Borgen

Resource Identifiers

- ▶ RDF (Resource Description Framework) talks about *resources*
 - ▶ Resources can be pretty much anything. People, types, relations...
- ▶ Resources are identified by IRIs (Internationalized Resource Identifiers)
- ▶ e.g. in dbpedia.org :
 - ▶ Norway: <http://dbpedia.org/resource/Norway>
 - ▶ has capital: <http://dbpedia.org/ontology/capital>
 - ▶ Oslo: <http://dbpedia.org/resource/Oslo>
 - ▶ has mayor: <http://dbpedia.org/ontology/leaderName>
 - ▶ Marianne Borgen: http://dbpedia.org/resource/Marianne_Borgen
- ▶ Not necessarily dereferenceable

Resource Identifiers

- ▶ RDF (Resource Description Framework) talks about *resources*
 - ▶ Resources can be pretty much anything. People, types, relations...
- ▶ Resources are identified by IRIs (Internationalized Resource Identifiers)
- ▶ e.g. in dbpedia.org :
 - ▶ Norway: <http://dbpedia.org/resource/Norway>
 - ▶ has capital: <http://dbpedia.org/ontology/capital>
 - ▶ Oslo: <http://dbpedia.org/resource/Oslo>
 - ▶ has mayor: <http://dbpedia.org/ontology/leaderName>
 - ▶ Marianne Borgen: http://dbpedia.org/resource/Marianne_Borgen
- ▶ Not necessarily dereferenceable
 - ▶ Separate a web page from what it talks about!

Resource Identifiers

- ▶ RDF (Resource Description Framework) talks about *resources*
 - ▶ Resources can be pretty much anything. People, types, relations...
- ▶ Resources are identified by IRIs (Internationalized Resource Identifiers)
- ▶ e.g. in dbpedia.org :
 - ▶ Norway: <http://dbpedia.org/resource/Norway>
 - ▶ has capital: <http://dbpedia.org/ontology/capital>
 - ▶ Oslo: <http://dbpedia.org/resource/Oslo>
 - ▶ has mayor: <http://dbpedia.org/ontology/leaderName>
 - ▶ Marianne Borgen: http://dbpedia.org/resource/Marianne_Borgen
- ▶ Not necessarily dereferenceable
 - ▶ Separate a web page from what it talks about!
- ▶ Helps to avoid name clashes

Resource Identifiers

- ▶ RDF (Resource Description Framework) talks about *resources*
 - ▶ Resources can be pretty much anything. People, types, relations...
- ▶ Resources are identified by IRIs (Internationalized Resource Identifiers)
- ▶ e.g. in dbpedia.org :
 - ▶ Norway: <http://dbpedia.org/resource/Norway>
 - ▶ has capital: <http://dbpedia.org/ontology/capital>
 - ▶ Oslo: <http://dbpedia.org/resource/Oslo>
 - ▶ has mayor: <http://dbpedia.org/ontology/leaderName>
 - ▶ Marianne Borgen: http://dbpedia.org/resource/Marianne_Borgen
- ▶ Not necessarily dereferenceable
 - ▶ Separate a web page from what it talks about!
- ▶ Helps to avoid name clashes
- ▶ Can be used for cross-referencing

RDF Serializations

There are many serializations for the RDF data model:

RDF/XML the original W3C standard. Very complicated!

```
<?xml version="1.0"?>
<rdf:RDF xmlns:dbp="http://dbpedia.org/resource/"
           xmlns:foaf="http://xmlns.com/foaf/0.1/"
           xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    <rdf:Description rdf:about="http://dbpedia.org/resource/Marianne_Borgen">
        <foaf:name>Marianne Borgen</foaf:name>
    </rdf:Description>
</rdf:RDF>
```



RDF Serializations

There are many serializations for the RDF data model:

RDF/XML the original W3C standard. Very complicated!

```
<?xml version="1.0"?>
<rdf:RDF xmlns:dbp="http://dbpedia.org/resource/"
           xmlns:foaf="http://xmlns.com/foaf/0.1/"
           xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  <rdf:Description rdf:about="http://dbpedia.org/resource/Marianne_Borgen">
    <foaf:name>Marianne Borgen</foaf:name>
  </rdf:Description>
</rdf:RDF>
```



N-triples one triple per line. No abbreviations.

```
<http://dbpedia.org/resource/Marianne_Borgen> <http://xmlns.com/foaf/0.1/name> "Marianne Borgen".
```

RDF Serializations

There are many serializations for the RDF data model:

RDF/XML the original W3C standard. Very complicated!

```
<?xml version="1.0"?>
<rdf:RDF xmlns:dbp="http://dbpedia.org/resource/"
           xmlns:foaf="http://xmlns.com/foaf/0.1/"
           xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  <rdf:Description rdf:about="http://dbpedia.org/resource/Marianne_Borgen">
    <foaf:name>Marianne Borgen</foaf:name>
  </rdf:Description>
</rdf:RDF>
```



N-triples one triple per line. No abbreviations.

```
<http://dbpedia.org/resource/Marianne_Borgen> <http://xmlns.com/foaf/0.1/name> "Marianne Borgen".
```

Turtle/N3 convenient, human readable/writable.

```
@prefix dbp: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

dbp:Marianne_Borgen foaf:name "Marianne Borgen".
```

RDF Serializations

There are many serializations for the RDF data model:

RDF/XML the original W3C standard. Very complicated!

```
<?xml version="1.0"?>
<rdf:RDF xmlns:dbp="http://dbpedia.org/resource/"
           xmlns:foaf="http://xmlns.com/foaf/0.1/"
           xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  <rdf:Description rdf:about="http://dbpedia.org/resource/Marianne_Borgen">
    <foaf:name>Marianne Borgen</foaf:name>
  </rdf:Description>
</rdf:RDF>
```



N-triples one triple per line. No abbreviations.

```
<http://dbpedia.org/resource/Marianne_Borgen> <http://xmlns.com/foaf/0.1/name> "Marianne Borgen".
```

Turtle/N3 convenient, human readable/writable.

```
@prefix dbp: <http://dbpedia.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

dbp:Marianne_Borgen foaf:name "Marianne Borgen".
```

Others TriX, TriG, RDF/JSON, ...

QNames

- ▶ IRIs are often long and hard to type



QNames

- ▶ IRIs are often long and hard to type
- ▶ Most serializations use an abbreviation mechanism



QNames

- ▶ IRIs are often long and hard to type
- ▶ Most serializations use an abbreviation mechanism
 - ▶ Define “prefixes”, “namespaces”



QNames

- ▶ IRIs are often long and hard to type
- ▶ Most serializations use an abbreviation mechanism
 - ▶ Define “prefixes”, “namespaces”
 - ▶ RDF/XML format: XML namespaces and entities



QNames

- ▶ IRIs are often long and hard to type
- ▶ Most serializations use an abbreviation mechanism
 - ▶ Define “prefixes”, “namespaces”
 - ▶ RDF/XML format: XML namespaces and entities
- ▶ E.g. in Turtle format:

```
@prefix dbp: <http://dbpedia.org/resource/> .  
@prefix dbp-ont: <http://dbpedia.org/ontology/> .
```



QNames

- ▶ IRIs are often long and hard to type
- ▶ Most serializations use an abbreviation mechanism
 - ▶ Define “prefixes”, “namespaces”
 - ▶ RDF/XML format: XML namespaces and entities
- ▶ E.g. in Turtle format:

```
@prefix dbp: <http://dbpedia.org/resource/> .  
@prefix dbp-ont: <http://dbpedia.org/ontology/> .
```
- ▶ A *QName* like dbp:Oslo stands for <http://dbpedia.org/resource/Oslo>



QNames

- ▶ IRIs are often long and hard to type
- ▶ Most serializations use an abbreviation mechanism
 - ▶ Define “prefixes”, “namespaces”
 - ▶ RDF/XML format: XML namespaces and entities
- ▶ E.g. in Turtle format:

```
@prefix dbp: <http://dbpedia.org/resource/> .  
@prefix dbp-ont: <http://dbpedia.org/ontology/> .
```
- ▶ A *QName* like dbp:Oslo stands for <http://dbpedia.org/resource/Oslo>
- ▶ Remember: It's all just IRIs!



RDF Graphs

- ▶ An *RDF graph* is a set of triples. E.g.

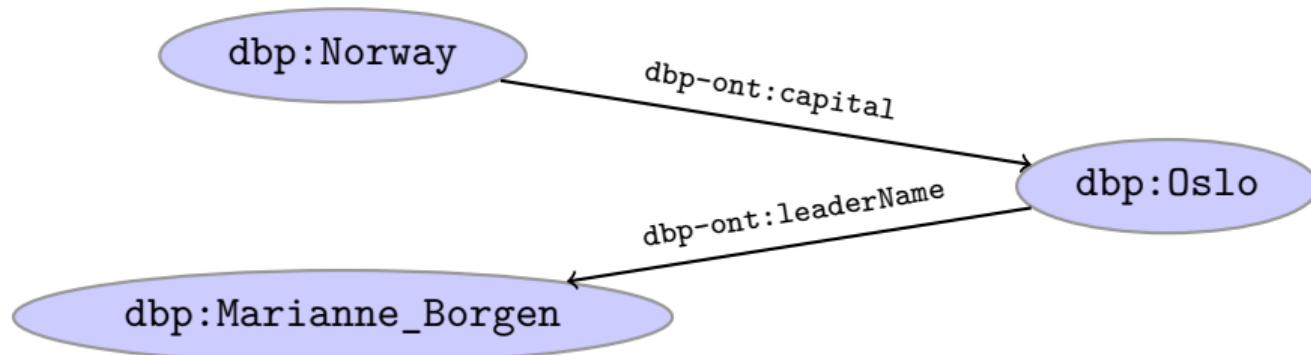
```
dbp:Norway dbp-ont:capital dbp:Oslo .  
dbp:Oslo dbp-ont:leaderName dbp:Marianne_Borgen .
```

RDF Graphs

- ▶ An *RDF graph* is a set of triples. E.g.

```
dbp:Norway dbp-ont:capital dbp:Oslo .  
dbp:Oslo dbp-ont:leaderName dbp:Marianne_Borgen .
```

- ▶ Often represented as a directed graph:



Literals

- Objects of triples can also be *literals*



Literals

- ▶ Objects of triples can also be *literals*
 - ▶ I.e. nodes in an RDF graph can be *resources* or *literals*



Literals

- ▶ Objects of triples can also be *literals*
 - ▶ I.e. nodes in an RDF graph can be *resources* or *literals*
 - ▶ Subjects and predicates of triples can *not* be literals



Literals

- ▶ Objects of triples can also be *literals*
 - ▶ I.e. nodes in an RDF graph can be *resources* or *literals*
 - ▶ Subjects and predicates of triples can *not* be literals
- ▶ Literals...



