Structured descriptions

The syntax of FOL makes it easy to say things about objects. Frames organize knowledge in terms of categories of objects.

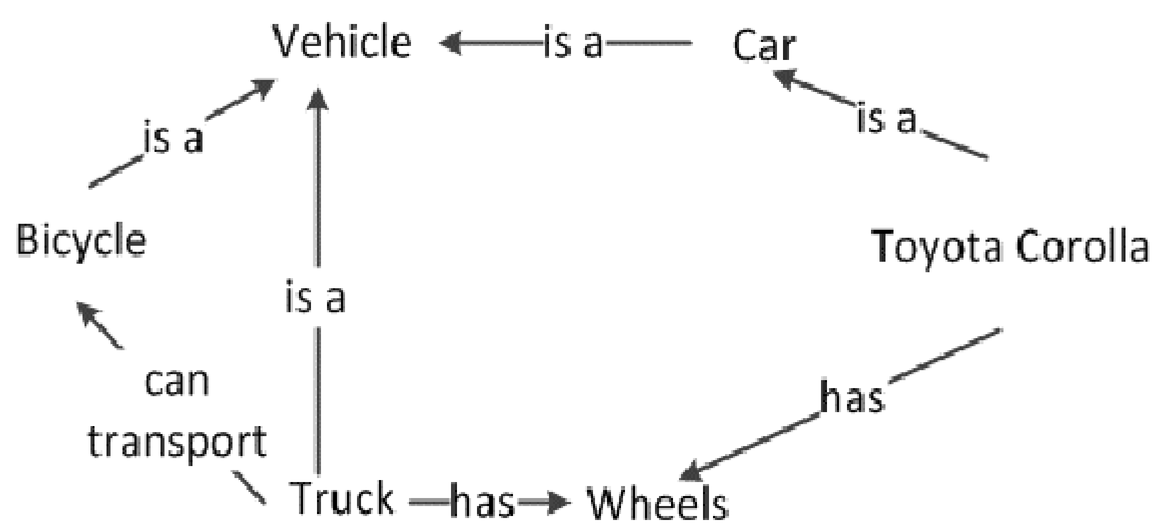
Description logics are notations that are designed to make it easier to describe definitions and properties of categories, by adding structure to the definition of objects.

The focus is on declarative aspects of objects-oriented representation, going back to concepts like predicates and entailment from FOL.

These slides are mainly based on Ronald J. Brachman, Hector J. Levesque. Knowledge representation and reasoning, Morgan Kaufmann, 2004.

Structured descriptions

Description logic systems evolved from frames/semantic networks by formalizing what the networks mean, while keeping the emphasis on taxonomic structure as an organizing principle (that helps in organizing a hierarchy of categories).



Example of a semantic network, taken from [1].

1. E. Rajangam and C. Annamalai. Graph Models for Knowledge Representation and Reasoning for Contemporary and Emerging Needs – A survey. In International Journal of Information Technology and Computer Science, February 2016.

Structured descriptions

The principal inference tasks for description logics are <u>subsumption</u> (checking if a category is a subset of another by comparing their definitions) and <u>satisfaction</u> (checking whether an object belongs to a category).

In standard FOL systems, predicting the solution time is often impossible. In description logics, the subsumption testing can be solved in time polynomial in the size of the description. But (hard) problems either cannot be stated at all in description logics or they require exponentially large descriptions.

In FOL, we represent categories of objects with simple predicates like Mother(x), Boat(x), Company(x).

To represent more interesting types of constructions like “a man whose children are all girls” we need predicates with internal structure.

We would expect that if Child(x,y) and FatherOfOnlyGirls(x) were true, then y would have to be a girl (somehow) by definition.

Structured descriptions

We have category nouns like FatherOfOnlyGirls, Girl describing classes of objects and relational nouns like Child that are parts/attributes/properties of other objects.

In description logics, we refer to the first type as a concept and to the second type as a role (in frame systems we saw a similar distinction between frames/slots).

