Structured Descriptions
From sentences to objects

As we saw with frames, it useful to shift the focus away from the true *sentences* of an application towards the categories of *objects* in the application and their properties.

In frame systems, this was done *procedurally*, and we concentrated on hierarchies of frames as a way of organizing collections of procedures.

In this section, we look at the categories of objects themselves:

- 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
- the relation among the parts of an object can be critical in its being a member of a category
  - e.g. a stack vs. a pile of bricks
Noun phrases

In FOL, all categories and properties of objects are 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.

Intuitively, these predicates have an internal structure and connections to other predicates.

  e.g. A married person must be a person.

These connections hold by definition (by virtue of what the predicates themselves mean), not by virtue of the facts we believe about the world.

In FOL, there is no way to break apart a predicate to see how it is formed from other predicates.

Here we will examine a logic that allows us to have both atomic and non-atomic predicates: a description logic.
Concepts, roles, constants

In a description logic, there are sentences that will be true or false (as in FOL).

In addition, there are 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

These correspond to unary predicates, binary predicates and constants (respectively) in FOL.

  See also: generic frames, slots, and individual frames.
  However: roles can have multiple fillers.

However, unlike in FOL, concepts need not be atomic and can have semantic relationships to each other.

  roles will remain atomic (for now)
The symbols of DL

Three types of non-logical symbols:

- atomic concepts:
  
  Dog, Teenager, GraduateStudent
  
  We include a distinguished concept: Thing

- roles: (all are atomic)
  
  :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: $\Xi$, $\therefore$, and $\rightarrow$
The syntax of DL

The set of concepts is the least set satisfying:

- Every atomic concept is a concept.
- If $r$ is a role and $d$ is a concept, then $[\text{ALL } r \ d]$ is a concept.
- If $r$ is a role and $n$ is an integer, then $[\text{EXISTS } n \ r]$ is a concept.
- If $r$ is a role and $c$ is a constant, then $[\text{FILLS } r \ c]$ is a concept.
- If $d_1, ..., d_k$ are concepts, then so is $[\text{AND } d_1, ..., d_k]$.

Three types of sentences in DL:

- If $d$ and $e$ are concepts, then $(d \equiv e)$ is a sentence.
- If $d$ and $e$ are concepts, then $(d \equiv e)$ is a sentence.
- If $d$ is a concept and $c$ is a constant, then $(c \rightarrow d)$ is a sentence.
The meaning of 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:

- \([\text{EXISTS } n \ r]\) describes those individuals that stand in relation \(r\) to at least \(n\) other individuals
- \([\text{FILLS } r \ c]\) describes those individuals that stand in the relation \(r\) to the individual denoted by \(c\)
- \([\text{ALL } r \ d]\) describes those individuals that stand in relation \(r\) only to individuals that are described by \(d\)
- \([\text{AND } d_1 \ldots d_k]\) describes those individuals that are described by all of the \(d_i\).

For example:

\[
\text{[AND } \text{Company} \\
\text{[EXISTS 7 :Director]} \\
\text{[ALL :Manager [AND Woman [FILLS :Degree PhD]]]} \\
\text{[FILLS :MinSalary $24.00/hour]]}
\]

“a company with at least 7 directors, whose managers are all women with PhDs, and whose min salary is $24/hr”
A DL knowledge base

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

• give names to definitions

  e.g. (FatherOfDaughters ⊑
       [AND Male [EXISTS 1 :Child]
       [ALL :Child Female]])

  “A FatherOfDaughters is precisely
  a male with at least one child and
  all of whose children are female”

• give names to partial definitions

  e.g. (Dog ⊑ [AND Mammal Pet
           CarnivorousAnimal
           [FILLS :VoiceCall barking]])

  gives necessary but not sufficient conditions

  “A dog is among other things a
  mammal that is a pet and a
carnivorous animal whose voice
  call includes barking”

• assert properties of individuals

  e.g. (joe ⊘
       [AND FatherOfDaughters Surgeon]])

  “Joe is a FatherOfDaughters and
  a Surgeon”

Other types of DL sentences are typically not used in a KB.

  e.g. ( [AND Rational Animal] ⊙ [AND Featherless Biped])
Formal semantics

Interpretation \( \mathcal{I} = \langle D, I \rangle \) as in FOL, where

- for every constant \( c \), \( I[c] \subseteq D \)
- 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 as subsets of the domain as follows:

- \( I[\text{Thing}] = D \)
- \( I[\text{ALL } r \ d] = \{ x \in D \mid \text{for any } y, \text{ if } <x,y> \in I[r] \text{ then } y \in I[d]\} \)
- \( I[\text{EXISTS } n \ r] = \{ x \in D \mid \text{there are at least } n \ y \text{ such that } <x,y> \in I[r]\} \)
- \( I[\text{FILLS } r \ c] = \{ x \in D \mid <x,I[c]> \in I[r]\} \)
- \( I[\text{AND } d_1 \ldots d_k] = I[d_1] \cap \ldots \cap I[d_k] \)

A sentence of DL will then be true or false as follows:

- \( \mathcal{I} \models (d \equiv e) \) iff \( I[d] \subseteq I[e] \)
- \( \mathcal{I} \models (d \equiv e) \) iff \( I[d] = I[e] \)
- \( \mathcal{I} \models (c \rightarrow e) \) iff \( I[c] \subseteq I[e] \)
Entailment and reasoning

Entailment in DL is defined as in FOL:

A set of DL sentences $S$ entails a sentence $\alpha$ (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:

1. determining if $KB \models (c \rightarrow e)$
   whether a named individual satisfies a certain description

2. determining if $KB \models (d \equiv e)$
   whether one description is subsumed by another

the other case, $KB \models (d \equiv e)$ reduces to $KB \models (d \equiv e)$ and $KB \models (d \equiv e)$
Entailment vs. validity

In some cases, an entailment will hold because the sentence in question is valid.

- ([AND Doctor Female]  \equiv  Doctor)
- ([FILLS :Child sue]  \equiv  [EXISTS 1 :Child])
- (john \rightarrow [ALL :Hobby Thing])

But in most other cases, the entailment depends on the sentences in the KB.

For example,

([AND Surgeon Female]  \equiv  Doctor)

is not valid.

But it is entailed by a KB that contains

(Surgeon  \equiv  [AND Specialist [FILLS :Specialty surgery]])

(Specialist  \equiv  Doctor)
Computing subsumption

We begin with computing subsumption, that is, determining whether or not KB $\models (d \equiv e)$. and therefore whether $d \equiv e$

Some simplifications to the KB:

- we can remove the $(c \rightarrow d)$ assertions from the KB
- we can replace $(d \equiv e)$ in KB by $(d \equiv [\text{AND } e \ a])$, where $a$ is a new atomic concept
- we assume that in the KB for each $(d \equiv e)$, the $d$ is atomic and appears only once on the LHS
- we assume that the definitions in the KB are acyclic vs. cyclic $(d \equiv [\text{AND } e \ f]), \ (e \equiv [\text{AND } d \ g])$

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:
  \[ \text{[AND ... [AND } \, d \, e \, f \, \ldots] \Rightarrow \text{[AND ... } d \, e \, f \, \ldots] \]
• combine the ALL operations with the same role:
  \[ \text{[AND ... [ALL } \, r \, d \ldots [ALL \, r \, e \ldots] \Rightarrow \text{[AND ... [ALL } \, r \, [\text{AND } d \, e] \ldots] \]
• combine the EXISTS operations with the same role:
  \[ \text{[AND ... [EXISTS } \, n_1 \, r \ldots [\text{EXISTS } \, n_2 \, r \ldots] \Rightarrow \text{[AND ... [EXISTS } \, n \, [\text{AND } r] \ldots]} \] (where \( n = \text{Max}(n_1, n_2) \))
• remove a vacuous concept: Thing, [ALL \( r \) \text{ Thing}], [AND]
• remove a duplicate expression

In the end, we end up with a normalized concept of the following form:

\[ \text{[AND } \, a_1 \ldots a_i \Rightarrow \text{[FILLS } \, r_1 \, c_1 \ldots [\text{FILLS } \, r_j \, c_j] \Rightarrow \text{[EXISTS } \, n_1 \, s_1 \ldots [\text{EXISTS } \, n_k \, s_k] \Rightarrow \text{[ALL } \, t_1 \, e_1 \ldots [\text{ALL } \, t_m \, e_m] \right] } \]
Normalization example

[AND Person
  [ALL :Friend Doctor]
  [EXISTS 1 :Accountant]
  [ALL :Accountant [EXISTS 1 :Degree]]
  [ALL :Friend Rich]
  [ALL :Accountant [AND Lawyer [EXISTS 2 :Degree]]]]

[AND Person
  [EXISTS 1 :Accountant]
  [ALL :Friend [AND Rich Doctor]]
  [ALL :Accountant [AND Lawyer [EXISTS 1 :Degree] [EXISTS 2 :Degree]]]]

[AND Person
  [EXISTS 1 :Accountant]
  [ALL :Friend [AND Rich Doctor]]
  [ALL :Accountant [AND Lawyer [EXISTS 2 :Degree]]]]
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 $[\text{AND} \ e_1 \ldots \ e_m]$ subsumes a normalized concept $[\text{AND} \ d_1 \ldots \ d_n]$, we do the following:

For each component $e_j$, check that there is a matching component $d_i$, where

- if $e_j$ is atomic or $[\text{FILLS} \ r \ c]$, then $d_i$ must be identical to it;
- if $e_j = [\text{ EXISTS } 1 \ r]$, then $d_i$ must be $[\text{ EXISTS } n \ r]$ or $[\text{FILLS} \ r \ c]$;
- if $e_j = [\text{ EXISTS } n \ r]$ where $n > 1$, then $d_i$ must be of the form $[\text{ EXISTS } m \ r]$ where $m \geq n$;
- if $e_j = [\text{ALL} \ r \ e']$, then $d_i$ must be $[\text{ALL} \ r \ d']$, 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 $\text{KB} \models (d \equiv e)$. 
Structure matching example