Literals

- ▶ Objects of triples can also be *literals*
 - ▶ I.e. nodes in an RDF graph can be *resources* or *literals*
 - ▶ Subjects and predicates of triples can *not* be literals
- ▶ Literals...
 - ▶ Have a datatype, identified by an IRI

```
dbp:Oslo dbp-ont:population "658390"^^xsd:integer .
```



Literals

- ▶ Objects of triples can also be *literals*
 - ▶ I.e. nodes in an RDF graph can be *resources* or *literals*
 - ▶ Subjects and predicates of triples can *not* be literals
- ▶ Literals...
 - ▶ Have a datatype, identified by an IRI

```
dbp:Oslo dbp-ont:population "658390"^^xsd:integer .
```
 - ▶ The datatype defaults to xsd:string

```
dbp:Oslo dbp-ont:officialName "Oslo".
```



Literals

- ▶ Objects of triples can also be *literals*
 - ▶ I.e. nodes in an RDF graph can be *resources* or *literals*
 - ▶ Subjects and predicates of triples can *not* be literals
- ▶ Literals...
 - ▶ Have a datatype, identified by an IRI

```
dbp:Oslo dbp-ont:population "658390"^^xsd:integer .
```
 - ▶ The datatype defaults to xsd:string

```
dbp:Oslo dbp-ont:officialName "Oslo".
```
 - ▶ Type rdf:langString is a string with language tag:

```
dbp:Norway rdfs:label "Norge"@no .  
dbp:Norway rdfs:label "Norwegen"@de .
```



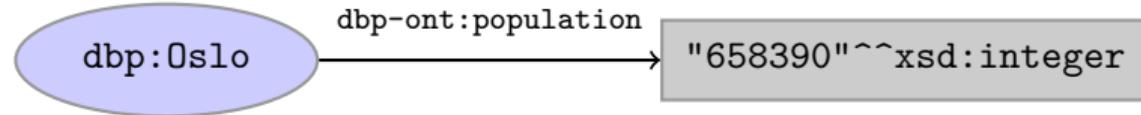
Literals

- ▶ Objects of triples can also be *literals*
 - ▶ I.e. nodes in an RDF graph can be *resources* or *literals*
 - ▶ Subjects and predicates of triples can *not* be literals
- ▶ Literals...
 - ▶ Have a datatype, identified by an IRI

```
dbp:Oslo dbp-ont:population "658390"^^xsd:integer .
```
 - ▶ The datatype defaults to xsd:string

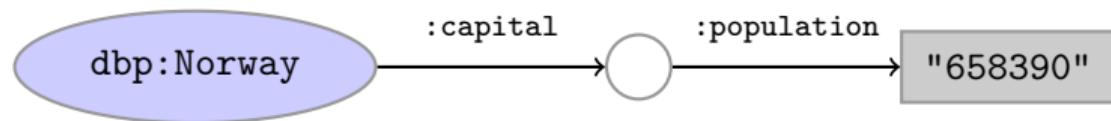
```
dbp:Oslo dbp-ont:officialName "Oslo".
```
 - ▶ Type rdf:langString is a string with language tag:

```
dbp:Norway rdfs:label "Norge"@no .  
dbp:Norway rdfs:label "Norwegen"@de .
```
- ▶ Usually represented with rectangles:

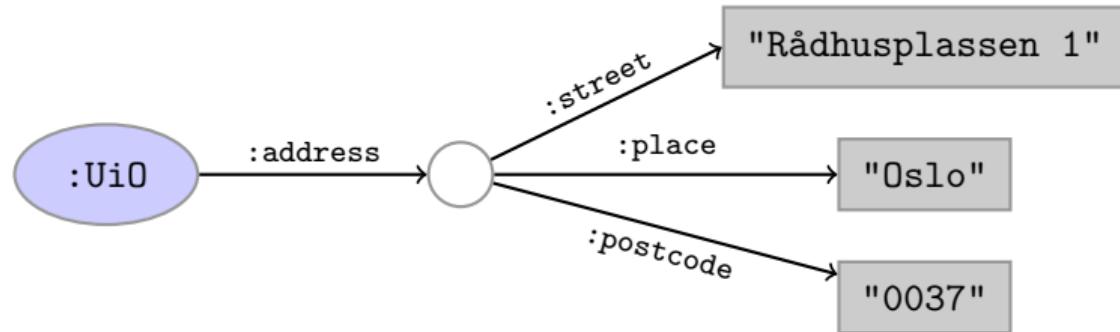


Blank nodes

- ▶ Blank nodes are like resources without a URI
- ▶ Can be used as subject or object, but not as predicate
- ▶ Norway's capital has population 658390:

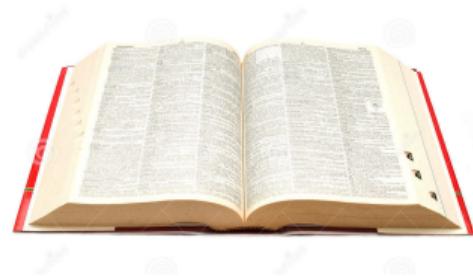


- ▶ The address of the Oslo City Hall is Rådhusplassen 1, 0037 Oslo:



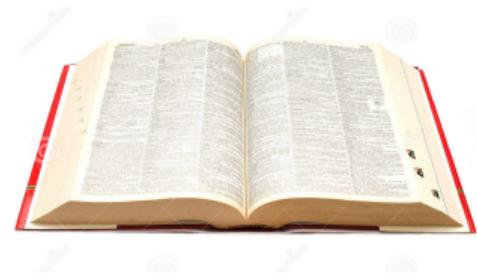
Vocabularies

- ▶ Families of related notions are grouped into *vocabularies*



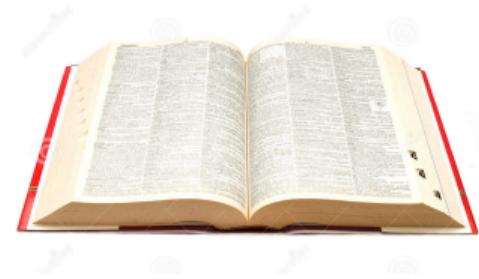
Vocabularies

- ▶ Families of related notions are grouped into *vocabularies*
- ▶ Usually the same namespace/prefix is shared



Vocabularies

- ▶ Families of related notions are grouped into *vocabularies*
- ▶ Usually the same namespace/prefix is shared
- ▶ Usually, a description is published at the namespace base URI



Vocabularies

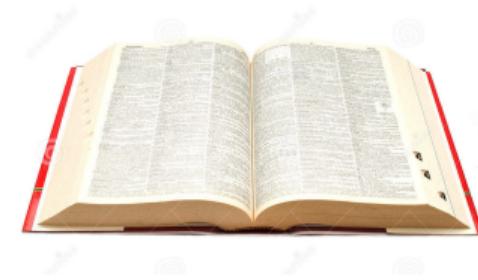
- ▶ Families of related notions are grouped into *vocabularies*
- ▶ Usually the same namespace/prefix is shared
- ▶ Usually, a description is published at the namespace base URI
- ▶ Important, well-known namespaces:

rdf: <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>> – RDF

rdfs: <<http://www.w3.org/2000/01/rdf-schema#>> – RDF Schema

dcterms: <<http://purl.org/dc/terms/>> – Dublin Core

foaf: <<http://xmlns.com/foaf/0.1/>> – Friend of a friend



Vocabularies

- ▶ Families of related notions are grouped into *vocabularies*
- ▶ Usually the same namespace/prefix is shared
- ▶ Usually, a description is published at the namespace base URI
- ▶ Important, well-known namespaces:

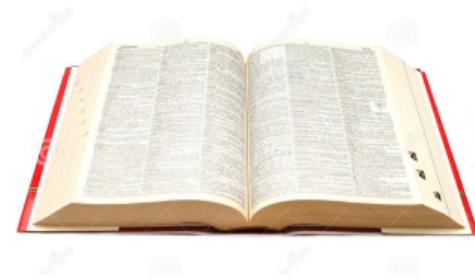
rdf: <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>> – RDF

rdfs: <<http://www.w3.org/2000/01/rdf-schema#>> – RDF Schema

dcterms: <<http://purl.org/dc/terms/>> – Dublin Core

foaf: <<http://xmlns.com/foaf/0.1/>> – Friend of a friend

- ▶ Important predicates:



Vocabularies

- ▶ Families of related notions are grouped into *vocabularies*
- ▶ Usually the same namespace/prefix is shared
- ▶ Usually, a description is published at the namespace base URI
- ▶ Important, well-known namespaces:

`rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>` – RDF

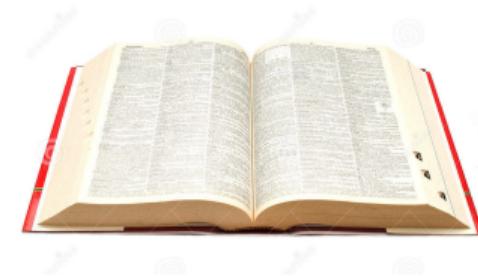
`rdfs: <http://www.w3.org/2000/01/rdf-schema#>` – RDF Schema

`dcterms: <http://purl.org/dc/terms/>` – Dublin Core

`foaf: <http://xmlns.com/foaf/0.1/>` – Friend of a friend

- ▶ Important predicates:

- ▶ `dbp:Oslo rdf:type dbp-ont:Place`



Vocabularies

- ▶ Families of related notions are grouped into *vocabularies*
- ▶ Usually the same namespace/prefix is shared
- ▶ Usually, a description is published at the namespace base URI
- ▶ Important, well-known namespaces:

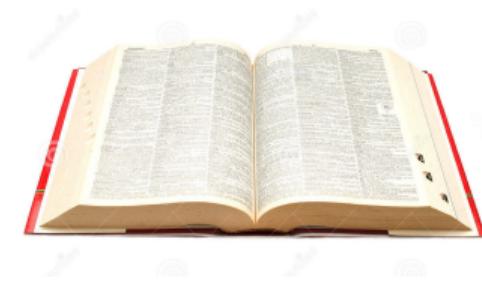
`rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>` – RDF

`rdfs: <http://www.w3.org/2000/01/rdf-schema#>` – RDF Schema

`dcterms: <http://purl.org/dc/terms/>` – Dublin Core

`foaf: <http://xmlns.com/foaf/0.1/>` – Friend of a friend

- ▶ Important predicates:
 - ▶ `dbp:Oslo rdf:type dbp-ont:Place`
 - ▶ `dbp:Norway rdfs:label "Norge"@no`



Vocabularies

- ▶ Families of related notions are grouped into *vocabularies*
- ▶ Usually the same namespace/prefix is shared
- ▶ Usually, a description is published at the namespace base URI
- ▶ Important, well-known namespaces:

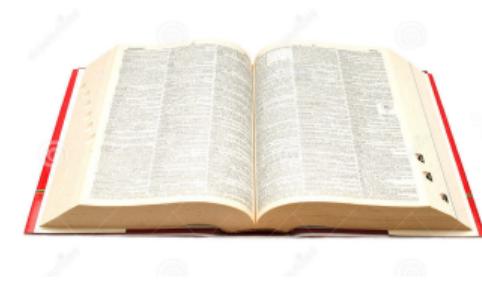
`rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>` – RDF

`rdfs: <http://www.w3.org/2000/01/rdf-schema#>` – RDF Schema

`dcterms: <http://purl.org/dc/terms/>` – Dublin Core

`foaf: <http://xmlns.com/foaf/0.1/>` – Friend of a friend

- ▶ Important predicates:
 - ▶ `dbp:Oslo rdf:type dbp-ont:Place`
 - ▶ `dbp:Norway rdfs:label "Norge"@no`
 - ▶ `:rdf-lecture dcterms:creator :martingi`



Vocabularies

- ▶ Families of related notions are grouped into *vocabularies*
- ▶ Usually the same namespace/prefix is shared
- ▶ Usually, a description is published at the namespace base URI
- ▶ Important, well-known namespaces:

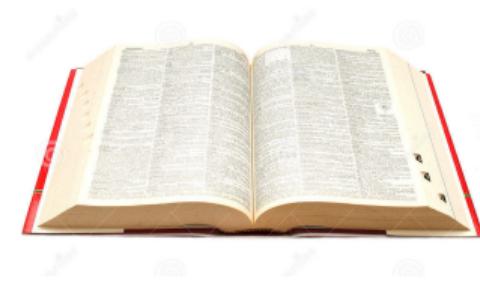
`rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>` – RDF

`rdfs: <http://www.w3.org/2000/01/rdf-schema#>` – RDF Schema

`dcterms: <http://purl.org/dc/terms/>` – Dublin Core

`foaf: <http://xmlns.com/foaf/0.1/>` – Friend of a friend

- ▶ Important predicates:
 - ▶ `dbp:Oslo rdf:type dbp-ont:Place`
 - ▶ `dbp:Norway rdfs:label "Norge"@no`
 - ▶ `:rdf-lecture dcterms:creator :martingi`
 - ▶ `dbp:Marianne_Borgen foaf:name "Marianne Borgen"`



The Turtle syntax

- ▶ “Terse RDF Triple Language”

The Turtle syntax

- ▶ “Terse RDF Triple Language”
- ▶ See: <https://www.w3.org/TR/turtle/>

The Turtle syntax

- ▶ “Terse RDF Triple Language”
- ▶ See: <https://www.w3.org/TR/turtle/>
- ▶ One of many serialization formats

The Turtle syntax

- ▶ “Terse RDF Triple Language”
- ▶ See: <https://www.w3.org/TR/turtle/>
- ▶ One of many serialization formats
- ▶ Good for human readability

Statements/assertions/triples

Full IRIs are surrounded by < and >

```
<http://dbpedia.org/resource/Oslo>
```

Statements are triples terminated by a period:

```
<http://dbpedia.org/resource/Oslo>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://dbpedia.org/ontology/Place> .
```

rdf:type may be abbreviated with 'a':

```
<http://dbpedia.org/resource/Oslo>
a
<http://dbpedia.org/ontology/Place> .
```

Namespaces

Namespace prefixes are declared with @prefix:

```
@prefix dbp: <http://dbpedia.org/resource/> .  
  
dbp:Oslo a <http://dbpedia.org/ontology/Place> .
```

A base namespace may be declared:

```
@prefix dbp: <http://dbpedia.org/resource/> .  
@prefix : <http://dbpedia.org/ontology/> .  
  
dbp:Oslo a :Place .
```

Literals

Literal values are enclosed in double quotes:

```
@prefix dbp: <http://dbpedia.org/resource/> .  
@prefix : <http://dbpedia.org/ontology/> .  
  
dbp:Oslo :officialName "Oslo" .
```

Possibly with type or language information:

```
dbp:Norway rdfs:label Norge"@no .  
dbp:Oslo :population "658390"^^xsd:integer .
```

Statements with shared subjects

Statements may share a subject with ‘;’

```
dbp:Oslo :officialName "Oslo";  
          :population "658390"^^xsd:integer ;  
          :leaderName dbp:Marianne_Borgen .
```

Statements may share subject and predicate with ‘,’

```
dbp:Norway rdfs:label "Norway"@en ,  
           "Norwegen"@de ,  
           "Norge"@no .
```

Blank nodes

Blank nodes are designated with underscores or [...]

