Lecture 12: Unification in FOL
Reading: AIAMA 9.1-9.2

Today’s Schedule:
- Inference by reducing FOL to PL
- FOL Inference Part I: Unification and Lifting
Motivation

Given an FOL KB with facts, e.g.

\[
\text{likes}(\text{bob, movies}), \text{likes}(\text{sally, movies}), \text{likes}(\text{bill, pizza})
\]

and rules, e.g.

\[
\forall x, y, z \ \text{person}(x) \land \text{person}(y) \land \text{likes}(x,z) \land \text{likes}(y,z) \Rightarrow \text{friends}(x,y)
\]

we want to be able to infer answers to questions, e.g.

- \(\text{likes}(x, \text{movies})\) ”who likes movies?”
- \(\text{likes}(\text{bill, } x)\) ”what does bill like?”
- \(\text{friends}(\text{bob, bill})\) ”are bob and bill friends?”

Thus, we need an inference algorithm for FOL.
This and the next two lectures will be devoted to inference methods for FOL. We will look at two approaches:

- Convert FOL to a PL and use PL resolution
- Lifting Modus Ponens

There is a lot here and we don’t have time to do it full justice. We will examine in some detail a subset of inference methods that are useful in many practical situations.
Reducing FOL to PL: ground terms

The reduce the FOL to a PL, we need to address the quantifiers.

First, define a *ground term* as a term without variables

\[
groundterm = \text{constant} \mid \text{Function}(\text{groundterm}, ..)
\]

In the example above there are 5 ground terms: bob, sally, bill, pizza, movies.
Reducing FOL to PL: substitutions

Next, define a *substitution rule* \( \text{SUBST}(\theta, \alpha) \) as the application of (possible partial) substitutions in the set \( \theta \) for variables in the sentence \( \alpha \).

Examples:

\[
\alpha = \text{friends}(x, y), \quad \theta = \{x/bob, y/bill\}
\]
\[
\text{SUBST}(\theta, \alpha) = \text{friends}(bob, bill)
\]

\[
\alpha = \text{friends}(x, y), \quad \theta = \{x/sally\}
\]
\[
\text{SUBST}(\theta, \alpha) = \text{friends}(sally, y)
\]

\[
\alpha = \text{friends}(x, y), \quad \theta = \{x/sally, y/z\}
\]
\[
\text{SUBST}(\theta, \alpha) = \text{friends}(sally, z)
\]
Universal Instantiation

The rule of *Universal Instantiation* says that we can infer any sentence with universally quantified variables replaced by ground terms.

In our example above

\[ \forall x, y, z \ person(x) \land person(y) \land \text{likes}(x, z) \land \text{likes}(y, z) \Rightarrow \text{friends}(x, y) \]

each of the variables \( x, y, z \) can be replaced by by one of the five ground terms: bob, sally, bill, pizza, movies

\[
\theta = \{x/\text{bob}, y/\text{bob}, z/\text{bob}\} \cup \\
\{x/\text{bob}, y/\text{bob}, z/\text{sally}\} \cup \\
\ldots \\
\{x/\text{bob}, y/\text{bob}, z/\text{movies}\} \cup \\
\ldots
\]
Existential Instantiation

To deal with $\exists$ we invoke *Existential Instantiation*.

$$\exists x \ P(x) \ becomes \ P(C_1)$$

where $C_1$ is a unique constant not in the KB referred to as a *skolem constant*.

Example: "there exists a mammal that flies"

$$\exists x \ \text{mammal}(x) \land \text{flies}(x) \ becomes \ \text{mammal}(C_1) \land \text{flies}(C_1)$$

where $C_1$ could refer semantically to a bat for example.
Existential Instantiation is more complicated when there are predicates or functions that depend on more than one variable, e.g.

\[ \forall x \exists y \, P(x, y) \]

Since the skolemization constant \textbf{depends} on \( x \). This require a \textit{skolem function}, a function that maps \( x \) to constants such that the predicate holds. For example: ”everyone has a parent”

\[ \forall x \exists y \, parent(y, x) \]

becomes

\[ \forall x \, parent(F_1(x), x) \]

where \( F_1 \) is the skolem function that maps people to their parents.
Reducing FOL to PL

So Universal and Existential Instantiation remove all variables. We treat what remains as propositional symbols, e.g.

\[ \text{likes(bob, sally)} \] is treated as a PL symbol \text{likes\_bob\_sally} 

We can then proceed with PL inference, for example using resolution. It can be shown that this process of *propositionalizing* the FOL KB leads to a PL KB such that any query entailed by the FOL KB is entailed by the PL KB.

