KRR3: Inference in First-order logic

Yannick Pencolé
Yannick.Pencole@anu.edu.au

14 Mar 2005
<table>
<thead>
<tr>
<th>Year</th>
<th>Person</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>450 BC</td>
<td>Stoics</td>
<td>Propositional logic, inference (maybe)</td>
</tr>
<tr>
<td>322 BC</td>
<td>Aristotle</td>
<td>“Syllogisms” (inference rules), quantifiers</td>
</tr>
<tr>
<td>1565</td>
<td>Cardano</td>
<td>Probability theory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Propositional logic + uncertainty)</td>
</tr>
<tr>
<td>1847</td>
<td>Boole</td>
<td>Propositional logic (again)</td>
</tr>
<tr>
<td>1879</td>
<td>Frege</td>
<td>First-order logic</td>
</tr>
<tr>
<td>1922</td>
<td>Wittgenstein</td>
<td>Proof by truth tables</td>
</tr>
<tr>
<td>1930</td>
<td>Gödel</td>
<td>existential complete algorithm for FOL :-(</td>
</tr>
<tr>
<td>1930</td>
<td>Herbrand</td>
<td>Complete algorithm for FOL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Reduce to propositional)</td>
</tr>
<tr>
<td>1931</td>
<td>Gödel</td>
<td>negation existential complete algorithm for arithmetic :-(</td>
</tr>
<tr>
<td>1960</td>
<td>Davis/Putnam</td>
<td>“Practical” algorithm for propositional logic</td>
</tr>
<tr>
<td>1965</td>
<td>Robinson</td>
<td>“Practical” algorithm for FOL — resolution</td>
</tr>
</tbody>
</table>
Outline

1. Reduction to propositional inference
2. Unification and lifting
3. Knowledge base: an example
4. Forward chaining
Outline

1. Reduction to propositional inference
2. Unification and lifting
3. Knowledge base: an example
4. Forward chaining
Universal instantiation (UI)

Definition

Inference rule: Every instantiation of a universally quantified sentence is entailed by it.

\[ \forall v \alpha \]

\[ \text{SUBST(}\{v/g\}, \alpha) \]

for any variable \( v \) and ground term \( g \) (term without variable)

Example

\[ \forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x) \] yields

\[ King(John) \land Greedy(John) \Rightarrow Evil(John) \quad [g = John] \]
\[ King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard) \quad [g = Richard] \]
\[ King(Father(John)) \land Greedy(Father(John)) \Rightarrow Evil(Father(John)) \quad [g = Father(John)] \]
Existential instantiation (EI)

Definition

Inference rule: For any sentence $\alpha$, variable $v$, and constant symbol $k$ that does not appear elsewhere in the knowledge base:

$$\exists v \alpha \quad \frac{}{\text{SUBST}\left(\{v/k\}, \alpha\right)}$$

Example

$$\exists x \ Crown(x) \land OnHead(x, John) \text{ yields } Crown(C_1) \land OnHead(C_1, John) \quad [k = C_1]$$

provided $C_1$ is a new constant symbol, called a Skolem constant.
EI and UI

UI

UI can be applied several times to add new sentences; the new KB is logically equivalent to the previous one.

\[ KB = \{ \forall x \ P(x) \} \]
\[ newKB = \{ \forall x \ P(x), P(Richard) \} \ldots \]

EI

EI can be applied once to replace the existential sentence; the new KB is not equivalent to the previous one!

\[ KB = \{ \exists x \ P(x) \} \]
\[ newKB = \{ P(SkolemConstant) \} \ldots \]
Example

Suppose the KB contains just the following:

\[ \forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x) \]

- King(John)
- Greedy(John)
- Brother(Richard, John)

Instantiating the universal sentence in all possible ways, we have:

- King(John) \land Greedy(John) \Rightarrow Evil(John)
- King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)
- King(John)
- Greedy(John)
- Brother(Richard, John)

Propositionalisation

The new KB is propositionalised: proposition symbols are

King(John), Greedy(John), Evil(John), King(Richard) \cdots
Theorem of Herbrand (1930)

A sentence $\alpha$ is entailed by an FOL KB if and only if it is entailed by a finite subset of the propositional KB.