Norway has a capital with population 658390

```
dbp:Norway :capital _:someplace .  
_:someplace :population "658390"^^xsd:integer .
```

There is a place with official name Oslo

```
[] a :Place ;  
:officialName "Oslo" .
```

UiO has address Problemveien 7, 0313 Oslo

```
:UiO :address [ :street "Problemveien 7" ;  
:place "Oslo" ;  
:postcode "0313" ] .
```

Supplementary reading—W3C specs:

- ▶ Concepts and Abstract Syntax:
 - ▶ <https://www.w3.org/TR/rdf11-concepts/>
- ▶ RDF Semantics:
 - ▶ <https://www.w3.org/TR/rdf11-mt/>
- ▶ Turtle syntax:
 - ▶ <https://www.w3.org/TR/turtle/>
- ▶ RDF/XML syntax:
 - ▶ <https://www.w3.org/TR/rdf-syntax-grammar/>
- ▶ RDF Primer (Turtle Version):
 - ▶ <https://www.w3.org/2007/02/turtle/primer/>



Selling point: Ontology-based data integration

- ▶ Use RDF triples to represent information.

Selling point: Ontology-based data integration

- ▶ Use RDF triples to represent information.
 - ▶ Lift from data level to information level

Selling point: Ontology-based data integration

- ▶ Use RDF triples to represent information.
 - ▶ Lift from data level to information level
- ▶ Use ontology (= model) to define common vocabulary

Selling point: Ontology-based data integration

- ▶ Use RDF triples to represent information.
 - ▶ Lift from data level to information level
- ▶ Use ontology (= model) to define common vocabulary
- ▶ Create mappings between the common vocabulary and what is in the data sources.

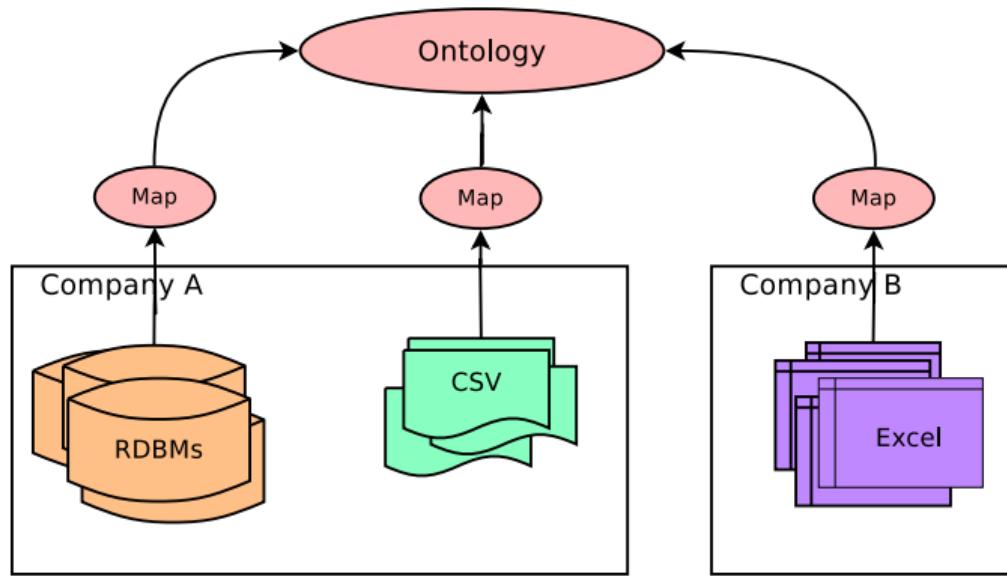
Selling point: Ontology-based data integration

- ▶ Use RDF triples to represent information.
 - ▶ Lift from data level to information level
- ▶ Use ontology (= model) to define common vocabulary
- ▶ Create mappings between the common vocabulary and what is in the data sources.
- ▶ Access data using queries expressed using the common vocabulary

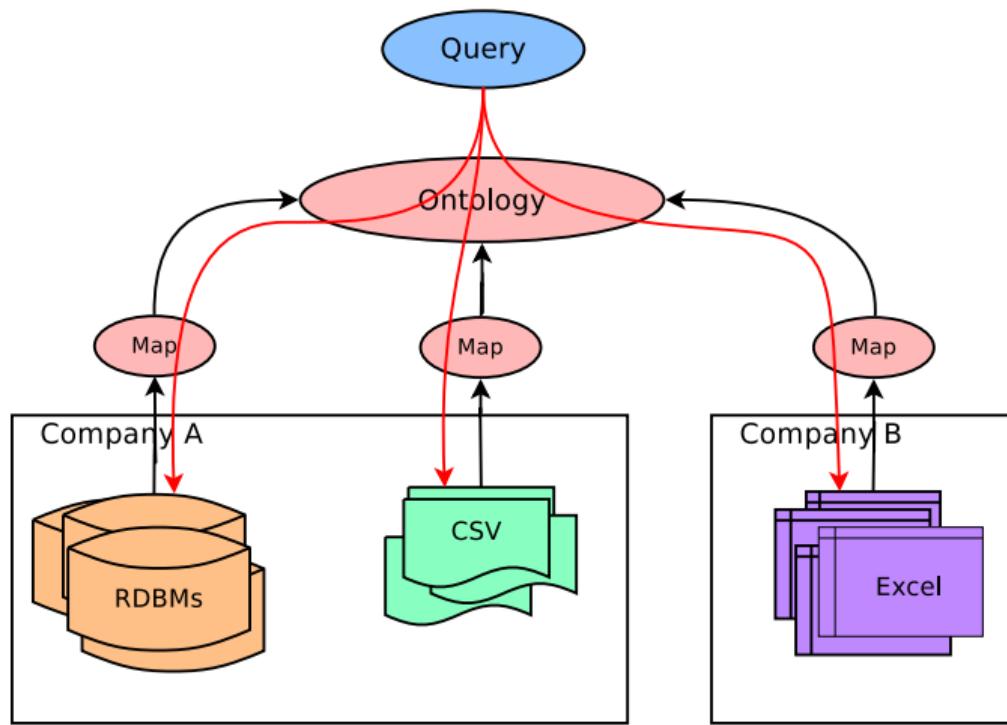
Selling point: Ontology-based data integration

- ▶ Use RDF triples to represent information.
 - ▶ Lift from data level to information level
- ▶ Use ontology (= model) to define common vocabulary
- ▶ Create mappings between the common vocabulary and what is in the data sources.
- ▶ Access data using queries expressed using the common vocabulary
- ▶ Background machinery gives answers as if data had always been stored according to a common data model

Ontology-based data integration (cont.)



Ontology-based data integration (cont.)



Virtual or Materialised?

Two main ways of implementing this:

- ▶ **Materialised RDF**: all information stored in a special RDF database, called a **triple store**
- ▶ **Virtual RDF**: information remains where it is, query gets translated using mappings, results translated back

Virtual or Materialised?

Two main ways of implementing this:

- ▶ **Materialised RDF**: all information stored in a special RDF database, called a **triple store**
- ▶ **Virtual RDF**: information remains where it is, query gets translated using mappings, results translated back

What is better? — depends. No simple answer!

Virtual or Materialised?

Two main ways of implementing this:

- ▶ **Materialised RDF**: all information stored in a special RDF database, called a **triple store**
- ▶ **Virtual RDF**: information remains where it is, query gets translated using mappings, results translated back

What is better? — depends. No simple answer!

- ▶ Example for materialised: RDFOx, Oxford, SIRIUS partner
- ▶ Example for virtual RDF: ontop, Bolzano, connected to SIRIUS

Will have tutorials on both in September!

Disclaimer: No Magic Inside

Data integration over separate sources includes many challenges:

- ▶ Identify equal concepts and relationships in data sources (schema alignment)

Disclaimer: No Magic Inside

Data integration over separate sources includes many challenges:

- ▶ Identify equal concepts and relationships in data sources (schema alignment)
 - ▶ a:Person ≡ b:Human

Disclaimer: No Magic Inside

Data integration over separate sources includes many challenges:

- ▶ Identify equal concepts and relationships in data sources (schema alignment)
 - ▶ a:Person \equiv b:Human
- ▶ Identify equal entities (entity resolution)

Disclaimer: No Magic Inside

Data integration over separate sources includes many challenges:

- ▶ Identify equal concepts and relationships in data sources (schema alignment)
 - ▶ a:Person \equiv b:Human
- ▶ Identify equal entities (entity resolution)
 - ▶ a:martingi \equiv b:GieseMartin

Disclaimer: No Magic Inside

Data integration over separate sources includes many challenges:

- ▶ Identify equal concepts and relationships in data sources (schema alignment)
 - ▶ a:Person \equiv b:Human
- ▶ Identify equal entities (entity resolution)
 - ▶ a:martingi \equiv b:GieseMartin
- ▶ Build and maintain a common ontology, and mappings

Disclaimer: No Magic Inside

Data integration over separate sources includes many challenges:

- ▶ Identify equal concepts and relationships in data sources (schema alignment)
 - ▶ a:Person \equiv b:Human
- ▶ Identify equal entities (entity resolution)
 - ▶ a:martingi \equiv b:GieseMartin
- ▶ Build and maintain a common ontology, and mappings
- ▶ Make a technical infrastructure that supports all this:

Disclaimer: No Magic Inside

Data integration over separate sources includes many challenges:

- ▶ Identify equal concepts and relationships in data sources (schema alignment)
 - ▶ a:Person \equiv b:Human
- ▶ Identify equal entities (entity resolution)
 - ▶ a:martingi \equiv b:GieseMartin
- ▶ Build and maintain a common ontology, and mappings
- ▶ Make a technical infrastructure that supports all this:
 - ▶ support asking for Humans as well as Persons

Disclaimer: No Magic Inside

Data integration over separate sources includes many challenges:

- ▶ Identify equal concepts and relationships in data sources (schema alignment)
 - ▶ a:Person \equiv b:Human
- ▶ Identify equal entities (entity resolution)
 - ▶ a:martingi \equiv b:GieseMartin
- ▶ Build and maintain a common ontology, and mappings
- ▶ Make a technical infrastructure that supports all this:
 - ▶ support asking for Humans as well as Persons
 - ▶ handle different identifiers under the hood

Disclaimer: No Magic Inside

Data integration over separate sources includes many challenges:

- ▶ Identify equal concepts and relationships in data sources (schema alignment)
 - ▶ a:Person \equiv b:Human
- ▶ Identify equal entities (entity resolution)
 - ▶ a:martingi \equiv b:GieseMartin
- ▶ Build and maintain a common ontology, and mappings
- ▶ Make a technical infrastructure that supports all this:
 - ▶ support asking for Humans as well as Persons
 - ▶ handle different identifiers under the hood
 - ▶ send queries to multiple sources and combine results

Disclaimer: No Magic Inside

Data integration over separate sources includes many challenges:

- ▶ Identify equal concepts and relationships in data sources (schema alignment)
 - ▶ a:Person \equiv b:Human
- ▶ Identify equal entities (entity resolution)
 - ▶ a:martingi \equiv b:GieseMartin
- ▶ Build and maintain a common ontology, and mappings
- ▶ Make a technical infrastructure that supports all this:
 - ▶ support asking for Humans as well as Persons
 - ▶ handle different identifiers under the hood
 - ▶ send queries to multiple sources and combine results

Sem. Tech. and Ontology Based Data Integration are a great framework

Disclaimer: No Magic Inside

Data integration over separate sources includes many challenges:

- ▶ Identify equal concepts and relationships in data sources (schema alignment)
 - ▶ a:Person \equiv b:Human
- ▶ Identify equal entities (entity resolution)
 - ▶ a:martingi \equiv b:GieseMartin
- ▶ Build and maintain a common ontology, and mappings
- ▶ Make a technical infrastructure that supports all this:
 - ▶ support asking for Humans as well as Persons
 - ▶ handle different identifiers under the hood
 - ▶ send queries to multiple sources and combine results



**Sem. Tech. and Ontology Based Data Integration are a great framework
— but not a universal solution**

Bonus: Migration

- ▶ Database schemas change
 - ▶ Fix mistakes
 - ▶ New functionality to implement
 - ▶ ...
 - ▶ Expensive and difficult to migrate data!
- ▶ RDF “schemas” (vocabularies) also change.
- ▶ But:
 - ▶ Can use old and new versions at the same time
 - ▶ Situation very much like Data integration
- ▶ Still no Magic. But Sem. Tech. can be part of a solution.