In contrast to the slots in frame systems, role can have multiple fillers. Thus, it can be described naturally a person with several children, a salad made from more than one type of vegetable.

Although much of the reasoning in description logics concerns generic categories, constants are included to allow for descriptions to be applied to individuals.

A description language

In a description language (DL) there are two types of symbols:

- logical symbols, with a fixed meaning
- nonlogical symbols, which are application dependent

There are four types of logical symbols:

- punctuation: [,],(,)
- positive integers: 1,2,3,…
- concept-forming operators: ALL, EXISTS, FILLS, AND
- connectives: $\sqsubseteq$, $\doteq$, $\rightarrow$

A description language

There are three types of nonlogical symbols:

- atomic concepts – the name starts with upper-case
  - Person, Company, FatherOfOnlyGirls
  - Thing – a special atomic concept
- roles – the name starts with upper-case, prefixed by:
  - :Age, :Child
- constants – the name starts with lower-case
  - table17, johnSmith

A description language

There are four types of legal syntactic expressions:

constants c

roles r

concepts d,e; atomic concepts a

sentences

The set of concepts of DL satisfies the following:

-every atomic concept is a concept

-if r is a role, d is a concept then [ALL r d] is a concept

-if r is a role, $n \in \mathbb{N}^*$ then [EXISTS n r] is a concept

-if r is a role, c is a constant then [FILLS r c] is a concept

-if $d_1,\dots,d_n$ are concepts then [AND $d_1 \dots d_n$] is a concept

A description language

There are three types of sentences in DL:

- if d_1, d_2 are concepts then $(d_1 \sqsubseteq d_2)$ is a sentence
- if d_1, d_2 are concepts then $(d_1 \doteq d_2)$ is a sentence
- if c is a constant and d concept then $(c \rightarrow d)$ is a sentence

A knowledge base KB in DL is a collection of sentences.

Constants represent individuals in the application domain; concepts represent categories or classes of individuals; and roles represent <u>binary</u> relations between individuals.

The meaning of a complex concept derives from the meaning of its parts.

A description language

For example, [EXISTS n r] represents the class of individuals in the domain that are related by relation r to at least n other individuals.

[EXISTS 1 :Child] represents persons who have at least one child.

If c is a constant that stands for some individual, the concept [FILLS r c] represents those individuals that are in relation r with c.

[FILLS :Cousin george] represents persons whose cousin is George.

If concept d represents a class of individuals, [ALL r d] represents the class of individuals who are in relation r only to individuals of class d.

[ALL :Employee UnionMember] describes companies whose employees are all union members.

The concept [AND $d_1 \ldots d_n$] represents anything described by d_1 and … d_n.

A description language

```
[AND Wine
        [FILLS :Color red]
        [EXISTS 2 :GrapeType]
]

(ProgressiveCompany≐[AND Company
                        [EXISTS 7 :Director]
                        [ALL :Manager [AND Women
                                           [FILLS :Degree phd]
                                      ]
                        ]
                        [FILLS :MinSalary $5000]
                    ]
)
```

A description language

In DL sentences are true or false in the domain (like in FOL).

d_1,d_2 concepts and c constant

($d_1 \sqsubseteq d_2$) says that d_1 is subsumed by d_2, that is all individuals that satisfy d_1 also satisfy d_2.

(Surgeon$\sqsubseteq$Doctor)

($d_1 \doteq d_2$) says that d_1 and d_2 are equivalent, that is the individuals that satisfy d_1 also exactly those that satisfy d_2. It is the same as saying that both ($d_1 \sqsubseteq d_2$) and ($d_2 \sqsubseteq d_1$) are true.

($c \rightarrow d$) says that the individual denoted by c satisfies the description expressed by d.

