CM30174 + CM50206
Intelligent Agents

Marina De Vos, Julian Padget

Communication and Ontologies / version 0.3

October 18, 2011

Authors/Credits for this lecture

- Chs. 6, 7, 8 of "An Introduction to Multiagent Systems" [Wooldridge, 2009].
- "Ontology Engineering" tutorial by Natalya Noy at the Semantic Web Working Symposium 2001.
- "Agents and the Semantic Web" tutorial by Terry Payne and Valentina Tamma at CEEMAS 2005.
- "RDF briefing" presentation by Frank van Harmelen. See `http://ubp.l3s.uni-hannover.de/ubp`.
- "A Semantic Web Primer" Grigoris Antoniou and Frank van Harmelen. See `http://www.ics.forth.gr/isl/swprimer/`

Content

Motivation

- Agents and MAS emerged from Distributed AI
  - Distribute problem-solving across several processes or machines
  - Coordination implies a need to:
    - Communicate
    - Plan
    - Coordinate actions
- Agents emerged as self-contained, autonomous entities that could perform (multiple) services
- Open Agent Systems
  - MAS developed by different organizations should interoperate
  - Only works when all the agents conform to the same MAS
  - ... not so open architecture

Agent Communication

Focus here is on macro aspects of intelligent agent technology: those issues relating to the agent society, rather than the individual agent:

- communication: speech acts; KQML & KIF; FIPA ACL.
- reaching agreements: kinds of auctions, negotiation, task-oriented domains
- cooperation: what is cooperation, cooperative versus non-cooperative encounters, the contract net protocol

Speech Acts 1/5

- Most treatments of communication in (multi-)agent systems take inspiration from speech act theory.
- Speech act theories are pragmatic theories of language, i.e., theories of language use: they attempt to account for how language is used by people every day to achieve their goals and intentions.
- The origin of speech act theories are usually traced to Austin's 1962 book, How to Do Things with Words.

Speech Acts 2/5

Austin noticed that some utterances are rather like 'physical actions' that appear to change the state of the world.

- Paradigm examples would be:
  - declaring war
  - baptism
  - 'I now pronounce you man and wife'
- In fact, everything is said with the intention of satisfying some goal or intention.
- Speech Act theory attempts to explain how utterances may achieve intentions.

Speech Acts 3/5

Searle (1969) identified various different types of speech act:

- representatives: such as informing,
  e.g., 'It is raining'
- directives: attempts to get the hearer to do something
  e.g., 'please make the tea'
- commisives: which commit the speaker to doing something,
  e.g., 'I promise to... '
- expressives: whereby a speaker expresses a mental state,
  e.g., 'thank you!'
- declarations: such as declaring war or baptism.

Speech Acts 4/5

There is some debate about whether this (or any!) typology of speech acts is appropriate.

- In general, a speech act can be seen to have two components:
  - a performative verb: (e.g., request, inform, . . .)
  - propositional content: (e.g., “the door is closed”) constructed from
    - a (formal) language, defining syntactic structures
    - an ontology, defining the concepts

These are the key observations as far as agent communication is concerned.

Speech Acts 5/5

Consider:

- performative = request
  content = “the door is closed”
  speech act = “please close the door”
- performative = inform
  content = “the door is closed”
  speech act = “the door is closed!”
- performative = inquire
  content = “the door is closed”
  speech act = “is the door closed?”

to see how the same content combined with different performatives takes on different meanings.