Selling point: Self-documentation

Once more: Data versus information:

Selling point: Self-documentation

Once more: Data versus information:

			...	
...	15	33	18	...
...	12	17	88	...
...	1	0	0	...
			...	

Selling point: Self-documentation

Once more: Data versus information:

...	15	33	18	...
...	12	17	88	...
...	1	0	0	...
...				

Better?

	...	Oct	Nov	Dec	...
...		
Cats	...	15	33	18	...
Dogs	...	12	17	88	...
Skunks	...	1	0	0	...
...			...		

Selling point: Self-documentation

Once more: Data versus information:

...	15	33	18	...
...	12	17	88	...
...	1	0	0	...
...				

Better?

	...	Oct	Nov	Dec	...
...			...		
Cats	...	15	33	18	...
Dogs	...	12	17	88	...
Skunks	...	1	0	0	...
...			...		

Sales? Orders? Price?

URLs Help!

Reminder: *FF9* was directed by Sam Mendes

movie:av2 movie:director people:jc.

Where `movie:av2` is just an abbreviation for

`http://heim.ifi.uio.no/martingi/movies#av2`



URLs Help!

Reminder: *FF9* was directed by Sam Mendes

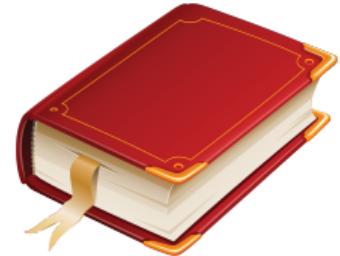
movie:av2 movie:director people:jc.

Where `movie:av2` is just an abbreviation for

<http://heim.ifi.uio.no/martingi/movies#av2>

HTTP: allows to get more information about the resource over the 'net

- ▶ about individuals (Sam Mendes, Spectre, Oslo, etc.)
- ▶ about types (Person, Movie, City, etc.)
- ▶ about relationships (has-director, has-mayor, has-inhabitants)



URLs Help!

Reminder: *FF9* was directed by Sam Mendes

movie:av2 movie:director people:jc.

Where `movie:av2` is just an abbreviation for

<http://heim.ifi.uio.no/martingi/movies#av2>

HTTP: allows to get more information about the resource over the 'net

- ▶ about individuals (Sam Mendes, Spectre, Oslo, etc.)
- ▶ about types (Person, Movie, City, etc.)
- ▶ about relationships (has-director, has-mayor, has-inhabitants)



Documentation travels with data—at a very basic level!

Example

Browse to:

http://dbpedia.org/page/Marianne_Borgen

Follow links to:

- ▶ dc:description
- ▶ foaf:name
- ▶ <http://schema.org/Person>
- ▶ <http://wikidata.org/entity/Q4572947>



Selling point: Sparse Data

	...	2012	Oct	Nov	Dec	2013	2014	...
...					...			
Baconburger	...	23						...
Cats	...		15	33	18			...
Cheeseburger	...					71		...
Dogs	...		12	17	88			...
Hamburger	...					13		...
Skunks	...		1	0	0			...
...				...				

Selling point: Sparse Data

	...	2012	Oct	Nov	Dec	2013	2014	...
...			
Baconburger	...	23	■	■	■	■	■	...
Cats	...	■	15	33	18	■	■	...
Cheeseburger	...	■	■	■	■	■	71	...
Dogs	...	■	12	17	88	■	■	...
Hamburger	...	■	■	■	■	13	■	...
Skunks	...	■	1	0	0	■	■	...
...			

- ▶ Many “holes” in the data

Selling point: Sparse Data

	...	2012	Oct	Nov	Dec	2013	2014	...
...
Baconburger	...	23	■	■	■	■	■	...
Cats	...	■	15	33	18	■	■	...
Cheeseburger	...	■	■	■	■	■	71	...
Dogs	...	■	12	17	88	■	■	...
Hamburger	...	■	■	■	■	13	■	...
Skunks	...	■	1	0	0	■	■	...
...

- ▶ Many “holes” in the data
- ▶ Typical for integration results

Selling point: Sparse Data

	...	2012	Oct	Nov	Dec	2013	2014	...
...			
Baconburger	...	23	■	■	■	■	■	...
Cats	...	■	15	33	18	■	■	...
Cheeseburger	...	■	■	■	■	■	71	...
Dogs	...	■	12	17	88	■	■	...
Hamburger	...	■	■	■	■	13	■	...
Skunks	...	■	1	0	0	■	■	...
...			

- ▶ Many “holes” in the data
- ▶ Typical for integration results
- ▶ Also in other situations: DBpedia, metadata of varying quality, etc.

The Problem with Sparse Data...

- ▶ In databases, also empty cells take space

The Problem with Sparse Data...

- ▶ In databases, also empty cells take space
- ▶ Same in Excel

The Problem with Sparse Data...

- ▶ In databases, also empty cells take space
- ▶ Same in Excel
 - ▶ Excessive memory use!

The Problem with Sparse Data...

- ▶ In databases, also empty cells take space
- ▶ Same in Excel
 - ▶ Excessive memory use!
- ▶ Coding for sparse data can be difficult and unfamiliar

The Problem with Sparse Data...

- ▶ In databases, also empty cells take space
- ▶ Same in Excel
 - ▶ Excessive memory use!
- ▶ Coding for sparse data can be difficult and unfamiliar
 - ▶ Easy to introduce errors!

The Problem with Sparse Data...

- ▶ In databases, also empty cells take space
- ▶ Same in Excel
 - ▶ Excessive memory use!
- ▶ Coding for sparse data can be difficult and unfamiliar
 - ▶ Easy to introduce errors!

- ▶ Graph Databases (e.g. triple stores) store information only where it is present



The Problem with Sparse Data...

- ▶ In databases, also empty cells take space
- ▶ Same in Excel
 - ▶ Excessive memory use!
- ▶ Coding for sparse data can be difficult and unfamiliar
 - ▶ Easy to introduce errors!

- ▶ Graph Databases (e.g. triple stores) store information only where it is present
 - ▶ Less memory usage



The Problem with Sparse Data...

- ▶ In databases, also empty cells take space
- ▶ Same in Excel
 - ▶ Excessive memory use!
- ▶ Coding for sparse data can be difficult and unfamiliar
 - ▶ Easy to introduce errors!

- ▶ Graph Databases (e.g. triple stores) store information only where it is present
 - ▶ Less memory usage
 - ▶ Coding naturally aligned with information-view



Bonus: Multiple Values

- ▶ Bonus : multiple values are as easy as no values:



Bonus: Multiple Values

- ▶ Bonus : multiple values are as easy as no values:
 - ▶ `city:o place:name "Oslo".`



Bonus: Multiple Values

- ▶ Bonus : multiple values are as easy as no values:
 - ▶ city:o place:name "Oslo".
 - ▶ city:o place:name "Christiania".



Bonus: Multiple Values

- ▶ Bonus : multiple values are as easy as no values:
 - ▶ city:o place:name "Oslo".
 - ▶ city:o place:name "Christiania".
- ▶ Useful in integration, internationalisation, etc.



Bonus: Multiple Values

- ▶ Bonus : multiple values are as easy as no values:
 - ▶ city:o place:name "Oslo".
 - ▶ city:o place:name "Christiania".
- ▶ Useful in integration, internationalisation, etc.
- ▶ Still no magic inside!



Bonus: Multiple Values

- ▶ Bonus : multiple values are as easy as no values:
 - ▶ city:o place:name "Oslo".
 - ▶ city:o place:name "Christiania".
- ▶ Useful in integration, internationalisation, etc.
- ▶ Still no magic inside!
 - ▶ Still have to find out what to do with the sparse data



Bonus: Multiple Values

- ▶ Bonus : multiple values are as easy as no values:
 - ▶ city:o place:name "Oslo".
 - ▶ city:o place:name "Christiania".
- ▶ Useful in integration, internationalisation, etc.
- ▶ Still no magic inside!
 - ▶ Still have to find out what to do with the sparse data
 - ▶ And with the multiple values



Bonus: Multiple Values

- ▶ Bonus : multiple values are as easy as no values:
 - ▶ city:o place:name "Oslo".
 - ▶ city:o place:name "Christiania".
- ▶ Useful in integration, internationalisation, etc.
- ▶ Still no magic inside!
 - ▶ Still have to find out what to do with the sparse data
 - ▶ And with the multiple values
 - ▶ Still have to code without assuming all info is there just once



Bonus: Multiple Values

- ▶ Bonus : multiple values are as easy as no values:
 - ▶ city:o place:name "Oslo".
 - ▶ city:o place:name "Christiania".
- ▶ Useful in integration, internationalisation, etc.
- ▶ Still no magic inside!
 - ▶ Still have to find out what to do with the sparse data
 - ▶ And with the multiple values
 - ▶ Still have to code without assuming all info is there just once
- ▶ If all your data is in a well-designed and stable RDB...



Bonus: Multiple Values

- ▶ Bonus : multiple values are as easy as no values:
 - ▶ city:o place:name "Oslo".
 - ▶ city:o place:name "Christiania".
- ▶ Useful in integration, internationalisation, etc.
- ▶ Still no magic inside!
 - ▶ Still have to find out what to do with the sparse data
 - ▶ And with the multiple values
 - ▶ Still have to code without assuming all info is there just once
- ▶ If all your data is in a well-designed and stable RDB...
 - ▶ ...and the tables in that RDB are mostly full...



Bonus: Multiple Values

- ▶ Bonus : multiple values are as easy as no values:
 - ▶ city:o place:name "Oslo".
 - ▶ city:o place:name "Christiania".
- ▶ Useful in integration, internationalisation, etc.
- ▶ Still no magic inside!
 - ▶ Still have to find out what to do with the sparse data
 - ▶ And with the multiple values
 - ▶ Still have to code without assuming all info is there just once
- ▶ If all your data is in a well-designed and stable RDB...
 - ▶ ...and the tables in that RDB are mostly full...
 - ▶ ...and you're happy with the code that runs on it...



Bonus: Multiple Values

- ▶ Bonus : multiple values are as easy as no values:
 - ▶ city:o place:name "Oslo".
 - ▶ city:o place:name "Christiania".
- ▶ Useful in integration, internationalisation, etc.
- ▶ Still no magic inside!
 - ▶ Still have to find out what to do with the sparse data
 - ▶ And with the multiple values
 - ▶ Still have to code without assuming all info is there just once
- ▶ If all your data is in a well-designed and stable RDB...
 - ▶ ...and the tables in that RDB are mostly full...
 - ▶ ...and you're happy with the code that runs on it...
 - ▶ ...you're probably best of keeping the data in the RDB.