A description language

Interpretations in DL

An interpretation $\mathcal{I}$ is a pair <D,I>, where D is a non-empty set of objects called the domain of the interpretation and I is the interpretation mapping that assigns a meaning to the nonlogical symbols of DL, so that:

1. for every constant c, $I[c] \in D$;
2. for every atomic concept a, $I[a] \subseteq D$;
3. for every role r, $I[r] \subseteq D \times D$

The set I[d] is called the extension of the concept d:

$I[Thing]=D$;

$I[[ALL\ r\ d]]=\{x \in D \mid \forall y \text{ if } \langle x,y \rangle \in I[r] \text{ then } y \in I[d]\}$

$I[[EXISTS\ n\ r]]=\{x \in D \mid \text{there are at least n distinct y such that } \langle x,y \rangle \in I[r]\}$

$I[[FILLS\ r\ c]]=\{x \in D \mid \langle x,I[c] \rangle \in I[r]\}$

$I[[AND\ d_1 \ldots d_n]]=I[d_1] \cap \ldots \cap I[d_n]$

A description language

Truth in an interpretation

The sentence (c→d) is true in $\mathcal{I}$ if the object denoted by c is in the extension of d $I[c] \in I[d]$

The sentence (d⊑d′) is true in $\mathcal{I}$ if the extension of d is a subset of the extension of d′ $I[d] \subseteq I[d']$

The sentence (d≐d′) is true in $\mathcal{I}$ if $I[d]=I[d']$

If a sentence α is true in $\mathcal{I}$, we write $\mathcal{I} \models \alpha$.

If S is a set of sentences, we will write $\mathcal{I} \models S$ to say that all the sentences in S are true in $\mathcal{I}$.

A description language

Entailment

Let S be a set of sentences in DL and α a sentence. S logically entails α, and we write $S \models \alpha$, iff for every interpretation $\mathscr{I}$, if $\mathscr{I} \models S$ then $\mathscr{I} \models \alpha$.

A sentence α is logically valid, and we write $\models \alpha$, if it is logically entailed by the empty set.

In DL, there are two basic types of reasoning: determining whether or not a constant c satisfies a concept d; and determining whether or not a concept d is subsumed by another concept d′:

$$KB \models (c \rightarrow d)$$

$$KB \models (d \sqsubseteq d')$$

A description language

Examples of valid sentences:

([AND Doctor Female] ⊑Doctor)

(john→Thing)

In more typical cases, the entailment depends on sentences in the KB. For example, if KB contains the sentence (Surgeon⊑Doctor), then we can logically entail that

KB ⊨ ([AND Surgeon Female]⊑Doctor)

We can reach the same conclusion if we have in the KB the sentence

(Surgeon≐ [AND Doctor [FILLS :Specialty surgery]]) instead of

(Surgeon⊑Doctor).

But with the empty KB, we would have no subsumption relation

([AND Surgeon Female]⊑Doctor) because we can choose an interpretation $\mathcal{I}$ in which the sentence is false.

For example, I[Doctor]=∅ and I[Surgeon]=I[Female]={anna}.

A description language

Computing entailments

Given a KB, we want to determine if KB $\models$ α for α of the form:

$(c \rightarrow d)$ where c is constant and d concept

$(d \sqsubseteq e)$ where d,e concepts

[KB $\models$ $(d \doteq e)$ iff KB $\models$ $(d \sqsubseteq e)$ and KB $\models$ $(e \sqsubseteq d)$]

A description language

Simplifying the KB

Prop. Subsumption entailments are not affected by the presence of sentences ($c \rightarrow d$) in KB.

That is to say that KB$\models$ ($d \sqsubseteq e$) iff KB′ $\models$ ($d \sqsubseteq e$),

where KB′=KB – {all sentences ($c \rightarrow d$)}.

For subsumption questions, we assume that the KB contains no ($c \rightarrow d$) sentences.

Moreover, we can replace sentences of the form ($d \sqsubseteq e$) by ($d \doteq$[AND e a]), where a is a new atomic concept used nowhere else.