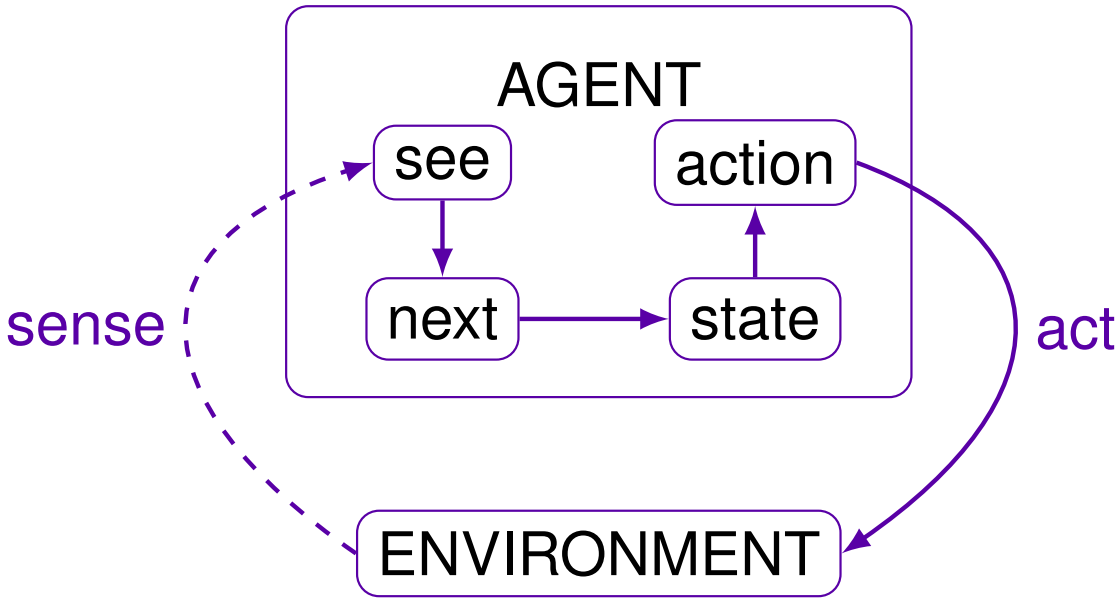
Plan-Based Semantics

Cohen & Perrault (1979) defined semantics of speech acts using the precondition/delete/add list formalism of planning research. Example: $request(s, h, \phi)$

- preconditions
  - s believes h can do ϕ
    you don't ask someone to do something unless you think they can do it
  - s believes h believes h can do ϕ
    you don't ask someone unless they believe they can do it
  - s believes s wants ϕ
    you don't ask someone unless you want it!
- postconditions:
  - h believes s believes s want ϕ
    the effect is to make them aware of your desire

BDI connection

- Speech acts can be delivered as *percepts* — introduction to agent architectures



- Likewise *percepts* for practical reasoning agents (BDI)
- BDI agents are plan-driven — thus realizing Cohen-Perrault model

Content

KQML and KIF

- ACLs: standard formats for the exchange of messages.
- ARPA knowledge sharing initiative (1990-1994)
- KQML: knowledge query and manipulation language
- 'outer' language, that defines 'communicative verbs', or performatives. Example performatives are:
  - ask-if ('is it true that... ')
  - perform ('please perform the following action... ')
  - tell ('it is true that... ')
  - reply ('the answer is ... ')
- KIF: knowledge interchange format
- 'inner' language for expressing message content.

FIPA ACL

- FIPA: second generation, simpler (1998-2002)
- FIPA's agent communication language is probably the most widely used now.
- Basic structure is quite similar to KQML:
  - performative: 20 performatives in FIPA.
  - `inform` and `request` are the two basic performatives: the rest are macros
  - housekeeping: e.g., sender, receiver etc.
  - content: the actual content of the message.
- Example:

```
1  (inform
2    :sender   agent1
3    :receiver agent5
4    :content  (price good200 150)
5    :language sl
6    :ontology hpl-auction
7  )
```

The FIPA Performatives

performative	passing information	requesting information	negotiation	performing actions	error handling
accept-proposal			x		
agree				x	
cancel		x	x		
cfp			x		
confirm	x				
disconfirm	x				
failure					x
inform	x				
inform-if	x				
inform-ref	x				
not-understood					x
propose			x		
query-if		x			
query-ref		x			
refuse				x	
reject-proposal			x		
request				x	
request-when				x	
request-whenever				x	
subscribe		x			

Inform and Request Semantics

- Semantics defined in two parts:
  - pre-condition: what must be true for the speech act to succeed. c.f. Cohen and Perrault.
  - “rational effect” what the sender of the message hopes to bring about.
- “inform”: content is a statement, and sender:
  - Holds that the content is true
  - Intends that the recipient believe the content
  - Does not already believe that the recipient is aware of whether content is true or not.
- “request”: content is an action, and sender:
  - Intends action content to be performed
  - Believes recipient is capable of performing this action
  - Does not believe that recipient already intends to perform action.