Complete algorithm

For $n = 0$ to $maxDepth$
- create a $KB_{prop,n}$ by reduction with depth-$m$ terms ($m = 1 \ldots n$)
- if $\alpha_{prop}$ is entailed by $KB_{prop,n}$ (i.e. $KB_{prop,n} \models \alpha_{prop}$) then STOP.

Problem with the algorithm

If $KB \models \alpha$ then the algorithm stops but if $KB \not\models \alpha$, the algorithm does not stop.
If $KB$ contains a function then the terms can have infinite depth ($maxDepth = \infty$): $Father(Father(Father(\ldots Father(John)\ldots))$)

Can we solve the problem?

NO. Entailment in FOL is semi-decidable Turing(1936) Church(1936)
Problems with propositionalization

Propositionalization seems to generate lots of irrelevant sentences.

Example

From $KB = \{ $
\forall x \ King(x) \land Greedy(x) \implies Evil(x),$
$King(John),$
$\forall y \ Greedy(y),$
$Brother(Richard, John)\}$

it seems obvious that $Evil(John)$, but propositionalization produces lots of facts such as $Greedy(Richard)$ that are irrelevant.

With $p$ $k$-ary predicates and $n$ constants, there are $p \cdot n^k$ instantiations. With function symbols, it gets much much much worse!
Outline

1. Reduction to propositional inference
2. Unification and lifting
3. Knowledge base: an example
4. Forward chaining
Inference rules for FOL

We can get the inference immediately on the FOL KB.

Idea

We use rules from propositional logic that are lifted like the Generalised Modus Ponens. Then we can update FC, BC and Resolution for FOL :-)

Implementation of the idea

The problem is the instantiation of the variables. We need some new operators:

1. Substitution (SUBST)
2. Unification (UNIFY)

...to define the inference rules and to choose “clever” instantiations
Unification

Definition

Let $p, q$ be two sentences of FOL, the result of the Unification is:

$$\text{UNIFY}(p, q) = \theta$$

where $\theta$ is a substitution on variables of $\alpha, \beta$ such that:

$$\text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$$
Unification

Definition
Let \( p, q \) be two sentences of FOL, the result of the Unification is:

\[
\text{UNIFY}(p, q) = \theta
\]

where \( \theta \) is a substitution on variables of \( \alpha, \beta \) such that:

\[
\text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)
\]

Example

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( \theta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows(John, x)</td>
<td>Knows(John, Jane)</td>
<td></td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, Bill)</td>
<td></td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, Mother(y))</td>
<td></td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(x, Elizabeth)</td>
<td></td>
</tr>
</tbody>
</table>
Definition

Let $p, q$ be two sentences of FOL, the result of the **Unification** is:

$$\text{UNIFY}(p, q) = \theta$$

where $\theta$ is a substitution on variables of $\alpha, \beta$ such that:

$$\text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$$

Example

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows(John, x)</td>
<td>Knows(John, Jane)</td>
<td>${x/Jane}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, Bill)</td>
<td></td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, Mother(y))</td>
<td></td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(x, Elizabeth)</td>
<td></td>
</tr>
</tbody>
</table>
Unification

**Definition**

Let \( p, q \) be two sentences of FOL, the result of the **Unification** is:

\[
\text{UNIFY}(p, q) = \theta
\]

where \( \theta \) is a substitution on variables of \( \alpha, \beta \) such that:

\[
\text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)
\]

**Example**

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( \theta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Knows}(\text{John}, x) )</td>
<td>( \text{Knows}(\text{John}, \text{Jane}) )</td>
<td>{x/\text{Jane}}</td>
</tr>
<tr>
<td>( \text{Knows}(\text{John}, x) )</td>
<td>( \text{Knows}(y, \text{Bill}) )</td>
<td>{x/\text{Bill}, y/\text{John}}</td>
</tr>
<tr>
<td>( \text{Knows}(\text{John}, x) )</td>
<td>( \text{Knows}(y, \text{Mother}(y)) )</td>
<td></td>
</tr>
<tr>
<td>( \text{Knows}(\text{John}, x) )</td>
<td>( \text{Knows}(x, \text{Elizabeth}) )</td>
<td></td>
</tr>
</tbody>
</table>
Unification