Bonus: Multiple Values

- ▶ Bonus : multiple values are as easy as no values:
 - ▶ city:o place:name "Oslo".
 - ▶ city:o place:name "Christiania".
- ▶ Useful in integration, internationalisation, etc.
- ▶ Still no magic inside!
 - ▶ Still have to find out what to do with the sparse data
 - ▶ And with the multiple values
 - ▶ Still have to code without assuming all info is there just once
- ▶ If all your data is in a well-designed and stable RDB...
 - ▶ ...and the tables in that RDB are mostly full...
 - ▶ ...and you're happy with the code that runs on it...
 - ▶ ...you're probably best off keeping the data in the RDB.
 - ▶ You can still add a “virtual” semantic layer!



Exercise: RDF

- ▶ Write RDF in Turtle format
- ▶ Validate RDF using online validators
- ▶ Visualize RDF graph

SPARQL

Exercises to explore our data

Universals and Instances

Universals

- ▶ Human
- ▶ Burger shop
- ▶ Cheeseburger
- ▶ sells
- ▶ Sandstone

Instances

- ▶ Robert
- ▶ MyBurger
- ▶ cheeseburger123
- ▶ core sample 2 from 6507/10-1

Knowledge

- ▶ All humans are mammals.
- ▶ All Cheeseburgers are Hamburgers.
- ▶ Everything that sells a hamburger is a Burger Shop.

- ▶ Robert is a Human.
- ▶ Robert buys cb123.
- ▶ cb123 is a Cheeseburger.

Universals and Instances (cont.)

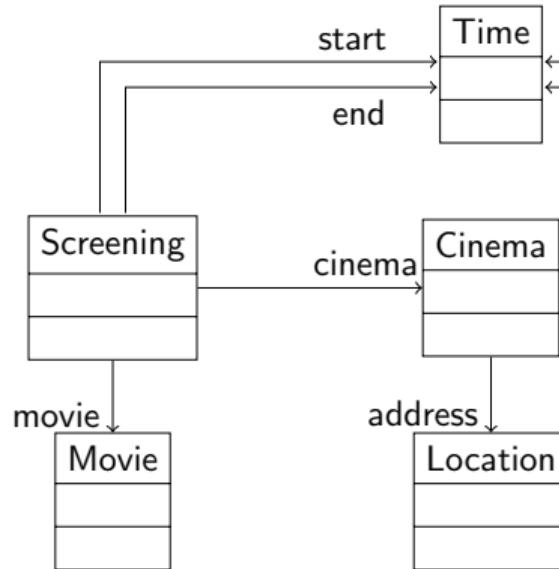
Knowledge

- ▶ All humans are mammals.
 - ▶ All Cheeseburgers are Hamburgers.
 - ▶ Everything that sells a hamburger is a Burger Shop.
 - ▶ Robert is a Human.
 - ▶ Robert buys cb123.
 - ▶ cb123 is a Cheeseburger.
-

- ▶ Conceptual model
- ▶ Ontology
- ▶ TBox
- ▶ Languages: **RDFS, OWL**
- ▶ Instance data
- ▶ Assertional knowledge
- ▶ ABox
- ▶ Language: **RDF**

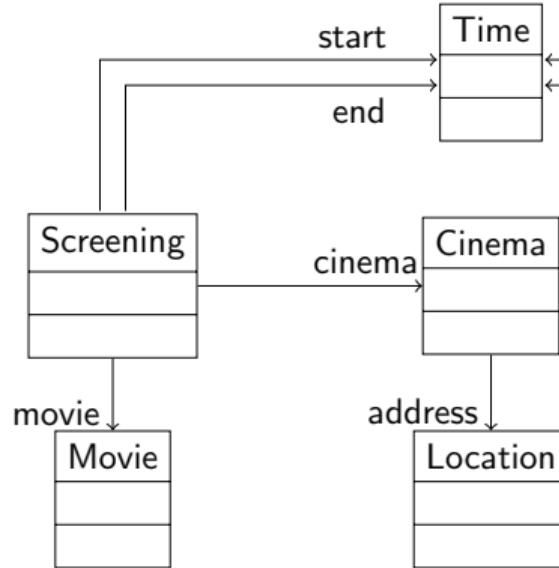
Modeling

- ▶ Conceptual Modeling, Domain Modeling



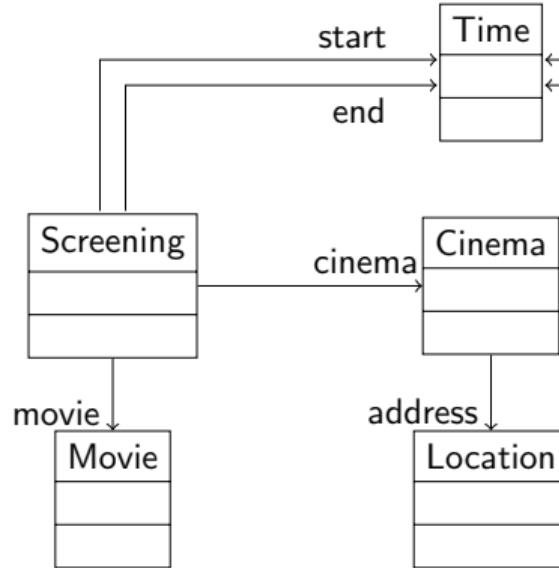
Modeling

- ▶ Conceptual Modeling, Domain Modeling
- ▶ Remember what we said about modeling in the intro...



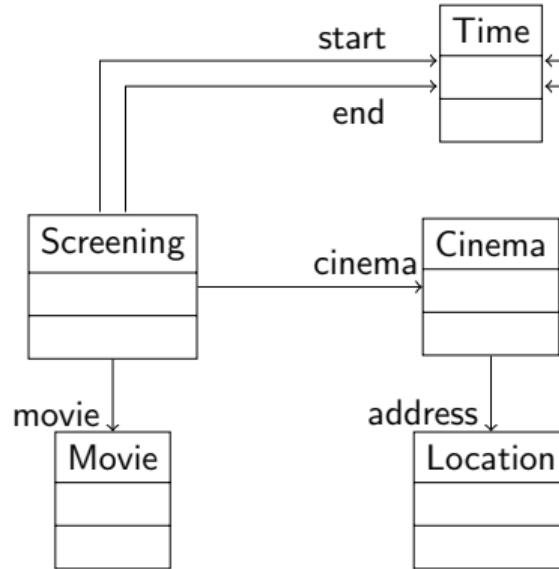
Modeling

- ▶ Conceptual Modeling, Domain Modeling
- ▶ Remember what we said about modeling in the intro...
 - ▶ Fix a vocabulary



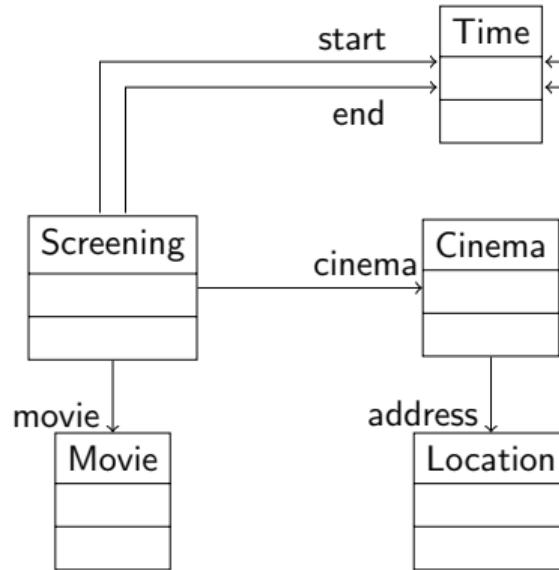
Modeling

- ▶ Conceptual Modeling, Domain Modeling
- ▶ Remember what we said about modeling in the intro...
 - ▶ Fix a vocabulary
 - ▶ Describe interaction of terms in the vocabulary



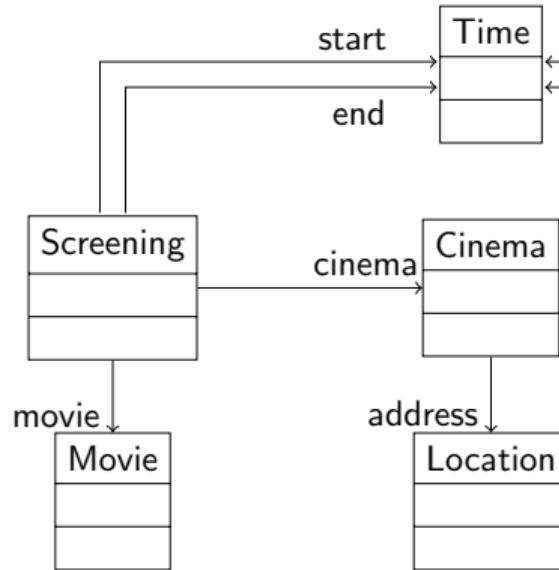
Modeling

- ▶ Conceptual Modeling, Domain Modeling
- ▶ Remember what we said about modeling in the intro...
 - ▶ Fix a vocabulary
 - ▶ Describe interaction of terms in the vocabulary
- ▶ So far, this has been like W3C-ized database technology



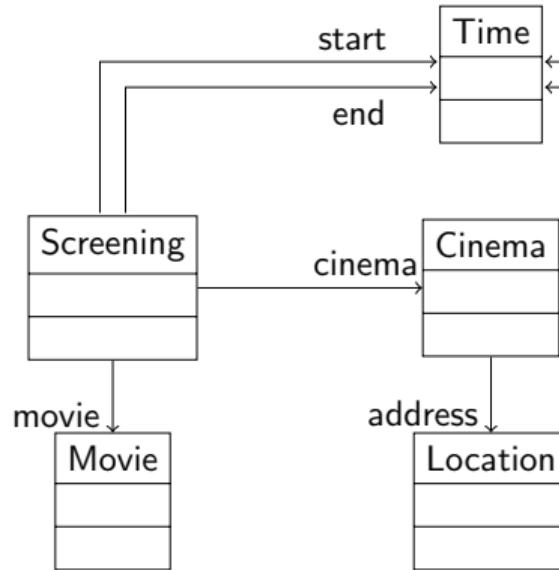
Modeling

- ▶ Conceptual Modeling, Domain Modeling
- ▶ Remember what we said about modeling in the intro...
 - ▶ Fix a vocabulary
 - ▶ Describe interaction of terms in the vocabulary
- ▶ So far, this has been like W3C-ized database technology
 - ▶ Triples instead of tables



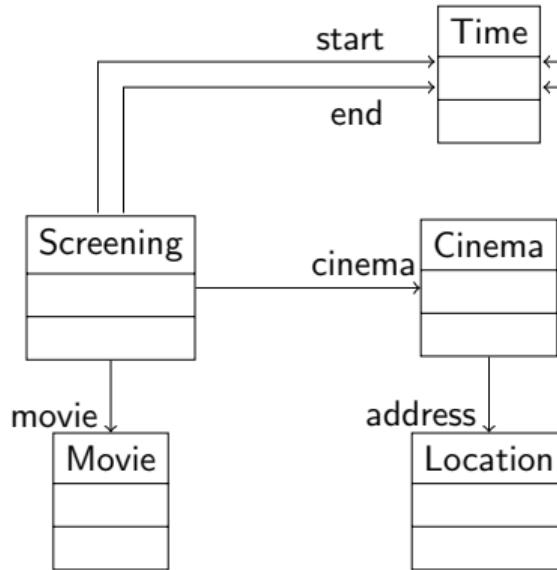
Modeling

- ▶ Conceptual Modeling, Domain Modeling
- ▶ Remember what we said about modeling in the intro...
 - ▶ Fix a vocabulary
 - ▶ Describe interaction of terms in the vocabulary
- ▶ So far, this has been like W3C-ized database technology
 - ▶ Triples instead of tables
 - ▶ SPARQL instead of SQL