Representing Messages

- Agents use a combination of
  - agent communication language—defines message structure
  - performative, e.g. `inform`, `request` (FIPA, KQML)
  - content language—e.g. first order logic + concepts (ontology)
- Why this structure?
  - Sender and receiver have been designed and built at different times by different people—yet they have to interoperate
  - Sender and receiver must be protected from each other
  - Communications may have to be verifiable by third-parties

Content

Need for Ontologies

In order to be able to communicate, agents must have agreed a common set of terms.

- An ontology is a formal specification of a set of terms.

  Grüber (1993): "Formal, explicit specifications of a shared conceptualisation"

What is an ontology?

Depends on subject and use, but common features are:

- A formal description of (the relevant parts) of a domain: *"the nature of things, and the relationships between them"*
- A set of classes (concepts) and their hierarchical relations
- A set of properties (slots or roles), defining arbitrary relations
- The same property may be ascribed to several independent classes
- Constraints — restrictions on properties (type, number)
- Individuals — some concrete instances of classes

Example ontology

- Wordnet is a live domain-neutral ontology:

  `http://wordnetweb.princeton.edu/perl/webwn`

- Words are nodes in a network of relationships, for example:
  - hyponym: more specialized concepts
  - meronym: parts of this concept
  - hypernym: generalized concept
  - holonym: part of something larger
  - etc.

Kinds of ontology

- Degrees of formality:
  - Controlled vocabularies
  - Glossaries
  - Thesauri
  - Informal Is-a hierarchy
  - Formal Is-a hierarchy
  - Formal instances
  - Frames
  - Value restriction
  - General logic constraints

What Is "Ontology Engineering"?

Ontology Engineering: Defining terms in the domain and relations among them

- Defining concepts in the domain (classes)
- Arranging the concepts in a hierarchy (subclass-superclass hierarchy)
- Defining which attributes and properties (slots) classes can have and constraints on their values
- Defining individuals and filling in slot values

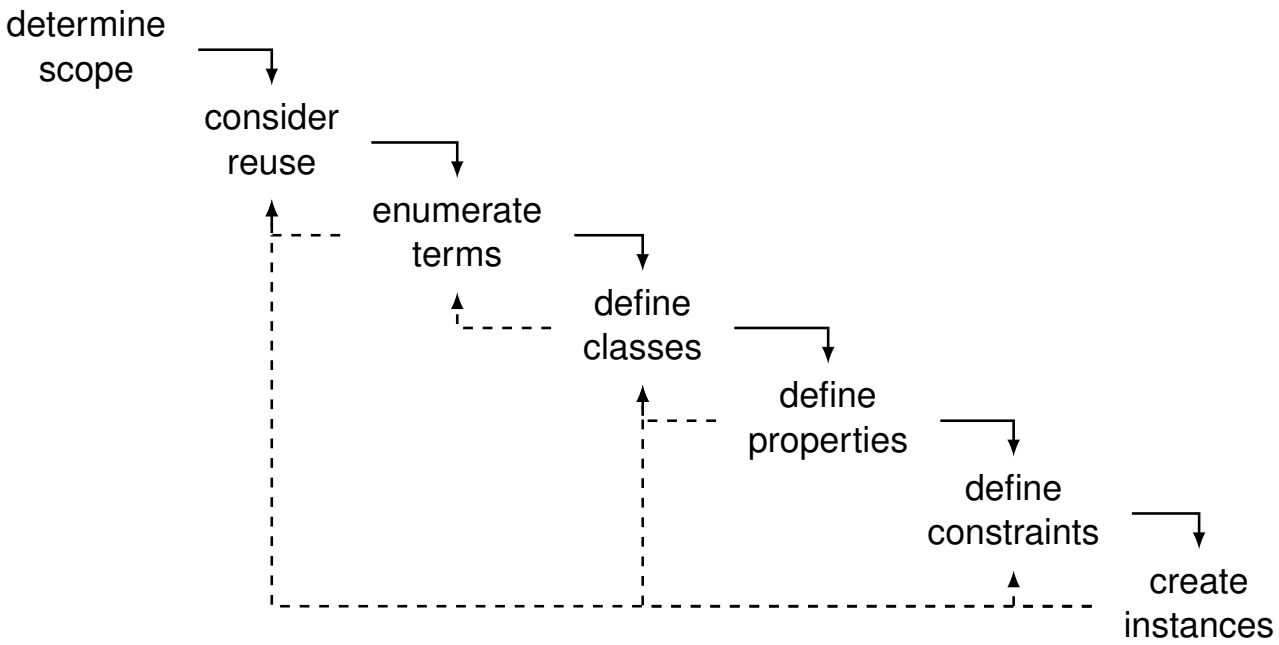
Why use an ontology?