**Definition**

Let $p, q$ be two sentences of FOL, the result of the Unification is:

\[ \text{UNIFY}(p, q) = \theta \]

where $\theta$ is a substitution on variables of $\alpha, \beta$ such that:

\[ \text{SUBST}(\theta, p) = \text{SUBST}(\theta, q) \]

**Example**

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows(John, x)</td>
<td>Knows(John, Jane)</td>
<td>${x/\text{Jane}}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, Bill)</td>
<td>${x/\text{Bill}, y/\text{John}}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, Mother(y))</td>
<td>${y/\text{John}, x/\text{Mother(John)}}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(x, Elizabeth)</td>
<td></td>
</tr>
</tbody>
</table>
Unification

Definition

Let $p$, $q$ be two sentences of FOL, the result of the Unification is:

$$\text{UNIFY}(p, q) = \theta$$

where $\theta$ is a substitution on variables of $\alpha, \beta$ such that:

$$\text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$$

Example

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Knows}(John, x)$</td>
<td>$\text{Knows}(John, Jane)$</td>
<td>${x/Jane}$</td>
</tr>
<tr>
<td>$\text{Knows}(John, x)$</td>
<td>$\text{Knows}(y, Bill)$</td>
<td>${x/Bill, y/John}$</td>
</tr>
<tr>
<td>$\text{Knows}(John, x)$</td>
<td>$\text{Knows}(y, \text{Mother}(y))$</td>
<td>${y/John, x/\text{Mother}(John)}$</td>
</tr>
<tr>
<td>$\text{Knows}(John, x)$</td>
<td>$\text{Knows}(x, Elizabeth)$</td>
<td>fail</td>
</tr>
</tbody>
</table>
Unification

Definition

Let $p, q$ be two sentences of FOL, the result of the Unification is:

$$\text{UNIFY}(p, q) = \theta$$

where $\theta$ is a substitution on variables of $\alpha, \beta$ such that:

$$\text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$$

Example

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows(John, $x$)</td>
<td>Knows(John, Jane)</td>
<td>{ $x$/Jane}</td>
</tr>
<tr>
<td>Knows(John, $x$)</td>
<td>Knows($y$, Bill)</td>
<td>{ $x$/Bill, $y$/John}</td>
</tr>
<tr>
<td>Knows(John, $x$)</td>
<td>Knows($y$, Mother($y$))</td>
<td>{ $y$/John, $x$/Mother(John)}</td>
</tr>
<tr>
<td>Knows(John, $x$)</td>
<td>Knows($x$, Elizabeth)</td>
<td>fail</td>
</tr>
</tbody>
</table>

Standardizing apart eliminates overlap of variables (renaming of variables)

Knows($z_{17}$, Elizabeth)
**Unification**

**Definition**

Let \( p, q \) be two sentences of FOL, the result of the *Unification* is:

\[
\text{UNIFY}(p, q) = \theta
\]

where \( \theta \) is a substitution on variables of \( \alpha, \beta \) such that:

\[
\text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)
\]

**Example**

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( \theta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows(John, x)</td>
<td>Knows(John, Jane)</td>
<td>{x/Jane}</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, Bill)</td>
<td>{x/Bill, y/John}</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, Mother(y))</td>
<td>{y/John, x/Mother(John)}</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(x, Elizabeth)</td>
<td>fail</td>
</tr>
</tbody>
</table>

*Standardizing apart* eliminates overlap of variables (renaming of variables)

\begin{align*}
\text{Knows}(z_{17}, Elizabeth) \\
\text{Knows}(John, x) & \quad \text{Knows}(z_{17}, Elizabeth) & \{x/Elizabeth, z_{17}/John\}
\end{align*}
Generalized Modus Ponens (GMP)

**Definition**

\[
p_1', \ p_2', \ldots, \ p_n', \ (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \Rightarrow \text{SUBST}(\theta, q)
\]

where \( \theta = \text{UNIFY}(p_i, p'_i) \) for all \( i \).

