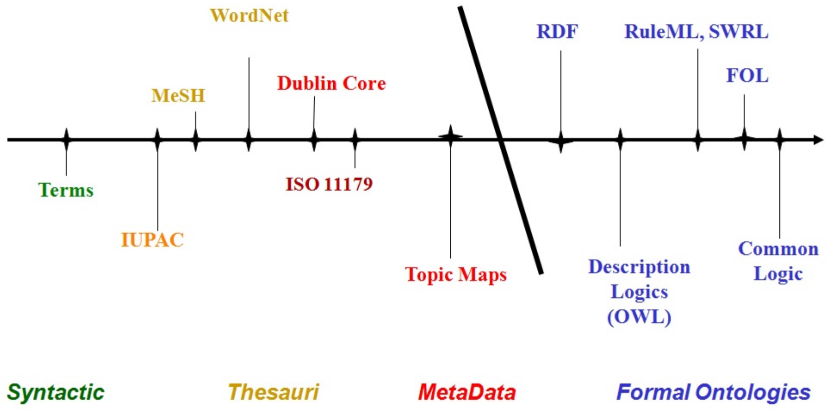
What Is an Ontology

- An ontology defines objects – the types, properties, and interrelationships of entities that exist in a particular domain;
- Ontologies differ on the language used for the specification of meaning;
- A formal ontology includes a declaration of terminology together with a logical specification of the meaning (semantics) of the terms;

Semantic Spectrum[1]



[1]Source: Handbook on Ontologies, Springer, 2009. Editors: Staab, Steffen, Studer, Rudi.

Observations on Objects

- Objects are members of multiple categories (e.g. a doctor, a wife, a mother of two);
- Categories of objects can be more or less specific than others (e.g. a doctor, a professional, a surgeon);
- Categories of objects can have parts, sometimes in multiples (e.g. books have titles, tables have legs and a surface);
- The relation among the parts of an object can be critical in its being a member of a category (e.g. a stack of bricks vs. a pile of bricks)

Noun Phrases: Objects in English

- Categories and properties of objects can be represented by atomic predicates
  - In some cases, these correspond to simple nouns in English such as Person or City;
  - In other cases, the predicates seem to be more like noun phrases such as MarriedPerson or CanadianCity or AnimalWithFourLegs.
- By definition, these predicates have an internal structure and connections to other predicates (e.g. A married person must be a person.)
- This lecture introduces a formal framework that allows us to have both atomic and non-atomic predicates: a description logic.

Concepts, Roles, Constants: Objects in DL

In a description logic, there are sentences that will be true or false. It allows three sorts of expressions that act like nouns and noun phrases in English:

- *Concepts* are like category nouns (Dog, Teenager, GraduateStudent)
- *roles* are like relational nouns (:Age, :Parent, :AreaOfStudy)
- *constants* are like proper nouns (johnSmith, chair128)

Symbols of DL

- Three types of non-logical symbols:
  - atomic concepts: (Dog, Teenager, GraduateStudent)
  - roles: (:Age, :Parent, :AreaOfStudy)
  - constants: (johnSmith, chair128)
- Four types of logical symbols:
  - punctuation: [,], (,)
  - positive integers: 1, 2, 3, ...
  - concept-forming operators: ALL, EXISTS, FILLS, AND
  - connectives: $\sqsubseteq$, $\doteq$, and $\rightarrow$

Syntax of DL

- The set of concepts is the one satisfying:
  - Every atomic concept is a concept,
  - If r is a role and d is a concept, then $[ALL\ r\ d]$ is a concept,
  - If r is a role and n is an integer, then $[EXISTS\ n\ r]$ is a concept,
  - If r is a role and c is a constant, then $[FILLS\ r\ c]$ is a concept,
  - If d_1, ..., d_k are concepts, then so is $[AND\ d_1, \ldots, d_k]$.
- Three types of sentences in DL:
  - If d and e are concepts, then $(d \sqsubseteq e)$ is a sentence,
  - if d and e are concepts, then $(d \doteq e)$ is a sentence,
  - If d is a concept and c is a constant, then $(c \rightarrow d)$ is a sentence.