A description language

We will consider the following restrictions in the KB:

-the left-hand sides of ≐ is an atomic concept other than Thing

-each atom appears on the left-hand side of ≐ exactly once in KB – such sentences provide definitions of the atomic concepts

```
(RedBordeauxWine ≐ [AND Wine
                        [FILLS :Color red]
                        [FILLS :Region bordeaux]
                   ]
)
```

-we assume that sentences ≐ in KB are acyclic. We rule out a KB that contains

$(d_1 \doteq [\text{AND } d_2 \ldots]), (d_2 \doteq [\text{ALL } r\ d_3]), (d_3 \doteq [\text{AND } d_1 \ldots])$.

A description language

Under these restrictions, to determine if KB $\models$ (d$\sqsubseteq$e) we do the following:

1. put d and e into a special normalized form
2. determine whether each part of the normalized e is accounted for by some part of the normalized d.

   We are looking for a structural relation between two normalized concepts.

   For example, if e contains [ALL r e′] then d must contain [ALL r d′] with d′$\sqsubseteq$e′.

A description language

Normalization

It is a preprocessing that simplifies the structure-matching between concepts. It applies to one concept at a time and involves the following steps:

1. Expand definitions – any atomic concept in the left-hand side of ≐ is replaced by its definition

   Example:

   If in KB we have the sentence

   (Surgeon≐[AND Doctor [FILLS :Specialty surgery]])

   The concept [AND…Surgeon…] expands to

   [AND…[AND Doctor [FILLS :Specialty surgery]]…]

2. Flatten the AND operators

   [AND…[AND d_1…d_n]…] becomes [AND…d_1…d_n…]

3. Combine the ALL operators

   [AND…[ALL r d_1]…[ALL r d_2]…] becomes
   [AND…[ALL r [AND d_1 d_2]]…]

A description language

4. Combine the EXISTS operators
   [AND…[EXISTS n_1 r]…[EXISTS n_2 r]…] becomes
   [AND…[EXISTS n r]…] where $n=\max(n_1,n_2)$.

5. Thing concept – remove Thing, [ALL r Thing] and AND with no arguments if they appear as arguments in an AND concept
   [AND…Thing…] becomes [AND…]
   [AND Company [ALL :Employee Thing]] becomes Company

6. Remove redundant expressions – eliminate duplicates within the same AND expression.

A description language

These six steps are applied repeatedly until no steps are applicable. The result is either Thing, an atomic concept or a concept of the following form:

$$
\begin{array}{l}
[\text{AND } a_1 \ldots a_m \\
\qquad [\text{FILLS } r_1\ c_1] \ldots [\text{FILLS } r_{m1}\ c_{m1}] \\
\qquad [\text{EXISTS } n_1\ s_1] \ldots [\text{EXISTS } n_{m2}\ s_{m2}] \\
\qquad [\text{ALL } t_1\ e_1] \ldots [\text{ALL } t_{m3}\ e_{m3}] \\
]
\end{array}
$$

where $a_1, \ldots, a_m$ are atomic concepts (other than Thing), r_i, s_i, t_i are roles, c_i are constants, n_i are positive integers and e_i are normalized concepts.

A description language

Example 1 We are given the following KB:

```
(WellRoundedCo≐[AND Company
                    [ALL :Manager [AND B-SchoolGrad
                                       [EXISTS 1 :TechnicalDegree]
                                  ]
                    ]
               ])

(HighTechCo≐[AND Company
                 [FILLS :Exchange nasdaq]
                 [ALL :Manager Techie]
            ])

(Techie≐[EXISTS 2 :TechnicalDegree])
```

Normalize the concept [AND WellRoundedCo HighTechCo]

A description language

1

⇒ [AND [AND Company [ALL :Manager [AND B-SchoolGrad
[EXISTS 1 :TechnicalDegree]
]]]

2   3

[AND Company
[FILLS :Exchange nasdaq]
[ALL :Manager [EXISTS 2 :TechnicalDegree]]]

]

A description language

2

2,3

⇒ [AND Company [ALL :Manager [AND [AND B-SchoolGrad

[EXISTS 1 :TechnicalDegree]