Variables are universally quantified.

**Example**

\[
\text{King}(John), \ \text{Greedy}(y) \quad (\text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)) \Rightarrow \text{SUBST}(\theta, \text{Evil}(x))
\]

What is \( \theta \)? What is \( \text{SUBST}(\theta, \text{Evil}(x)) \)?
Generalized Modus Ponens (GMP)

**Definition**

\[ p_1', \ p_2', \ldots, \ p_n', \ (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \]

\[ \text{SUBST}(\theta, q) \]

where \( \theta = \text{UNIFY}(p_i, p_i') \) for all \( i \).

Variables are universally quantified.

**Example**

\[ \text{King}(John), \ \text{Greedy}(y) \ (\text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)) \]

\[ \text{SUBST}(\theta, \text{Evil}(x)) \]

What is \( \theta \)? What is \( \text{SUBST}(\theta, \text{Evil}(x)) \)?

\( \theta = \text{UNIFY}(\text{King}(John), \text{King}(x)) = \text{UNIFY}(\text{Greedy}(y), \text{Greedy}(x)) \)

\( \theta = \{x/John, \ y/John\} \)
Generalized Modus Ponens (GMP)

**Definition**

\[
p_1', \ p_2', \ldots, \ p_n', \ (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \quad \text{SUBST}(\theta, q)
\]

where \( \theta = \text{UNIFY}(p_i, p'_i) \) for all \( i \).
Variables are universally quantified.

**Example**

\[
\text{King}(John), \ \text{Greedy}(y) \quad (\text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x))
\]

\[
\text{SUBST}(\theta, \text{Evil}(x))
\]

What is \( \theta \)? What is \( \text{SUBST}(\theta, \text{Evil}(x)) \)?

\[
\theta = \text{UNIFY} (\text{King}(John), \text{King}(x)) = \text{UNIFY} (\text{Greedy}(y), \text{Greedy}(x))
\]

\[
\theta = \{ x/John, y/John \}
\]

\[
\text{SUBST}(\theta, \text{Evil}(x)) = \text{Evil}(John)
\]
GMP is sound

Property

GMP is sound. We have:

\[ p_1', \ p_2', \ \ldots, \ p_n' \ (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \models \text{SUBST}(\theta, q) \]

provided \( \theta = \text{UNIFY}(p_i, p_i') \) for all \( i \).

Proof idea

1. for a sentence \( p \) (with variables universally quantified), we have for all \( \theta \):

\[ p \models \text{SUBST}(\theta, p) \]

2. because of 1 and UI, from \( p_1', \ p_2', \ \ldots, \ p_n' \) we infer

\[ \text{SUBST}(\theta, p_1') \land \ldots \land \text{SUBST}(\theta, p_n') \]

3. because of 1 and UI, from \( p_1 \land p_2 \land \ldots \land p_n \Rightarrow q \) we infer

\[ \text{SUBST}(\theta, p_1) \land \ldots \land \text{SUBST}(\theta, p_n) \Rightarrow \text{SUBST}(\theta, q) \]

4. If \( \theta = \text{UNIFY}(p_i, p_i') \) for all \( i \) then from 2 and 3 we infer \( \text{SUBST}(\theta, q) \) \( \square \).
Outline

1. Reduction to propositional inference
2. Unification and lifting
3. Knowledge base: an example
4. Forward chaining
Example

“One says that a person who gives good lectures about FOL to students is a good teacher. This group of people, studying at the ANU, have very good lectures about Logic and all of those lectures are given by Yannick who is a person.”