The Meaning of DL Concepts

- Constants stand for individuals, concepts for sets of individuals, and roles for binary relations.
- The meaning of a complex concept is derived from the meaning of its parts the same way a noun phrases is:
  - [*EXISTS n r*] describes those individuals that stand in relation *r* to at least *n* other individuals (e.g., [*EXISTS* 1 :*Child*]);
  - [*FILLS r c*] describes those individuals that stand in the relation *r* to the individual denoted by *c* (e.g., [*FILLS* :*Cousin joe*]);
  - [*ALL r d*] describes those individuals that stand in relation *r* only to individuals that are described by *d* (e.g., [*ALL* :*Employee UnionMember*]);
  - [*AND* $d_1 \ldots d_k$] describes those individuals that are described by all of the d_i (e.g., [*AND Professor Canadian*]).

The Meaning of DL Concepts: An Example

- a company with at least 7 directors, whose managers are all women with PhDs, and whose min salary is \$40/hr"
- $[\mathit{AND}\ \mathit{Company}$
  $\quad [\mathit{EXISTS}\ 7\ :\mathit{Director}]$
  $\quad [\mathit{ALL}\ :\mathit{Manager}\ [\mathit{AND}\ \mathit{Woman}\ [\mathit{FILLS}\ :\mathit{Degree}\ \mathit{phD}]]]$
  $\quad [\mathit{FILLS}\ :\mathit{MinSalary}\ \$40.00/\mathit{hour}]$
  $]$

A DL Knowledge Base

A DL knowledge base is a set of DL sentences serving mainly to

- give names to definitions (e.g., "A FatherOfDaughters is precisely a male with at least one child and all of whose children are female")
  $FatherOfDaughters \doteq$
  $[AND\ Male\ [EXISTS\ 1\ :Child][ALL\ :Child\ Female]]$
- give names to partial definitions (e.g., "A dog is among other things a mammal that is a pet whose voice call includes barking"[2])
  $Dog \sqsubseteq [AND\ \ Mammal\ \ Pet\ \ [FILLS\ \ :VoiceCall\ \ barking]]$
- assert properties of individuals (e.g., "Joe is a FatherOfDaughters and a Surgeon")
  $joe \rightarrow [AND\ FatherOfDaughters\ Surgeon]$

[2] This gives necessary but not sufficient conditions.

Formal Semantics

- Interpretation $\mathcal{I} = \langle D, I \rangle$, where
  - for every constant c, $I[c] \in D$
  - for every atomic concept a, $I[a] \subseteq D$
  - for every role r, $I[r] \subseteq D \times D$

  We then extend the interpretation to all concepts:
  - $I[[ALL\ r\ d]] = \{x \in D | \text{for any y}, if \langle x, y \rangle \in I[r] \text{ then } y \in I[d]\}$
  - $I[[EXISTS\ n\ r]] =$
    $\{x \in D | \text{there are at least n } y \text{ such that} \langle x, y \rangle \in I[r]\}$
  - $I[[FILLS\ r\ c]] = \{x \in D | \langle x, I[c] \rangle \in I[r]\}$
  - $[[AND\ \ d_1 \ldots d_k]] = I[d_1] \cap \ldots \cap I[d_k]$
- A sentence of DL will then be true or false as follows:
  - $\mathcal{I} \models (d \sqsubseteq e)$ iff $I[d] \subseteq I[e]$
  - $\mathcal{I} \models (d \doteq e)$ iff $I[d] = I[e]$
  - $\mathcal{I} \models (c \rightarrow e)$ iff $I[c] \in I[e]$