6

] 4

[EXISTS 2 :TechnicalDegree]

]

Company

[FILLS :Exchange nasdaq]

]

A description language

2

2,3

⇒ [AND Company [ALL :Manager [AND [AND B-SchoolGrad

6

[EXISTS 1 :TechnicalDegree]

] 4

[EXISTS 2 :TechnicalDegree]

]

Company

[FILLS :Exchange nasdaq]

]

2,4,6

⇒ [AND Company

[ALL :Manager [AND B-SchoolGrad

[EXISTS 2 :TechnicalDegree]]]

[FILLS :Exchange nasdaq]]

]

A description language

Structure matching procedure – subsumption computation

Input: d and e are two normalized concepts

d is [AND $d_1 \ldots d_m$]

e is [AND $e_1 \ldots e_{m'}$]

Output: YES or NO according to whether or not KB $\models$ ($d \sqsubseteq e$)

Return YES iff for each e_j, $j \in 1,m'$, there exists a component d_i, $i \in 1,m$ such that d_i matched e_j as follows:

1. If e_j is an atomic concept then d_i must be identical to e_j
2. If e_j is of the form [FILLS r c] then d_i must be identical to it
3. If e_j is of the form [EXISTS n r] then d_i must be of the form [EXISTS n′ r] for some $n' \geq n$; if n=1, d_i can also be of the form [FILLS r c] for any constant c
4. If e_j is of the form [ALL r e′], then d_i must be of the form [ALL r d′], where recursively $d' \sqsubseteq e'$

A description language

Example 2

[AND Company
d 1 [ALL :Manager B-SchoolGrad]
[EXISTS 1 :Exchange]]]
]

4

3 [AND Company
[ALL :Manager [AND B-SchoolGrad
[EXISTS 2 :TechnicalDegree]]]
d′ [FILLS :Exchange nasdaq]
]

So, d′⊑d.

A description language

Computing satisfaction

We are interested whether KB $\models$ (b$\rightarrow$e), where b is a constant and e is a concept.

To find out if an individual satisfies a description, we need to propagate the information implied by what we know about other individuals before checking for subsumption. This can be done by a *forward chaining* procedure.

In the case where there are no EXISTS terms in any concept, the procedure is as following:

1. Construct S a list of pairs (b,d), where b is any constant mentioned in KB and d is the normalized version of the concept [AND $d'_1 \ldots d'_n$] for all d'_i such that (b$\rightarrow d'_n$)$\in$KB.
2. Find two constants b_1 and b_2 such that $(b_1,d_1)\in S$ and $(b_2,d_2)\in S$, [FILLS r b_2] and [ALL r e] are both components of d_1 but KB $\not\models$ ($d_2 \sqsubseteq e$).
3. If no b_1 and b_2 can be found, then exit. Otherwise, replace the pair (b_2,d_2) in S by (b_2, d'_2), where d'_2 is the normalized version of [AND d_2 e] and go to step 2.

A description language

The procedure computes for each constant b the most specific concept d such that KB ⊨ (b→d). Now, to test whether or not KB ⊨ (b→e), we need only to test whether or not KB ⊨ (d⊑e).

Example 1 b_2 d_2

d_1

joe→Person

canCorp→[AND Company r e

b_1 [ALL :Manager Canadian]

KB [FILLS :Manager joe]]

r b_2

Question KB ⊨ (joe→Canadian)

d_2 e

⇒ S={(joe,[AND Person Canadian]),(canCorp, d_1)}. Now, the procedure terminates because KB ⊨ ([AND Person Canadian]⊑Canadian).

Because KB ⊨ ([AND Person Canadian]⊑Canadian) it follows that

KB ⊨ (joe→Canadian).

A description language

In the case where there are EXISTS terms of the form [EXISTS 1 r], we will use role chains

[AND…[ALL r_1…[AND…[ALL r_k a]…]…]…]

$\sigma = r_1 \cdot \ldots \cdot r_k$ is called a role chain.