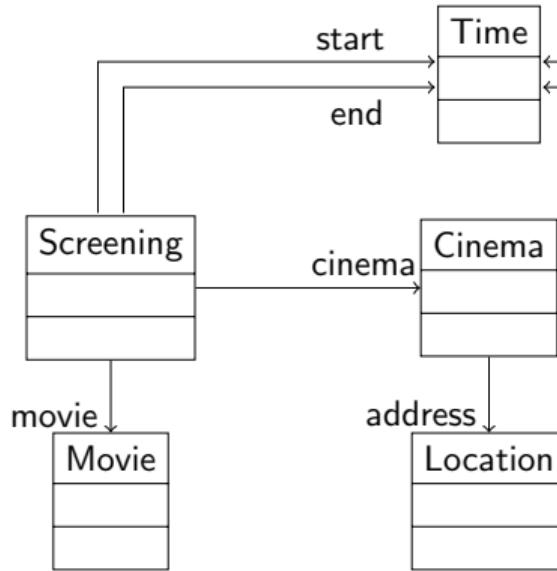
Modeling

- ▶ Conceptual Modeling, Domain Modeling
- ▶ Remember what we said about modeling in the intro...
 - ▶ Fix a vocabulary
 - ▶ Describe interaction of terms in the vocabulary
- ▶ So far, this has been like W3C-ized database technology
 - ▶ Triples instead of tables
 - ▶ SPARQL instead of SQL
- ▶ Now, we add a modeling language



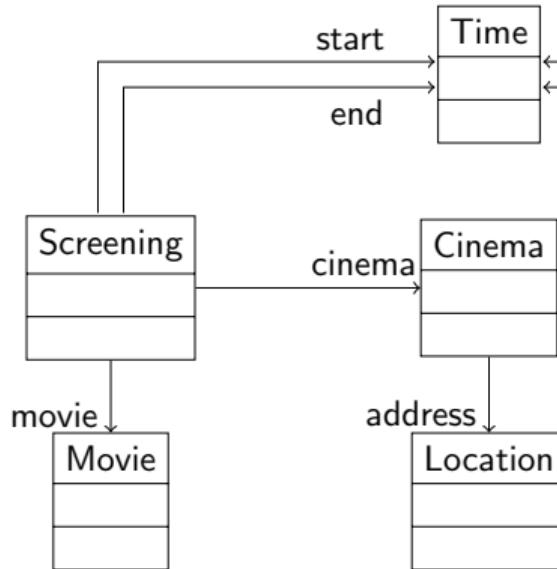
Modeling

- ▶ Conceptual Modeling, Domain Modeling
- ▶ Remember what we said about modeling in the intro...
 - ▶ Fix a vocabulary
 - ▶ Describe interaction of terms in the vocabulary
- ▶ So far, this has been like W3C-ized database technology
 - ▶ Triples instead of tables
 - ▶ SPARQL instead of SQL
- ▶ Now, we add a modeling language
 - ▶ Similar to a database schema



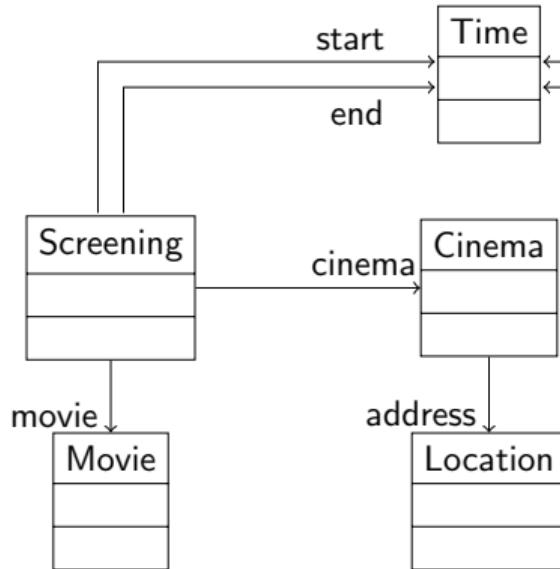
Modeling

- ▶ Conceptual Modeling, Domain Modeling
- ▶ Remember what we said about modeling in the intro...
 - ▶ Fix a vocabulary
 - ▶ Describe interaction of terms in the vocabulary
- ▶ So far, this has been like W3C-ized database technology
 - ▶ Triples instead of tables
 - ▶ SPARQL instead of SQL
- ▶ Now, we add a modeling language
 - ▶ Similar to a database schema
 - ▶ 'metadata'



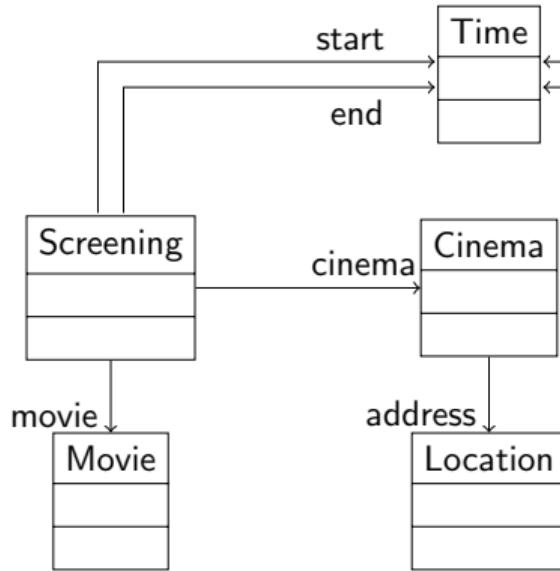
Modeling

- ▶ Conceptual Modeling, Domain Modeling
- ▶ Remember what we said about modeling in the intro...
 - ▶ Fix a vocabulary
 - ▶ Describe interaction of terms in the vocabulary
- ▶ So far, this has been like W3C-ized database technology
 - ▶ Triples instead of tables
 - ▶ SPARQL instead of SQL
- ▶ Now, we add a modeling language
 - ▶ Similar to a database schema
 - ▶ 'metadata'
 - ▶ Even more similar to UML class diagrams



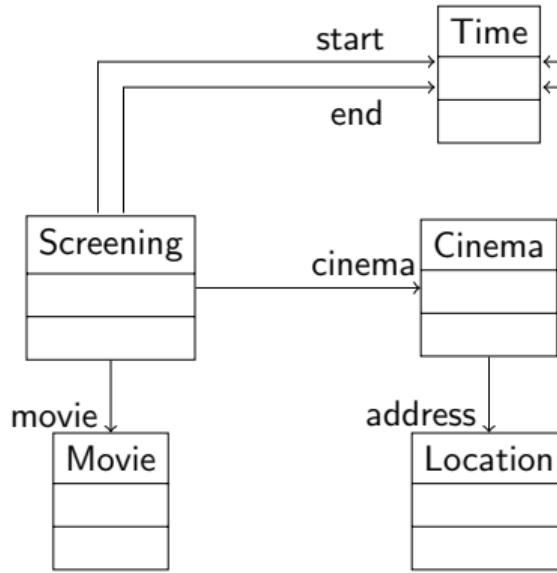
Modeling

- ▶ Conceptual Modeling, Domain Modeling
- ▶ Remember what we said about modeling in the intro...
 - ▶ Fix a vocabulary
 - ▶ Describe interaction of terms in the vocabulary
- ▶ So far, this has been like W3C-ized database technology
 - ▶ Triples instead of tables
 - ▶ SPARQL instead of SQL
- ▶ Now, we add a modeling language
 - ▶ Similar to a database schema
 - ▶ 'metadata'
 - ▶ Even more similar to UML class diagrams
 - ▶ But with functionality beyond both



Modeling

- ▶ Conceptual Modeling, Domain Modeling
- ▶ Remember what we said about modeling in the intro...
 - ▶ Fix a vocabulary
 - ▶ Describe interaction of terms in the vocabulary
- ▶ So far, this has been like W3C-ized database technology
 - ▶ Triples instead of tables
 - ▶ SPARQL instead of SQL
- ▶ Now, we add a modeling language
 - ▶ Similar to a database schema
 - ▶ ‘metadata’
 - ▶ Even more similar to UML class diagrams
 - ▶ But with functionality beyond both
 - ▶ Useful functionality to *align vocabularies*



The RDFS vocabulary

- ▶ RDFS adds the concept of “classes” which are like *types* or *sets* of resources

The RDFS vocabulary

- ▶ RDFS adds the concept of “classes” which are like *types* or *sets* of resources
- ▶ A predefined vocabulary allows statements about classes

The RDFS vocabulary

- ▶ RDFS adds the concept of “classes” which are like *types* or *sets* of resources
- ▶ A predefined vocabulary allows statements about classes
- ▶ Defined resources:

The RDFS vocabulary

- ▶ RDFS adds the concept of “classes” which are like *types* or *sets* of resources
- ▶ A predefined vocabulary allows statements about classes
- ▶ Defined resources:
 - ▶ `rdfs:Resource`: The class of resources, everything.

The RDFS vocabulary

- ▶ RDFS adds the concept of “classes” which are like *types* or *sets* of resources
- ▶ A predefined vocabulary allows statements about classes
- ▶ Defined resources:
 - ▶ `rdfs:Resource`: The class of resources, everything.
 - ▶ `rdfs:Class`: The class of classes.

The RDFS vocabulary

- ▶ RDFS adds the concept of “classes” which are like *types* or *sets* of resources
- ▶ A predefined vocabulary allows statements about classes
- ▶ Defined resources:
 - ▶ `rdfs:Resource`: The class of resources, everything.
 - ▶ `rdfs:Class`: The class of classes.
 - ▶ `rdf:Property`: The class of properties (from `rdf`)

The RDFS vocabulary

- ▶ RDFS adds the concept of “classes” which are like *types* or *sets* of resources
- ▶ A predefined vocabulary allows statements about classes
- ▶ Defined resources:
 - ▶ `rdfs:Resource`: The class of resources, everything.
 - ▶ `rdfs:Class`: The class of classes.
 - ▶ `rdf:Property`: The class of properties (from `rdf`)
- ▶ Defined properties:

The RDFS vocabulary

- ▶ RDFS adds the concept of “classes” which are like *types* or *sets* of resources
- ▶ A predefined vocabulary allows statements about classes
- ▶ Defined resources:
 - ▶ `rdfs:Resource`: The class of resources, everything.
 - ▶ `rdfs:Class`: The class of classes.
 - ▶ `rdf:Property`: The class of properties (from `rdf`)
- ▶ Defined properties:
 - ▶ `rdf:type`: relate resources to classes they are members of

The RDFS vocabulary

- ▶ RDFS adds the concept of “classes” which are like *types* or *sets* of resources
- ▶ A predefined vocabulary allows statements about classes
- ▶ Defined resources:
 - ▶ `rdfs:Resource`: The class of resources, everything.
 - ▶ `rdfs:Class`: The class of classes.
 - ▶ `rdf:Property`: The class of properties (from `rdf`)
- ▶ Defined properties:
 - ▶ `rdf:type`: relate resources to classes they are members of
 - ▶ `rdfs:domain`: The domain of a relation.

The RDFS vocabulary

- ▶ RDFS adds the concept of “classes” which are like *types* or *sets* of resources
- ▶ A predefined vocabulary allows statements about classes
- ▶ Defined resources:
 - ▶ `rdfs:Resource`: The class of resources, everything.
 - ▶ `rdfs:Class`: The class of classes.
 - ▶ `rdf:Property`: The class of properties (from `rdf`)
- ▶ Defined properties:
 - ▶ `rdf:type`: relate resources to classes they are members of
 - ▶ `rdfs:domain`: The domain of a relation.
 - ▶ `rdfs:range`: The range of a relation.

The RDFS vocabulary

- ▶ RDFS adds the concept of “classes” which are like *types* or *sets* of resources
- ▶ A predefined vocabulary allows statements about classes
- ▶ Defined resources:
 - ▶ `rdfs:Resource`: The class of resources, everything.
 - ▶ `rdfs:Class`: The class of classes.
 - ▶ `rdf:Property`: The class of properties (from `rdf`)
- ▶ Defined properties:
 - ▶ `rdf:type`: relate resources to classes they are members of
 - ▶ `rdfs:domain`: The domain of a relation.
 - ▶ `rdfs:range`: The range of a relation.
 - ▶ `rdfs:subClassOf`: Concept inclusion.

The RDFS vocabulary

- ▶ RDFS adds the concept of “classes” which are like *types* or *sets* of resources
- ▶ A predefined vocabulary allows statements about classes
- ▶ Defined resources:
 - ▶ `rdfs:Resource`: The class of resources, everything.
 - ▶ `rdfs:Class`: The class of classes.
 - ▶ `rdf:Property`: The class of properties (from `rdf`)
- ▶ Defined properties:
 - ▶ `rdf:type`: relate resources to classes they are members of
 - ▶ `rdfs:domain`: The domain of a relation.
 - ▶ `rdfs:range`: The range of a relation.
 - ▶ `rdfs:subClassOf`: Concept inclusion.
 - ▶ `rdfs:subPropertyOf`: Property inclusion.