We must prove that “Yannick is a good teacher”
Knowledge base: an example

Example

"...a person who gives good lectures about FOL to students is a good teacher...":
Example

"...a person who gives good lectures about FOL to students is a good teacher...":

\[ \forall x \forall y \forall z \text{ Person}(x) \land \text{ Gives}(x, y, z) \land \text{ GoodLecturesFOL}(y) \land \text{ Students}(z) \Rightarrow \text{ GoodTeacher}(x) \]
Knowledge base: an example

Example

“...a person who gives good lectures about FOL to students is a good teacher...”:

\[ \forall x \forall y \forall z \ Person(x) \land Gives(x, y, z) \land GoodLecturesFOL(y) \land Students(z) \Rightarrow GoodTeacher(x) \]

“This group of people.... have very good lectures about Logic”:
Knowledge base: an example

Example

“...a person who gives good lectures about FOL to students is a good teacher...”:
\[ \forall x \forall y \forall z \; \text{Person}(x) \land \text{Gives}(x, y, z) \land \text{GoodLecturesFOL}(y) \land \text{Students}(z) \Rightarrow \text{GoodTeacher}(x) \]

“This group of people.... have very good lectures about Logic”:
\[ \exists x \; \text{Have}(\text{People, } x) \land \text{GoodLecturesLogic}(x) \]
“...a person who gives good lectures about FOL to students is a good teacher...”:
\[ \forall x \forall y \forall z \ Person(x) \land Gives(x, y, z) \land GoodLecturesFOL(y) \land Students(z) \Rightarrow GoodTeacher(x) \]

“This group of people.... have very good lectures about Logic”:
\[ \exists x \ Have(People, x) \land GoodLecturesLogic(x) \]

“...all of their lectures are given by Yannick...”:
Example

“...a person who gives good lectures about FOL to students is a good teacher...”:
\[ \forall x \forall y \forall z \ Person(x) \land \ Gives(x, y, z) \land \ GoodLecturesFOL(y) \land \ Students(z) \Rightarrow \ GoodTeacher(x) \]

“This group of people.... have very good lectures about Logic”:
\[ \exists x \ \text{Have}(People, x) \land \ GoodLecturesLogic(x) \]

“...all of their lectures are given by Yannick...”:
\[ \forall x \ \text{GoodLecturesLogic}(x) \land \ \text{Have}(People, x) \Rightarrow \ \text{Gives}(\text{Yannick}, x, \text{People}) \]
“...a person who gives good lectures about FOL to students is a good teacher...”:
\[ \forall x \forall y \forall z \text{ Person}(x) \land \text{ Gives}(x, y, z) \land \text{ GoodLecturesFOL}(y) \land \text{ Students}(z) \implies \text{ GoodTeacher}(x) \]

“This group of people.... have very good lectures about Logic”:
\[ \exists x \text{ Have}(\text{ People}, x) \land \text{ GoodLecturesLogic}(x) \]

“...all of their lectures are given by Yannick...”:
\[ \forall x \text{ GoodLecturesLogic}(x) \land \text{ Have}(\text{ People}, x) \implies \text{ Gives}(\text{ Yannick}, x, \text{ People}) \]

“..This group of people, studying at the ANU”: 
Knowledge base: an example

Example

“...a person who gives good lectures about FOL to students is a good teacher...”:
\[ \forall x \forall y \forall z \ Person(x) \land \ Gives(x, y, z) \land \ GoodLecturesFOL(y) \land \ Students(z) \implies \ GoodTeacher(x) \]

“This group of people.... have very good lectures about Logic”:
\[ \exists x \ Have(\ People, x) \land \ GoodLecturesLogic(x) \]

“...all of their lectures are given by Yannick...”:
\[ \forall x \ GoodLecturesLogic(x) \land \ Have(\ People, x) \implies \ Gives(\ Yannick, x, People) \]

“..This group of people, studying at the ANU”:
\[ Study(\ People, \ ANU) \]
Knowledge base: an example

Example

“...a person who gives good lectures about FOL to students is a good teacher...”:
\[ \forall x \forall y \forall z \ Person(x) \land Gives(x, y, z) \land GoodLecturesFOL(y) \land Students(z) \Rightarrow GoodTeacher(x) \]