If b is a constant and r_1, r_2 roles, then $b \cdot r_1 \cdot r_2$ represents an individual (perhaps unnamed) that is in relation r_2 with an individual that is in relation r_1 with b.

If σ is empty, then $b \cdot \sigma$ is b.

The forward chaining procedure extends by adding two steps:

A description language

in slide 29

1. Construct S a list of pairs (b,d), where b is any constant mentioned in KB and d is the normalized version of the concept [AND $d'_1 \ldots d'_n$] for all d'_i such that $(b \rightarrow d'_n) \in KB$.
2. Find two constants b_1 and b_2 such that $(b_1,d_1) \in S$ and $(b_2,d_2) \in S$, [FILLS r b_2] and [ALL r e] are both components of d_1 but KB $\not\models$ $(d_2 \sqsubseteq e)$.
3. If no b_1 and b_2 can be found, then go to step 4. Otherwise, replace the pair (b_2,d_2) in S by (b_2, d'_2), where d'_2 is the normalized version of [AND d_2 e] and go to step 2.
4. Find a constant b, a role chain σ (possibly empty) and a role r such that $(b \cdot \sigma, d_1) \in S$ and $(b \cdot \sigma \cdot r, d_2) \in S$ (if no such pair exists, take d_2 to be Thing), where [EXISTS 1 r] and [ALL r e] are components of d_1, but KB $\not\models$ $(d_2 \sqsubseteq e)$.
5. If these can be found, remove $(b \cdot \sigma \cdot r, d_2)$ from S (if applicable) and add the pair $(b \cdot \sigma \cdot r, d'_2)$, where d'_2 is the normalized version of [AND d_2 e]; then go to step 2. Otherwise exit.

We start with a property of the individual b·σ and conclude something new about the (unnamed) individual b·σ·r. Eventually, this can lead to new information about a named individual.

A description language

Example 2. Assume that we have in KB the sentence:

b·σ, σ is empty

d_1

r

KB: ellen→ [AND [EXISTS 1 :Child]

e

[ALL :Child [AND [FILLS :Peditrician marianne]

r

[ALL : Peditrician Scandinavian]]

]

]

Question: KB ⊨ (marianne→Scandinavian)

S={(b·σ,d_1)} and (b·σ·r,d_2)=(ellen :Child, Thing)

Because KB ⊭ (Thing⊑e), S becomes

S={(b·σ,d_1),(ellen :Child, [AND [FILLS :Peditrician marianne]

b·σ·r [ALL :Peditrician Scandinavian]])}

From here, we conclude that (marianne→Scandinavian) (case with no EXISTS).

The case of terms of the form [EXISTS n r], n>1 is handled the same as for n=1. There is no need to create n different anonymous individuals because all of them would "produce" the same properties in the forward chaining.

A description language

Taxonomies and classification

Given a concept q, in DL it is common to ask for all of its instances, that is to find all c in KB so that KB $\models$ (c$\rightarrow$q).

Also, it is common to ask for all of the known categories that an individual satisfies. That is to say that given a constant c, we should find all concepts a so that KB $\models$ (c$\rightarrow$a).

When reasoning in DL, we should exploit the hierarchical organization of the concepts, with the most general ones at the top and the more specialized ones further down.

To represent sentences in KB, we use a taxonomy (a treelike data structure) that allows answering queries efficiently (time linear with the depth of the taxonomy, not with its size).

A description language

Obs. Subsumption is a partial order.

The taxonomy have atomic concepts as nodes and edges like

$$\begin{array}{c} a_j \\ \uparrow \\ a_i \end{array}$$

whenever $a_i \sqsubseteq a_j$ and there is no a_k such that $a_i \sqsubseteq a_k \sqsubseteq a_j$.

Each constant c in KB will be linked to the most specific atomic concept a_i such that KB $\vDash (c \rightarrow a_i)$.

Adding some new atomic concept or constant to a taxonomy corresponding to a KB is called classification. It involves creating a link from the new concept or constant to existing ones in the taxonomy.