- To share common understanding of the structure of information among people or software agents
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To analyze domain knowledge (through ontology construction)
- However:

  Ontologies do not usually succeed in being application independent and often require adaptation for use in a new application.

Ontology-Development Process

Ideally:



In reality — an iterative process with feedback between succeeding phases.

Ontology Engineering versus Object-Oriented Modelling

An ontology:

- Reflects the structure of the world
- Is often about structure of concepts
- Actual physical representation is not an issue

An OO class structure:

- Reflects the structure of the data and code
- Is usually about behaviour (methods)
- Describes the physical representation of data (long int, char, etc.)

Consider Reuse

Why reuse other ontologies?

- To save the effort
- To interact with the tools that use other ontologies
- To use ontologies that have been validated
- What to re-use?
  - Upper ontologies
    - IEEE Standard Upper Ontology (`suo.ieee.org`)
    - Cyc (`www.cyc.com`)
  - General ontologies
    - DMOZ (`www.dmoz.org`)
    - WordNet (`www.cogsci.princeton.edu/~wn/`)
  - Taxonomies (special kind of ontology)
    - Yahoo categories
    - GAMS: Guide to Available Mathematics

Define Classes and the Class Hierarchy

- A class is a concept in the domain
  - A class of cheese
  - A class of cheese producers
  - A class of blue cheeses
- A class is a collection of elements with similar properties
- Instances of classes
  - Casheil (Irish blue cheese)

Defining Slots and Properties

- Slots in a class definition describe attributes of instances of the class and relations to other instances
- Each wine has colour, sugar content, producer, etc.
- Types of properties:
  - intrinsic properties: aroma and colour of cheese
  - extrinsic properties: name and price of cheese
  - parts: ingredients of a particular cheese
  - objects: producer of cheese
- Simple and complex properties:
  - simple properties (attributes): contain primitive values (strings, numbers)
  - complex properties: contain (or point to) other objects (e.g., a manufacturer)

Slot and Class Inheritance

- A subclass inherits all the slots from the superclass

  If a cheese has a name and characteristic, a blue cheese also has a name and characteristic

- If a class has multiple superclasses, it inherits slots from all of them

  Roquefort is both a sheep cheese and a blue cheese. It inherits "milk source: sheep" from the former and "culture: penicillium" from the latter

- Domain of a slot: the class (or classes) of instances that can have the slot
- Range of a slot: the class (or classes) to which slot values belong

Property Constraints

- Property constraints (facets) describe or limit the set of possible values for a slot
  - The name of a cheese is a string
  - The cheese producer is an instance of Dairy
  - A dairy has exactly one (physical) location
- Common Facets
  - Slot cardinality: the number of values a slot has; exactly *n*, at least 1, at least 0 (= optional)
  - Slot value type: the type of values a slot has; string, number, boolean, enumerated type, complex type (another class)
  - Minimum and maximum value: a range of values for a numeric slot
  - Default value: the value a slot has unless explicitly specified otherwise

Defining Classes and a Class Hierarchy

- There is no single correct class hierarchy
- But there are some guidelines... The question to ask is:

  "Is each instance of the subclass an instance of its superclass?"