Entailment vs. Validity

- In some cases, an entailment will hold because the sentence in question is valid.
  - $[AND\ Doctor\ Female] \sqsubseteq Doctor$
  - $[FILLS\ :Child\ \ sue] \sqsubseteq [EXISTS\ \ 1\ :Child]$
- But in most other cases, the entailment depends on the sentences in the KB.
  For example,

$$[AND\ Surgeon\ Female] \sqsubseteq\ Doctor$$

  is not valid.
  But it is entailed by a KB that contains
  1. $Surgeon \doteq\ [AND\ Specialist\ [FILLS\ :Specialty\ surgery]]$
  2. $Specialist \sqsubseteq Doctor$

Entailment and Reasoning

- Entailment in DL:
  - A set of DL sentences S entails a sentence α (which we write $S \models \alpha$) iff for every $\mathcal{I}$, if $\mathcal{I} \models S$ then $\mathcal{I} \models \alpha$
  - A sentence is valid iff it is entailed by the empty set.
- Given a KB consisting of DL sentences, there are two basic sorts of reasoning we consider:
  - determining if $KB \models (c \rightarrow e)$: whether a named individual **satisfies** a certain description
  - determining if $KB \models (d \sqsubseteq e)$: whether one description is **subsumed** by another
  - the other case, $KB \models (d \doteq e)$ is reduces to $KB \models (e \sqsubseteq d)$ and $KB \models (d \sqsubseteq e)$

Computing Subsumption

- We begin with computing subsumption, that is, determining whether or not $KB \models (d \sqsubseteq e)$.
- Some simplifications to the KB:
  - we can remove the $(c \rightarrow d)$ assertions from the KB
  - we can replace $(d \sqsubseteq e)$ in KB by $(d \doteq [AND\ e\ a])$, where a is a new atomic concept
  - we assume that in the KB for each $(d \doteq e)$, the d is atomic and appears only once on the LHS
  - we assume that the definitions in the KB are acyclic vs. cyclic $(d \doteq [AND\ e\ f]), (e \doteq [AND\ d\ g])$
- Under these assumptions, it is sufficient to do the following:
  - normalization: using the definitions in the KB, put d and e into a special normal form, d' and e'
  - structure matching: determine if each part of e' is matched by a part of d'

Normalization

Repeatedly apply the following operations to the two concepts:

- expand a definition: replace an atomic concept by its KB definition
- flatten an AND concept:
  $[AND \ \ldots [AND\ d\ e\ f] \ldots] \Rightarrow [AND \ldots d\ e\ f\ \ \ldots]$
- combine the ALL operations with the same role:
  [AND ... [ALL r d] ... [ALL r e] ...] $\Rightarrow$
  [AND ... [ALL r [AND d e]] ...]
- combine the EXISTS operations with the same role:
  [AND ... [EXISTS n_1 r] ... [EXISTS n_2 r] ...] $\Rightarrow$
  [AND ... [EXISTS n r] ...] (where n = Max(n_1, n_2))

Normalization Example

$$
\begin{array}{l}
[\mathit{AND}\ \mathit{Person}\\
\qquad [\mathit{ALL}\ {:}\mathit{Friend}\ \mathit{Doctor}]\\
\qquad [\mathit{EXISTS}\ 1{:}\mathit{Accountant}]\\
\qquad [\mathit{ALL}\ {:}\mathit{Accountant}\ [\mathit{EXISTS}\ 1\ {:}\mathit{Degree}]]\\
\qquad [\mathit{ALL}\ {:}\mathit{Friend}\ \mathit{Rich}]\\
\qquad [\mathit{ALL}\ {:}\mathit{Accountant}\ [\mathit{AND}\ \mathit{Lawyer}\ [\mathit{EXISTS}\ 2\ {:}\mathit{Degree}]]]]
\end{array}
$$