This process exploits the structure of the taxonomy. We start with the concept Thing and then add incrementally new atomic concepts and constants.

A description language

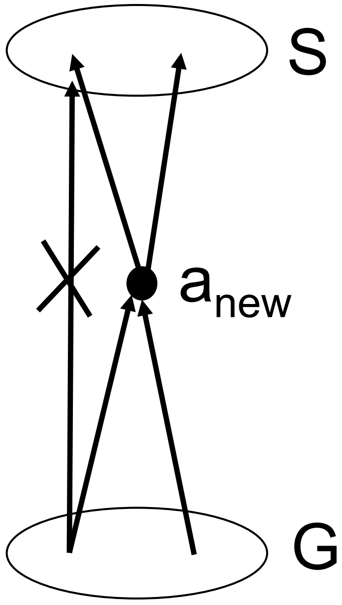
Computing classification

I. Add a sentence ($a_{new} \doteq d$) to the taxonomy, where a_{new} is an atomic concept not appearing anywhere in the KB and d is any concept:

1. Compute S, the most specific subsumers of d

   $S=\{a$ – concept in the taxonomy$|$ KB $\models (d \sqsubseteq a)$, but $\nexists a' \neq a$ so that KB $\models (d \sqsubseteq a')$ and KB $\models (a' \sqsubseteq a)\}$
2. Compute G, the most general subsumees of d

   $G=\{a$ – concept in the taxonomy$|$ KB $\models (a \sqsubseteq d)$, but $\nexists a' \neq a$ so that KB $\models (a' \sqsubseteq d)$ and KB $\models (a \sqsubseteq a')\}$
3. If $\exists a \in S \cap G$ then a_{new} is already in the taxonomy under a different name – no action needed
4. Otherwise remove all links (if any) from concepts in G up to concepts in S
5. Add links from a_{new} up to each concept in S and links from each concept in G up to a_{new}



A description language

6. Handling constants

   Compute C={c constant in taxonomy| $\forall a \in S$, KB $\models (c \rightarrow a)$ and $\nexists a' \in G$ such that KB $\models (c \rightarrow a')$}

   Then for each $c \in C$ we test if KB $\models (c \rightarrow d)$ and if so, we remove the links from c to S and add a single link from c to a_{new}.

II. Add a sentence ($a_{new} \sqsubseteq d$) reduces to adding links from a_{new} to the most specific subsumers of d.

III. Add a sentence ($c_{new} \rightarrow d$) reduces to adding links from c_{new} to the most specific subsumers of d.

A description language

Compute S – the most specific subsumers of d

Start with S={Thing}

For all $a \in S$ if $\exists a'$ so that

$$\begin{tikzcd} a \\ a' \arrow[u] \end{tikzcd}$$

and KB $\models (d \sqsubseteq a')$ then remove a from S and add all a' in S.

Repeat until no element in S has a child that subsumes d.

Compute G – the most general subsumees of d

Start with G=S

If $\exists a \in G$ so that KB $\not\models (a \sqsubseteq d)$, then replace a with all its children (or delete it if it has no children).

Repeat until each element in G is subsumed by d.

A description language

Compute S – the most specific subsumers of d

Start with S={Thing}

For all $a \in S$ if $\exists a'$ so that a

$\uparrow$

a'

and KB $\models (d \sqsubseteq a')$ then remove a from S and add all a' in S.

Repeat until no element in S has a child that subsumes d.

Compute G – the most general subsumees of d

Start with G=S

If $\exists a \in G$ so that KB $\not\models (a \sqsubseteq d)$, then replace a with all its children (or delete it if it has no children).

Repeat until each element in G is subsumed by d.

Answering questions in DL

To find all constants c that satisfy a concept q, we should classify q and then collect all the constants at the fringe of the tree bellow q in the taxonomy.

To find all atomic concepts that are satisfied by a constant c, we go from c up in the taxonomy, collecting all the nodes that can be reached.