- Multiple Inheritance
  - A class can have more than one superclass
  - A subclass inherits slots and facet restrictions from all the parents
  - Different systems resolve conflicts differently
- Disjoint Classes
  - Classes are disjoint if they cannot have common instances
  - Disjoint classes cannot have any common subclasses either

Inverse Slots

- Example: Maker and Product are inverse slots
- Inverse slots contain redundant information, but
- Allow acquisition of the information in either direction
- Enable additional verification
- Allow presentation of information in both directions
- Actual implementation differs from system to system
  - Are both values stored?
  - When are the inverse values filled in? What happens if we change the link to an inverse slot?

Limiting the Scope

- An ontology should not contain all the possible information about the domain
- No need to specialize or generalize more than the application requires
- No need to include all possible properties of a class
  - Only the most salient properties
  - Only the properties that the applications require

Exercise: the pizza ontology

Groups: 3–4 people
Objective: to start the process of building an ontology to describe forms of pizza
Plan:

- Decide whether to work top-down or bottom-up
- Apply methodology outlined on slide 24 [10 mins]
- Discussion [5 mins]

Content

Agents and the Web

- Web content is mostly intended for human readers
- Mostly inaccessible to programs
- Keyword-based search engines have programmatic interfaces, but have limitations:
  - High recall, low precision
  - Low or no recall
  - Results are highly sensitive to vocabulary
  - Results are single web pages
  - Human involvement is necessary to interpret and combine results
  - Results of web searches are not readily accessible by other software tools

From Web to Semantic Web

- The meaning of web content is not machine accessible: lack of semantics
- It is simply difficult, for a machine, to distinguish between different meanings:
  - I am a lecturer of computer science.
  - I am an assistant professor of computer science
- Step 1: Represent web content in a form that is more easily machine-processable
- Step 2: Use intelligent techniques to take advantage of these representations
- The Semantic Web should gradually evolve from existing Web

Semantic Web Enabled Knowledge Management

- Objectives:
  - Knowledge will be organized in conceptual spaces according to its meaning
  - Automated tools for maintenance and knowledge discovery
  - Semantic query answering over several documents
- How?
  - Explicit metadata
  - Ontologies
  - Logic and inference
  - Agents

From HTML to XML

- Humans have little problem understanding HTML content
- Software agents do:
  - How distinguish the name of the course from the name of the lecturer?
  - How determine the course aims?
  - How to infer to follow the link to the lecturer's home page to find the location of their office?
- A better representation might be:

```
1  <department>
2    <courseOffered>CM30174</courseOffered>
3    <departmentName>Computer Science</departmentName>
4    <staff>
5       <lecturer>Marina De Vos</lecturer>
6       <lecturer>Julian Padget</lecturer>
7       <teachingAssistant>who?</teachingAssistant>
8    </staff>
9  </department>
```

Web Ontology Languages

- RDF Schema:
  - RDF is a data model for objects and relations between them
  - RDF Schema is a vocabulary description language (classes, slots, etc.)
  - Describes properties and classes of RDF resources
  - Provides semantics for generalization hierarchies of properties and classes
- The Web Ontology Language (OWL):
  - A richer ontology language
  - Relations between classes relations, e.g., disjointness
  - Cardinality, e.g. “exactly one”
  - Richer typing of properties
  - Characteristics of properties (e.g., symmetry)

Three Species of OWL

- OWL Full:
  - All the OWL languages primitives
  - Arbitrary combination with RDF and RDF Schema
  - Fully upward-compatible with RDF
  - Powerful but undecidable
- OWL DL (Description Logic):
  - A sublanguage of OWL Full: may not apply constructors to constructors
  - Efficient reasoning support: correspondance with description logic
  - Not every RDF document is a valid OWL DL document
- OWL Lite:
  - A subset of OLW DL's constructors: excludes enumerated classes, disjointness statements and arbitrary cardinality
  - Easier to understand and to implement
  - Expressivity significantly restricted