However, there are some practical problems:

- the size of the KB expands considerably
- functions are recursive, so universal instantiation leads to an infinite number of recursions.
The second approach to FOL inference

This approach treats inference as *subject to a substitution*. Consider Modus Ponens:

\[
\frac{\alpha \Rightarrow \beta, \alpha}{\beta}
\]

Let \( \alpha, \beta \) have variables, say \( x \), and \( A \) be a constant. The *lifted* version on Modus Ponens is:

\[
\frac{\alpha(x) \Rightarrow \beta(x), \alpha(A)}{\text{SUBST}(\theta, \beta)}
\]

where \( \theta = \{ x/A \} \).
Unification

The lifted modus ponens

$$\frac{\alpha(x) \implies \beta(x), \alpha(A)}{\text{SUBST}(\theta, \beta)}$$

requires that we be able to find substitutions, given two sentences (the premise with the variable and the fact). This process is called unification. Given two sentences $p$ and $q$

$$\text{unify}(p, q) = \theta \text{ s.t. } \text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$$

Example: Suppose $p = \text{likes}(\text{bob}, \text{movies})$ and $q = \text{likes}(x, \text{movies})$

$$\text{SUBST}(\theta, \text{likes}(\text{bob}, \text{movies})) = \text{SUBST}(\theta, \text{likes}(x, \text{movies}))$$

$\theta = \{\}$

$\theta = \{x/\text{bob}\}$
Some comments before the unification algorithm

- If two sentences in the KB reuse the same variable name then the names clash. Example
  
  \[ p = \text{likes}(x, \text{movies}) \]
  
  \[ q = \text{likes}(\text{bill}, x) \]

  \( x \) can’t be both movies and bill, it is really two variables.

  This \textit{standardizing apart}.

- Unification is not unique, e.g. given

  \[ p = \text{likes}(\text{bill}, x) \text{ and } q = \text{likes}(y, z) \]

  \( \text{unify}(p,q) \) could return \( y/bill, z/x \) or \( y/bill, x/bill, z/bill \).

  The latter can be obtained by the former by another substitution (\( x/bill \)). We want the former, the most general unification (\textbf{MGU}).
The Unification Algorithm

function UNIFY\( (x, y, \theta) \) returns a substitution to make \( x \) and \( y \) identical
inputs: \( x \), a variable, constant, list, or compound expression
\( y \), a variable, constant, list, or compound expression
\( \theta \), the substitution built up so far (optional, defaults to empty)

if \( \theta = \text{failure} \) then return failure
else if \( x = y \) then return \( \theta \)
else if VARIABLE\(? (x) \) then return UNIFY-VAR\( (x, y, \theta) \)
else if VARIABLE\(? (y) \) then return UNIFY-VAR\( (y, x, \theta) \)
else if COMPOUND\(? (x) \) and COMPOUND\(? (y) \) then
    return UNIFY\( (x.ARGs, y.ARGs, UNIFY(x.OP, y.OP, \theta)) \)
else if LIST\(? (x) \) and LIST\(? (y) \) then
    return UNIFY\( (x.REST, y.REST, \text{UNIFY}(x.FIRST, y.FIRST, \theta)) \)
else return failure

function UNIFY-VAR\( (\text{var}, x, \theta) \) returns a substitution

if \( \{ \text{var}/\text{val} \} \in \theta \) then return UNIFY\( (\text{val}, x, \theta) \)
else if \( \{ x/\text{val} \} \in \theta \) then return UNIFY\( (\text{var}, \text{val}, \theta) \)
else if OCCUR-CHECK\(? (\text{var}, x) \) then return failure
else return add \( \{ \text{var}/x \} \) to \( \theta \)
Write down a FOL sentence capturing the following knowledge from the wumpus world: In any square, if we smell a stench, there is a wumpus in an adjacent square.

Given your sentence in the previous question and a new sentence stench(square(2,2)), what is the substitution list that unifies the two sentences?
Next Actions

- Reading on Forward/Backward Chaining: AIAMA 9.3-9.4
- Take warmup before noon on Thursday 3/5.

No class meeting on 3/17, but there is a reading and I will post a pre-recorded lecture.