- The meaning of this vocabulary is given through *inference rules*

- ▶ The meaning of this vocabulary is given through *inference rules*
- ▶ Given some statements, a rule allows to infer other statements

Semantics

- ▶ The meaning of this vocabulary is given through *inference rules*
- ▶ Given some statements, a rule allows to infer other statements
- ▶ Totally some 20 rules for RDFS

- ▶ The meaning of this vocabulary is given through *inference rules*
- ▶ Given some statements, a rule allows to infer other statements
- ▶ Totally some 20 rules for RDFS
- ▶ E.g. rule RDFS9:

$$\frac{C \text{ rdfs:subClassOf } D \\ x \text{ rdf:type } C}{x \text{ rdf:type } D}$$

- ▶ The meaning of this vocabulary is given through *inference rules*
- ▶ Given some statements, a rule allows to infer other statements
- ▶ Totally some 20 rules for RDFS
- ▶ E.g. rule RDFS9:

$$\frac{C \text{ rdfs:subClassOf } D}{\begin{array}{c} x \text{ rdf:type } C \\ \hline x \text{ rdf:type } D \end{array}}$$

- ▶ Example of inference:

$$\frac{\begin{array}{c} :City \text{ rdfs:subClassOf } :Place \\ :Oslo \text{ rdf:type } :City \end{array}}{:Oslo \text{ rdf:type } :Place}$$

- ▶ The meaning of this vocabulary is given through *inference rules*
- ▶ Given some statements, a rule allows to infer other statements
- ▶ Totally some 20 rules for RDFS
- ▶ E.g. rule RDFS9:

$$\frac{C \text{ rdfs:subClassOf } D}{\begin{array}{c} x \text{ rdf:type } C \\ \hline x \text{ rdf:type } D \end{array}}$$

- ▶ Example of inference:

$$\frac{\begin{array}{c} :City \text{ rdfs:subClassOf } :Place \\ :Oslo \text{ rdf:type } :City \end{array}}{:Oslo \text{ rdf:type } :Place}$$

- ▶ Asserted and inferred triples

Semantics

- ▶ The meaning of this vocabulary is given through *inference rules*
- ▶ Given some statements, a rule allows to infer other statements
- ▶ Totally some 20 rules for RDFS
- ▶ E.g. rule RDFS9:

$$\frac{C \text{ rdfs:subClassOf } D}{\begin{array}{c} x \text{ rdf:type } C \\ \hline x \text{ rdf:type } D \end{array}}$$

- ▶ Example of inference:

$$\frac{\begin{array}{c} \text{:City rdfs:subClassOf :Place} \\ \text{:Oslo rdf:type :City} \end{array}}{\text{:Oslo rdf:type :Place}}$$

- ▶ **Asserted** and inferred triples

- ▶ The meaning of this vocabulary is given through *inference rules*
- ▶ Given some statements, a rule allows to infer other statements
- ▶ Totally some 20 rules for RDFS
- ▶ E.g. rule RDFS9:

$$\frac{C \text{ rdfs:subClassOf } D}{\begin{array}{c} x \text{ rdf:type } C \\ \hline x \text{ rdf:type } D \end{array}}$$

- ▶ Example of inference:

$$\frac{\begin{array}{c} :City \text{ rdfs:subClassOf } :Place \\ :Oslo \text{ rdf:type } :City \end{array}}{:Oslo \text{ rdf:type } :Place}$$

- ▶ Asserted and **inferred** triples

Rules for Properties

- Sub-properties:

$$\frac{R \text{ rdfs:subPropertyOf } S}{\begin{array}{c} x R y \\ \hline x S y \end{array}}$$

Rules for Properties

- ▶ Sub-properties:

```
:loves rdfs:subPropertyOf foaf:knows
      :romeo :loves :juliet
-----
:romeo foaf:knows :juliet
```

- ▶ Domain:

Rules for Properties

- ▶ Sub-properties:

$$\begin{array}{c} \text{:loves rdfs:subPropertyOf foaf:knows} \\ \text{:romeo :loves :juliet} \\ \hline \text{:romeo foaf:knows :juliet} \end{array}$$

- ▶ Domain:

$$\begin{array}{c} R \text{ rdfs:domain } C \\ x R y \\ \hline x \text{ rdf:type } C \end{array}$$

- ▶ Range:

Rules for Properties

- ▶ Sub-properties:

```
:loves rdfs:subPropertyOf foaf:knows
      :romeo :loves :juliet
_____
:romeo foaf:knows :juliet
```

- ▶ Domain:

```
foaf:knows rdfs:domain foaf:Person
      :romeo foaf:knows :juliet
_____
:romeo rdf:type foaf:Person
```

- ▶ Range:

Rules for Properties

- ▶ Sub-properties:

$$\begin{array}{c} \text{:loves rdfs:subPropertyOf foaf:knows} \\ \text{:romeo :loves :juliet} \\ \hline \text{:romeo foaf:knows :juliet} \end{array}$$

- ▶ Domain:

$$\begin{array}{c} \text{foaf:knows rdfs:domain foaf:Person} \\ \text{:romeo foaf:knows :juliet} \\ \hline \text{:romeo rdf:type foaf:Person} \end{array}$$

- ▶ Range:

$$\begin{array}{c} R \text{ rdfs:range } C \\ xRy \\ \hline y \text{ rdf:type } C \end{array}$$

Rules for Properties

- ▶ Sub-properties:

```
:loves rdfs:subPropertyOf foaf:knows
      :romeo :loves :juliet
-----
:romeo foaf:knows :juliet
```

- ▶ Domain:

```
foaf:knows rdfs:domain foaf:Person
      :romeo foaf:knows :juliet
-----
:romeo rdf:type foaf:Person
```

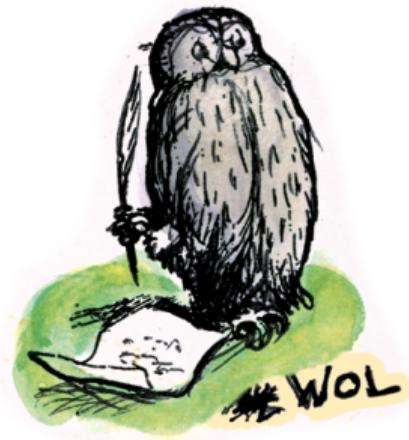
- ▶ Range:

```
foaf:knows rdfs:range foaf:Person
      :romeo foaf:knows :juliet
-----
:juliet rdf:type foaf:Person
```

OWL Quick Facts

OWL:

- ▶ The “Web Ontology Language”



OWL Quick Facts

OWL:

- ▶ The “Web Ontology Language”
- ▶ OWL is a language to express “ontologies”



OWL Quick Facts

OWL:

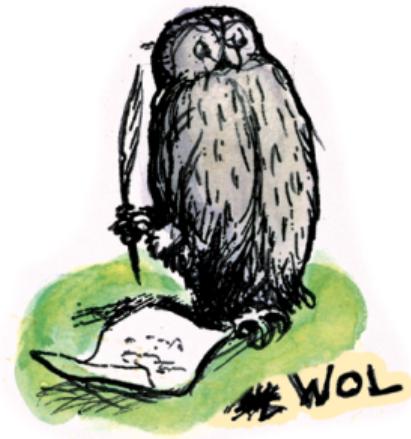
- ▶ The “Web Ontology Language”
- ▶ OWL is a language to express “ontologies”
- ▶ i.e. express facts about a domain, like RDFS



OWL Quick Facts

OWL:

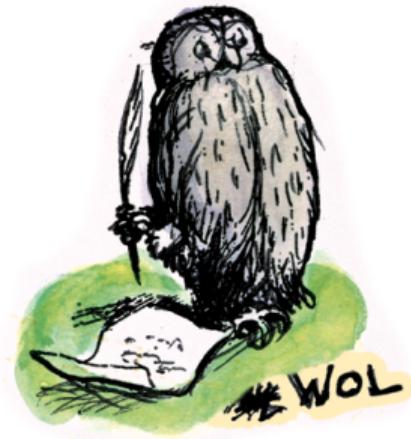
- ▶ The “Web Ontology Language”
- ▶ OWL is a language to express “ontologies”
- ▶ i.e. express facts about a domain, like RDFS
 - ▶ Extends (a subset of) RDFS



OWL Quick Facts

OWL:

- ▶ The “Web Ontology Language”
- ▶ OWL is a language to express “ontologies”
- ▶ i.e. express facts about a domain, like RDFS
 - ▶ Extends (a subset of) RDFS
 - ▶ But a much richer language than RDFS



OWL Quick Facts

OWL:

- ▶ The “Web Ontology Language”
- ▶ OWL is a language to express “ontologies”
- ▶ i.e. express facts about a domain, like RDFS
 - ▶ Extends (a subset of) RDFS
 - ▶ But a much richer language than RDFS
- ▶ Became a W3C recommendation in 2004.



OWL Quick Facts

OWL:

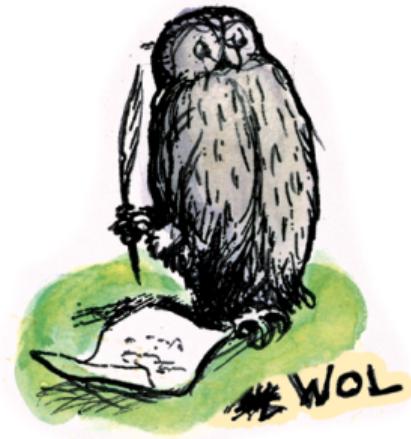
- ▶ The “Web Ontology Language”
- ▶ OWL is a language to express “ontologies”
- ▶ i.e. express facts about a domain, like RDFS
 - ▶ Extends (a subset of) RDFS
 - ▶ But a much richer language than RDFS
- ▶ Became a W3C recommendation in 2004.
- ▶ Superseded by OWL 2 in 2009.



OWL Quick Facts

OWL:

- ▶ The “Web Ontology Language”
- ▶ OWL is a language to express “ontologies”
- ▶ i.e. express facts about a domain, like RDFS
 - ▶ Extends (a subset of) RDFS
 - ▶ But a much richer language than RDFS
- ▶ Became a W3C recommendation in 2004.
- ▶ Superseded by OWL 2 in 2009.
- ▶ Built on Description Logics, separation of data and ontology



OWL Quick Facts

OWL:

- ▶ The “Web Ontology Language”
- ▶ OWL is a language to express “ontologies”
- ▶ i.e. express facts about a domain, like RDFS
 - ▶ Extends (a subset of) RDFS
 - ▶ But a much richer language than RDFS
- ▶ Became a W3C recommendation in 2004.
- ▶ Superseded by OWL 2 in 2009.
- ▶ Built on Description Logics, separation of data and ontology
- ▶ Combines DL expressiveness with RDF technology (IRIs, namespaces, etc.)



OWL Syntaxes

- ▶ Reminder: RDF is an abstract construction, several concrete syntaxes: RDF/XML, Turtle,...