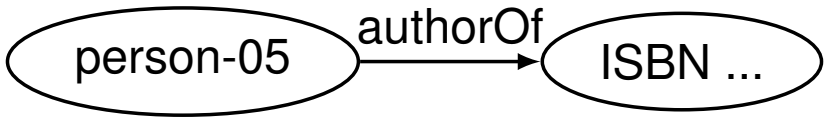
Representing Ontologies: RDF 1/2

- Resource Description Format (RDF), where terms take the form of triples

$$\langle \text{object}, \text{attribute}, \text{value} \rangle$$

- XML syntax:

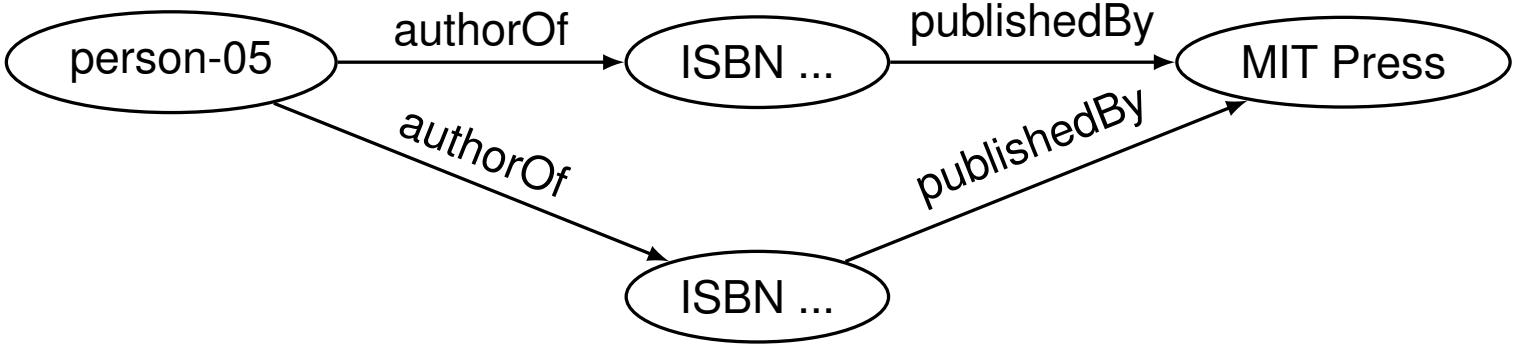
```
1  <rdf:Description rdf:about="#person-05">
2    <authorOf>ISBN...</authorOf>
3  </rdf:Description>
```



