Knowledge Representation & Reasoning

Shyamanta M Hazarika
CSE, SoE
Tezpur University

Computing satisfaction

To determine if KB $\models (c \rightarrow e)$, we use the following procedure:

1. find the most specific concept d such that KB $\models (c \rightarrow d)$
2. 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!

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).

The individuals we need to consider need not be named by constants; they can be individuals that arise from EXISTS (like Skolem constants).

Taxonomies

Two common sorts of queries in a DL system:

- given a query concept q, find all constants c such that KB $\models (c \rightarrow q)$

  e.g. q is [AND Stock FallingPrice MyHolding]

  might want to trigger a procedure for each such c

- given a query constant c, find all *atomic* concepts a such that KB $\models (c \rightarrow a)$

We can exploit the fact that concepts tend to be structured hierarchically to answer queries like these more efficiently.

Taxonomies arise naturally out of a DL KB:

- the nodes are the atomic concepts that appear on the LHS of a sentence $(a \sqsubseteq d)$ or $(a \doteq d)$ in the KB
- there is an edge from a_i to a_j if $(a_i \sqsubseteq a_j)$ is entailed and there is no distinct a_k such that $(a_i \sqsubseteq a_k)$ and $(a_k \sqsubseteq a_j)$.

  can link every constant c to the most specific atomic concepts a in the taxonomy such that KB $\models (c \rightarrow a)$

Positioning a new atom in a taxonomy is called classification

Computing classification

Consider adding ($a_{new} \doteq d$) to the KB.

- find S, the most specific subsumers of d: the atoms a such that KB $\models (d \sqsubseteq a)$, but nothing below a — see below
- find G, the most general subsumees of d: the atoms a such that KB $\models (a \sqsubseteq d)$, but nothing above a

  if $S \cap G$ is not empty, then a_{new} is not new
- remove any links from atoms in G to atoms in S
- add links from all the atoms in G to a_{new} and from a_{new} to all the atoms in S
- reorganize the constants:

  for each constant c such that KB $\models (c \rightarrow a)$ for all $a \in S$,
  but KB $\models (c \rightarrow a)$ for no $a \in G$, and where KB $\models (c \rightarrow d)$,
  remove links from c to S and put a single link from c to a_{new}.

Adding ($a_{new} \sqsubseteq d$) is similar, but with no subsumees.

Subsumers and subsumees

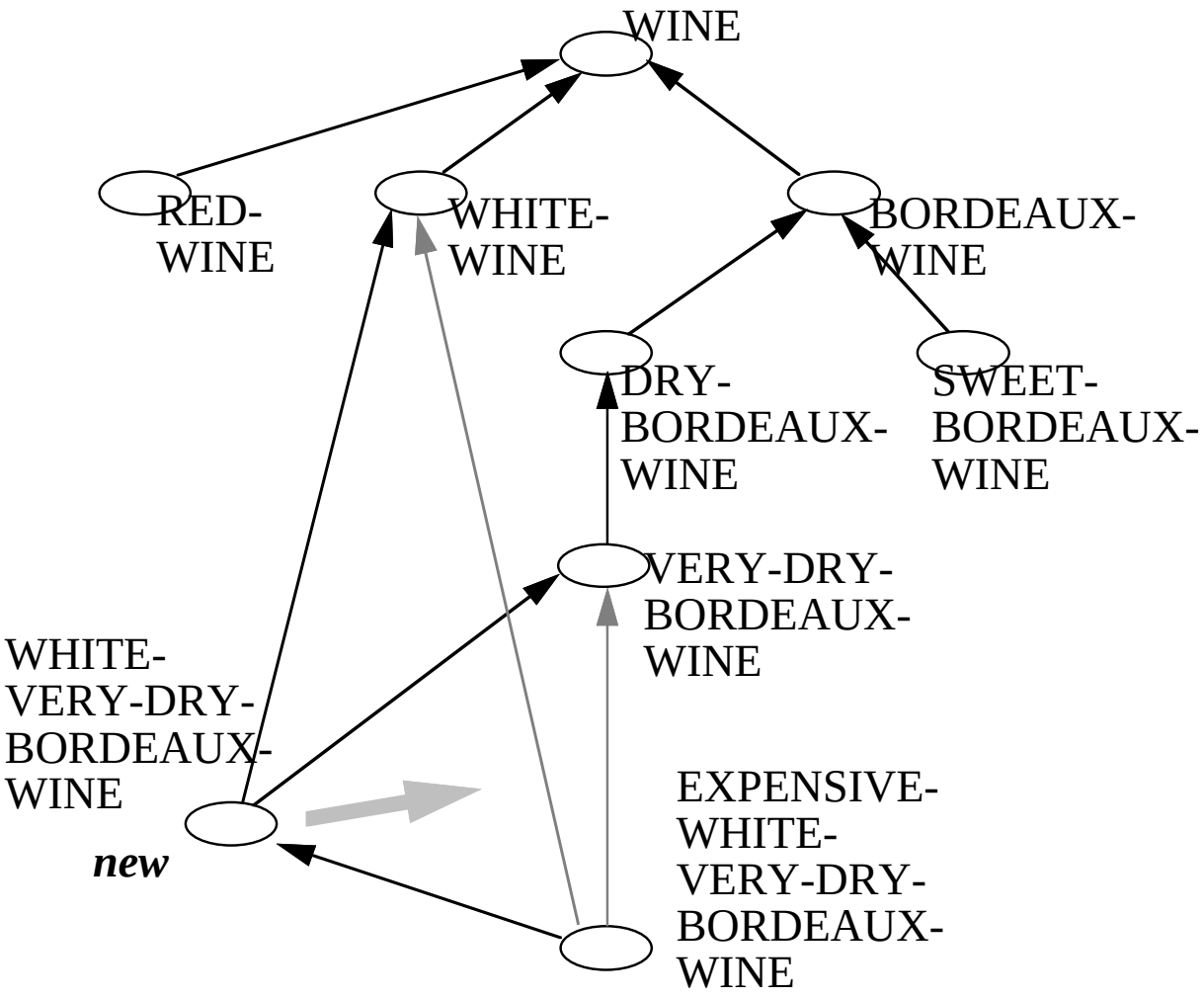
Calculating the most specific subsumers of a concept d:

- Start with $S = \{\text{Thing}\}$.
- Repeatedly do the following:
  - Suppose that some $a \in S$ has at least one child a' just below it in the taxonomy such that KB $\models (d \sqsubseteq a')$.
  - Then remove a from S and replace it by all such children a'.

Calculating the most general subsumees of a concept d:

- Start with G = the most specific subsumers.
- Repeatedly do the following:
  - Suppose that for some $a \in G$, KB $\not\models (a \sqsubseteq d)$.
  - Then remove a from G and replace it by all of its children (or delete it, if there are none).
- Repeatedly delete any element of G that has a parent subsumed by d.

An example of classification



Using the taxonomic structure

Note that classification uses the structure of the taxonomy:

> If there is an a' just below a in the taxonomy such that KB $\not\models (d \sqsubseteq a')$, we never look below this a'. If this concept is sufficiently high in the taxonomy (e.g. just below Thing), an entire subtree will be ignored.

Queries can also exploit the structure:

> For example, to find the constants described by a concept q, we simply classify q and then look for constants in the part of the taxonomy subtended by q. The rest of the taxonomy not below q is ignored.

This natural structure allows us to build and use very large knowledge bases.

- the time taken will grow linearly with the *depth* of the taxonomy
- we would expect the depth of the taxonomy to grow *logarithmically* with the size of the KB
- under these assumptions, we can handle a KB with thousands or even millions of concepts and constants.