OWL Syntaxes

- ▶ Reminder: RDF is an abstract construction, several concrete syntaxes: RDF/XML, Turtle,...
- ▶ Same for OWL:



OWL Syntaxes

- ▶ Reminder: RDF is an abstract construction, several concrete syntaxes: RDF/XML, Turtle,...
- ▶ Same for OWL:
- ▶ Defined as a set of things that can be said about classes, properties, instances



OWL Syntaxes

- ▶ Reminder: RDF is an abstract construction, several concrete syntaxes: RDF/XML, Turtle,...
- ▶ Same for OWL:
- ▶ Defined as a set of things that can be said about classes, properties, instances
- ▶ OWL/RDF: Uses RDF to express OWL ontologies



OWL Syntaxes

- ▶ Reminder: RDF is an abstract construction, several concrete syntaxes: RDF/XML, Turtle,...
- ▶ Same for OWL:
- ▶ Defined as a set of things that can be said about classes, properties, instances
- ▶ OWL/RDF: Uses RDF to express OWL ontologies
 - ▶ Then use any of the RDF serializations



OWL Syntaxes

- ▶ Reminder: RDF is an abstract construction, several concrete syntaxes: RDF/XML, Turtle,...
- ▶ Same for OWL:
- ▶ Defined as a set of things that can be said about classes, properties, instances
- ▶ OWL/RDF: Uses RDF to express OWL ontologies
 - ▶ Then use any of the RDF serializations
- ▶ OWL/XML: a non-RDF XML format



OWL Syntaxes

- ▶ Reminder: RDF is an abstract construction, several concrete syntaxes: RDF/XML, Turtle,...
- ▶ Same for OWL:
- ▶ Defined as a set of things that can be said about classes, properties, instances
- ▶ OWL/RDF: Uses RDF to express OWL ontologies
 - ▶ Then use any of the RDF serializations
- ▶ OWL/XML: a non-RDF XML format
- ▶ Functional OWL syntax: simple, used in definition



OWL Syntaxes

- ▶ Reminder: RDF is an abstract construction, several concrete syntaxes: RDF/XML, Turtle,...
- ▶ Same for OWL:
- ▶ Defined as a set of things that can be said about classes, properties, instances
- ▶ OWL/RDF: Uses RDF to express OWL ontologies
 - ▶ Then use any of the RDF serializations
- ▶ OWL/XML: a non-RDF XML format
- ▶ Functional OWL syntax: simple, used in definition
- ▶ Manchester OWL syntax: textual format used in some tools



OWL Concept Descriptions

- ▶ In OWL, classes can be combined to form new concepts

OWL Concept Descriptions

- ▶ In OWL, classes can be combined to form new concepts
- ▶ Boolean combinations:

OWL Concept Descriptions

- ▶ In OWL, classes can be combined to form new concepts
- ▶ Boolean combinations:
 - ▶ (foaf:Person **and** **not** gender:Female) **or** rodents:Mouse

OWL Concept Descriptions

- ▶ In OWL, classes can be combined to form new concepts
- ▶ Boolean combinations:
 - ▶ (foaf:Person and not gender:Female) or rodents:Mouse
 - ▶ All things which are either persons and not female or mice



OWL Concept Descriptions

- ▶ In OWL, classes can be combined to form new concepts
- ▶ Boolean combinations:
 - ▶ (foaf:Person and not gender:Female) or rodents:Mouse
 - ▶ All things which are either persons and not female or mice
- ▶ Existential restrictions:



OWL Concept Descriptions

- ▶ In OWL, classes can be combined to form new concepts
- ▶ Boolean combinations:
 - ▶ (foaf:Person and not gender:Female) or rodents:Mouse
 - ▶ All things which are either persons and not female or mice
- ▶ Existential restrictions:
 - ▶ foaf:knows some gender:Female



OWL Concept Descriptions

- ▶ In OWL, classes can be combined to form new concepts
- ▶ Boolean combinations:
 - ▶ (foaf:Person and not gender:Female) or rodents:Mouse
 - ▶ All things which are either persons and not female or mice
- ▶ Existential restrictions:
 - ▶ foaf:knows some gender:Female
 - ▶ All things which know some (at least one) female being



OWL Concept Descriptions

- ▶ In OWL, classes can be combined to form new concepts
- ▶ Boolean combinations:
 - ▶ (foaf:Person and not gender:Female) or rodents:Mouse
 - ▶ All things which are either persons and not female or mice
- ▶ Existential restrictions:
 - ▶ foaf:knows some gender:Female
 - ▶ All things which know some (at least one) female being
- ▶ Universal restrictions:



OWL Concept Descriptions

- ▶ In OWL, classes can be combined to form new concepts
- ▶ Boolean combinations:
 - ▶ (foaf:Person and not gender:Female) or rodents:Mouse
 - ▶ All things which are either persons and not female or mice
- ▶ Existential restrictions:
 - ▶ foaf:knows some gender:Female
 - ▶ All things which know some (at least one) female being
- ▶ Universal restrictions:
 - ▶ :ownsCar only cars:Porsche



OWL Concept Descriptions

- ▶ In OWL, classes can be combined to form new concepts
- ▶ Boolean combinations:
 - ▶ `(foaf:Person and not gender:Female) or rodents:Mouse`
 - ▶ All things which are either persons and not female or mice
- ▶ Existential restrictions:
 - ▶ `foaf:knows some gender:Female`
 - ▶ All things which know some (at least one) female being
- ▶ Universal restrictions:
 - ▶ `:ownsCar only cars:Porsche`
 - ▶ All things which own *only* Porsche cars (if any...)



OWL Concept Descriptions

- ▶ In OWL, classes can be combined to form new concepts
- ▶ Boolean combinations:
 - ▶ `(foaf:Person and not gender:Female) or rodents:Mouse`
 - ▶ All things which are either persons and not female or mice
- ▶ Existential restrictions:
 - ▶ `foaf:knows some gender:Female`
 - ▶ All things which know some (at least one) female being
- ▶ Universal restrictions:
 - ▶ `:ownsCar only cars:Porsche`
 - ▶ All things which own *only* Porsche cars (if any...)
- ▶ And quite a few more.



OWL Axioms

- ▶ Concept descriptions don't say anything – they are not statements

OWL Axioms

- ▶ Concept descriptions don't say anything – they are not statements
- ▶ There are essentially two kinds of things you can say in OWL:

OWL Axioms

- ▶ Concept descriptions don't say anything – they are not statements
- ▶ There are essentially two kinds of things you can say in OWL:
- ▶ Concept membership

OWL Axioms

- ▶ Concept descriptions don't say anything – they are not statements
- ▶ There are essentially two kinds of things you can say in OWL:
- ▶ Concept membership
 - ▶ A resource belongs to a concept (`rdf:type`)

OWL Axioms

- ▶ Concept descriptions don't say anything – they are not statements
- ▶ There are essentially two kinds of things you can say in OWL:
- ▶ Concept membership
 - ▶ A resource belongs to a concept (`rdf:type`)
 - ▶ `:martingi rdf:type not (:ownsCar only :Porsche)`.

OWL Axioms

- ▶ Concept descriptions don't say anything – they are not statements
- ▶ There are essentially two kinds of things you can say in OWL:
- ▶ Concept membership
 - ▶ A resource belongs to a concept (`rdf:type`)
 - ▶ `:martingi rdf:type not (:ownsCar only :Porsche)`.
- ▶ Concept subsumption

OWL Axioms

- ▶ Concept descriptions don't say anything – they are not statements
- ▶ There are essentially two kinds of things you can say in OWL:
- ▶ Concept membership
 - ▶ A resource belongs to a concept (`rdf:type`)
 - ▶ `:martingi rdf:type not (:ownsCar only :Porsche)`.
- ▶ Concept subsumption
 - ▶ Everything in one concept belongs to another concept (`rdfs:subClassOf`)

OWL Axioms

- ▶ Concept descriptions don't say anything – they are not statements
- ▶ There are essentially two kinds of things you can say in OWL:
- ▶ Concept membership
 - ▶ A resource belongs to a concept (`rdf:type`)
 - ▶ `:martingi rdf:type not (:ownsCar only :Porsche)`.
- ▶ Concept subsumption
 - ▶ Everything in one concept belongs to another concept (`rdfs:subClassOf`)
 - ▶ `gender:Male rdfs:subClassOf (not gender:Female)`

OWL Axioms

- ▶ Concept descriptions don't say anything – they are not statements
- ▶ There are essentially two kinds of things you can say in OWL:
- ▶ Concept membership
 - ▶ A resource belongs to a concept (`rdf:type`)
 - ▶ `:martingi rdf:type not (:ownsCar only :Porsche)`.
- ▶ Concept subsumption
 - ▶ Everything in one concept belongs to another concept (`rdfs:subClassOf`)
 - ▶ `gender:Male rdfs:subClassOf (not gender:Female)`
- ▶ Subsumption both ways: equivalence

OWL Axioms

- ▶ Concept descriptions don't say anything – they are not statements
- ▶ There are essentially two kinds of things you can say in OWL:
- ▶ Concept membership
 - ▶ A resource belongs to a concept (`rdf:type`)
 - ▶ `:martingi rdf:type not (:ownsCar only :Porsche)`.
- ▶ Concept subsumption
 - ▶ Everything in one concept belongs to another concept (`rdfs:subClassOf`)
 - ▶ `gender:Male rdfs:subClassOf (not gender:Female)`
- ▶ Subsumption both ways: equivalence
 - ▶ Can be used to *define* terms

OWL Axioms

- ▶ Concept descriptions don't say anything – they are not statements
- ▶ There are essentially two kinds of things you can say in OWL:
- ▶ Concept membership
 - ▶ A resource belongs to a concept (`rdf:type`)
 - ▶ `:martingi rdf:type not (:ownsCar only :Porsche)`.
- ▶ Concept subsumption
 - ▶ Everything in one concept belongs to another concept (`rdfs:subClassOf`)
 - ▶ `gender:Male rdfs:subClassOf (not gender:Female)`
- ▶ Subsumption both ways: equivalence
 - ▶ Can be used to *define* terms
 - ▶ `:Woman owl:equivalentClass (foaf:Person and gender:Female)`

OWL Axioms

- ▶ Concept descriptions don't say anything – they are not statements
- ▶ There are essentially two kinds of things you can say in OWL:
- ▶ Concept membership
 - ▶ A resource belongs to a concept (`rdf:type`)
 - ▶ `:martingi rdf:type not (:ownsCar only :Porsche)`.
- ▶ Concept subsumption
 - ▶ Everything in one concept belongs to another concept (`rdfs:subClassOf`)
 - ▶ `gender:Male rdfs:subClassOf (not gender:Female)`
- ▶ Subsumption both ways: equivalence
 - ▶ Can be used to *define* terms
 - ▶ `:Woman owl:equivalentClass (foaf:Person and gender:Female)`
 - ▶ `:Mother owl:equivalentClass (:Woman and (:hasChild some foaf:Person))`

Demo: Using Protégé

- Create a Car class
- Create an Axle class
- Create FrontAxle and RearAxle as subclasses
- Make the axle classes disjoint
- Add a driveAxle object property
- Add domain Car and range Axle
- Add 2CV, subclass of Car
- Add superclass driveAxle only FrontAxle
- Add Lotus, subclass of Car
- Add superclass driveAxle only RearAxle
- Add LandRover, subclass of Car
- Add superclass driveAxle some FrontAxle
- Add superclass driveAxle some RearAxle
- Add 4WD as subclass of Thing
- Make equivalent to driveAxle some RearAxle and driveAxle some FrontAxle
- Classify.
- Show inferred class hierarchy: Car ⊑ 4WD ⊑ LandRover
- Tell story of 2CV Sahara, which is a 2CV with two motors, one front, one back
- Add Sahara as subclass of 2CV
- Add 4WD as superclass of Sahara
- Classify.
- Show that Sahara is equivalent to bottom.
- Explain why. In particular, disjointness of front and rear axles

Protégé Recap

- ▶ Almost like using an OO modeling tool
- ▶ Remember: In the end it's
 - ▶ OWL concept descriptions
 - ▶ `rdf:type`
 - ▶ `rdfs:subClassOf`
- ▶ Many ways of saying things in OWL, more in Protégé

Topics Covered

- ▶ Some overview and general selling points

Topics Covered

- ▶ Some overview and general selling points
- ▶ RDF principles

Topics Covered

- ▶ Some overview and general selling points
- ▶ RDF principles
- ▶ Basics of the RDFS and OWL ontology languages

Topics *Not* Covered

- ▶ The SPARQL query language

Topics *Not* Covered

- ▶ The SPARQL query language
- ▶ Lots of details in OWL

Topics *Not* Covered

- ▶ The SPARQL query language
- ▶ Lots of details in OWL
- ▶ Integrity constraints: SHACL, ShEx, etc.

Topics *Not* Covered

- ▶ The SPARQL query language
- ▶ Lots of details in OWL
- ▶ Integrity constraints: SHACL, ShEx, etc.
- ▶ Rule Languages (SWRL, datalog, etc.)

Topics Not Covered

- ▶ The SPARQL query language
- ▶ Lots of details in OWL
- ▶ Integrity constraints: SHACL, ShEx, etc.
- ▶ Rule Languages (SWRL, datalog, etc.)
- ▶ Mathematical semantics, logical theory: Soundness, Completeness, ...

That's all Folks!