- Object denotes a web resource
- Value is another object

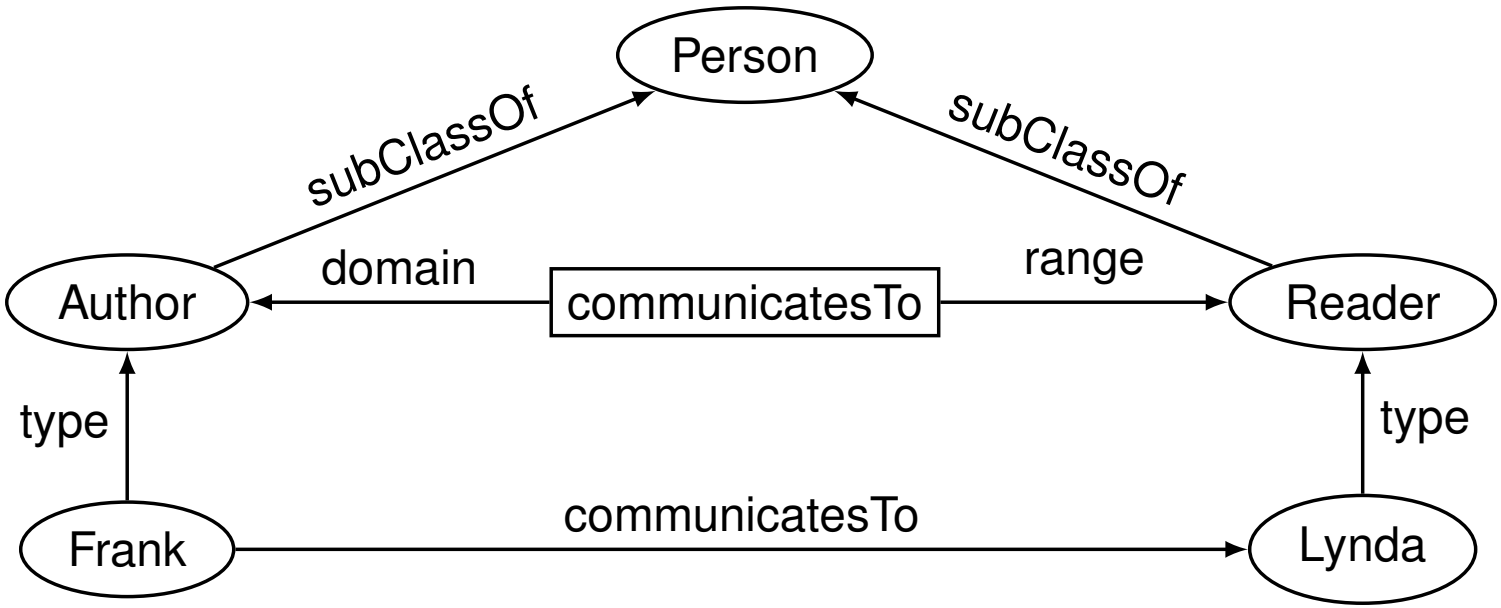
Representing Ontologies: RDF 2/2

- Triples can be linked
- Data model = graph



RDF Schema

- Defines vocabulary for RDF
- Organizes vocabulary in a typed hierarchy
  - Class, subClassOf, type
  - Property, subPropertyOf
  - domain, range



RDF(S) Semantics

- RDF(S) has a (very small) formal semantics:

$$X\ R\ Y + R \text{ domain } T \Rightarrow X \text{ IsOfType } T$$
$$X\ R\ Y + R \text{ range } T \Rightarrow Y \text{ IsOfType } T$$
$$T1 \text{ SubClassOf } T2 + T2 \text{ SubClassOf } T3 \Rightarrow T1 \text{ SubClassOf } T3$$
$$X \text{ IsOfType } T1 + T1 \text{ SubClassOf } T2 \Rightarrow X \text{ IsOfType } T2$$

- Defines what other statements are implied by a given set of RDF(S) statements
- Described as simple entailments with acceptable practical complexity, for example:
  - Aspirin isOfType Painkiller + Painkiller subClassOf Drug $\Rightarrow$ Aspirin isOfType Drug
  - Aspirin alleviates Headache + alleviates range Symptom $\Rightarrow$ Headache isOfType Symptom

Why RDF and OWL are useful

- RDF
  - uniform representation
  - easily generated
  - semantic annotation straightforward
  - “solves” communication syntax problem
- OWL(-S)
  - ontology construction
  - describe semantic properties of concepts
  - describe semantic properties of (web/agent) services
    - inputs, outputs, preconditions, effects (IOPE)
    - semantic matchmaking

Summary

- Agent Communication
- Agent Communication Languages
- Ontology Engineering
- The Semantic Web and Web Ontology Languages

Background Reading (optional)

- Chs 6, 7, 8 of [Wooldridge, 2009].

Wooldridge, M. (2009).
An introduction to multiagent systems (second edition).

Wiley.
ISBN: 978-0-470-51946-2.

- Additional resources include:
  - Protégé: `http://protege.stanford.edu/`
  - OntoWeb: `http://www.ontoweb.org/`
  - The Semantic Web: `http://www.w3.org/2001/sw/`
  - The Co-ode project: `http://www.co-ode.org`