$$\Downarrow$$

$$
\begin{array}{l}
[\mathit{AND}\ \mathit{Person}\\
\qquad [\mathit{EXISTS}\ 1\ {:}\mathit{Accountant}]\\
\qquad [\mathit{ALL}\ {:}\mathit{Friend}\ [\mathit{AND}\ \mathit{Rich}\ \mathit{Doctor}]]\\
\qquad [\mathit{ALL}\ {:}\mathit{Accountant}\ [\mathit{AND}\ \mathit{Lawyer}\ [\mathit{EXISTS}\ 1\ {:}\mathit{Degree}]\ [\mathit{EXISTS}\ 2\ {:}\mathit{Degree}]]]]
\end{array}
$$

$$\Downarrow$$

$$
\begin{array}{l}
[\mathit{AND}\ \mathit{Person}\\
\qquad [\mathit{EXISTS}\ 1\ {:}\mathit{Accountant}]\\
\qquad [\mathit{ALL}\ {:}\mathit{Friend}\ [\mathit{AND}\ \mathit{Rich}\ \mathit{Doctor}]]\\
\qquad [\mathit{ALL}\ {:}\mathit{Accountant}\ [\mathit{AND}\ \mathit{Lawyer}\ [\mathit{EXISTS}\ 2\ {:}\mathit{Degree}]]]]
\end{array}
$$

Structure matching

Once we have replaced atomic concepts by their definitions, we no longer need to use the KB. To see if a normalized concept $[AND\ e_1 \ldots e_m]$ subsumes a normalized concept $[AND\ d_1 \ldots d_n]$, we do the following – for each component e_j, check that there is a matching component d_i:

- if e_j is atomic or $[FILLS\ r\ c]$, then d_i must be identical to it;
- if $e_j = [EXISTS\ 1\ r]$, then d_i must be $[EXISTS\ n\ r]$ or $[FILLS\ r\ c]$;
- if $e_j = [EXISTS\ n\ r]$ where $n > 1$, then d_i must be of the form $[EXISTS\ m\ r]$ where $m \geq n$;
- if $e_j = [ALL\ r\ e']$, then d_i must be $[ALL\ r\ d']$, where recursively e' subsumes d'.

In other words, for every part of the more general concept, there must be a corresponding part in the more specific one. It can be shown that this procedure is sound and complete: it returns YES iff $KB \models (d \sqsubseteq e)$.

Structure Matching Example

```
[AND Person
     [FILLS :Age 27]
          [EXISTS 2 :Child]
               [EXISTS 1 :Spouse]
                    [ALL :Friend [AND Doctor Rich]] ]
```

```
[AND Person
     Female
        [FILLS :Age 27]
             [EXISTS 3 :Child]
                  [FILLS :Spouse jack]
                       [FILLS :Mother sue]
                            [ALL :Friend [AND Doctor
                                      [FILLS :Specialty surgery]
                                      Rich]] ]
```

not needed

Computing Satisfaction

- To determine if $KB \models (c \rightarrow e)$, we use the following procedure:
  - find the most specific concept d such that $KB \models (c \rightarrow d)$
  - determine whether or not $KB \models (d \sqsubseteq e)$, as before.
- To a first approximation, the d we need is the AND of every d_i such that $(c \rightarrow d_i) \in KB$. However, this can miss some inferences.
- for example: Suppose the KB contains $(joe \rightarrow Person)$ $(canCorp \rightarrow$ $[AND\,Company\,[ALL{:}Manager\,Canadian][FILLS{:}Manager\,joe]])$ then the $KB \models (joe \rightarrow Canadian)$.
- To find the d, a more complex procedure is used that propagates constraints from one individual (canCorp) to another (joe).

Some Applications of DL

- DL have been and can be used in a number of applications:
  - To develop directly meta-data, not only representing taxonomies and inheritance of properties of objects, but also reasoning over them;
  - To formalize the semantics of the tags of those syntax-based markup languages and tools such as XML;
  - To create interface for relational database (through providing a higher level view of the data based on objects);
  - To detect contradiction in a system.