“This group of people.... have very good lectures about Logic”:
\[ \exists x \ Have(People, x) \land GoodLecturesLogic(x) \]

“...all of their lectures are given by Yannick...”:
\[ \forall x \ GoodLecturesLogic(x) \land Have(People, x) \Rightarrow Gives(Yannick, x, People) \]

“..This group of people, studying at the ANU”:
\[ Study(People, ANU) \]

“..Yannick who is a person..”:
Knowledge base: an example

Example

“...a person who gives good lectures about FOL to students is a good teacher...”:
\[ \forall x \forall y \forall z \ Person(x) \land \ Gives(x, y, z) \land \ GoodLecturesFOL(y) \land \ Students(z) \Rightarrow \ GoodTeacher(x) \]

“This group of people.... have very good lectures about Logic”:
\[ \exists x \ Have(People, x) \land \ GoodLecturesLogic(x) \]

“...all of their lectures are given by Yannick...”:
\[ \forall x \ GoodLecturesLogic(x) \land \ Have(People, x) \Rightarrow \ Gives(Yannick, x, People) \]

“..This group of people, studying at the ANU”:
\[ \text{Study(People, ANU)} \]

“..Yannick who is a person..”:
\[ \text{Person(Yannick)} \]
Knowledge base: an example

Example

“...a person who gives good lectures about FOL to students is a good teacher...”:

\[
\forall x \forall y \forall z \ \text{Person}(x) \land \text{Gives}(x, y, z) \land \text{GoodLecturesFOL}(y) \land \text{Students}(z) \Rightarrow \text{GoodTeacher}(x)
\]

“This group of people.... have very good lectures about Logic”:

\[
\exists x \ \text{Have}(\text{People}, x) \land \text{GoodLecturesLogic}(x)
\]

“...all of their lectures are given by Yannick...”:

\[
\forall x \ \text{GoodLecturesLogic}(x) \land \text{Have}(\text{People}, x) \Rightarrow \text{Gives}(\text{Yannick}, x, \text{People})
\]

“..This group of people, studying at the ANU”:

\[
\text{Study}(\text{People}, \text{ANU})
\]

“..Yannick who is a person..”:

\[
\text{Person}(\text{Yannick})
\]

People that study at the ANU, are students
Example

“...a person who gives good lectures about FOL to students is a good teacher...”:
\[ \forall x \forall y \forall z \text{Person}(x) \land \text{Gives}(x,y,z) \land \text{GoodLecturesFOL}(y) \land \text{Students}(z) \Rightarrow \text{GoodTeacher}(x) \]

“This group of people... have very good lectures about Logic”:
\[ \exists x \text{Have}(\text{People}, x) \land \text{GoodLecturesLogic}(x) \]

“...all of their lectures are given by Yannick...”:
\[ \forall x \text{GoodLecturesLogic}(x) \land \text{Have}(\text{People}, x) \Rightarrow \text{Gives}(\text{Yannick}, x, \text{People}) \]

“.This group of people, studying at the ANU”:
\[ \text{Study}(\text{People, ANU}) \]

“.Yannick who is a person...”:
\[ \text{Person}(\text{Yannick}) \]

People that study at the ANU, are students
\[ \forall x \text{Study}(x, \text{ANU}) \Rightarrow \text{Students}(x) \]
Knowledge base: an example

Example

“...a person who gives good lectures about FOL to students is a good teacher...”:

$$\forall x \forall y \forall z \text{ Person}(x) \land \text{ Gives}(x, y, z) \land \text{ GoodLecturesFOL}(y) \land \text{ Students}(z) \Rightarrow \text{ GoodTeacher}(x)$$

“This group of people.... have very good lectures about Logic”:

$$\exists x \ \text{ Have}(\text{ People}, x) \land \text{ GoodLecturesLogic}(x)$$

“...all of their lectures are given by Yannick...”:

$$\forall x \ \text{ GoodLecturesLogic}(x) \land \text{ Have}(\text{ People}, x) \Rightarrow \text{ Gives}(\text{ Yannick}, x, \text{ People})$$

“..This group of people, studying at the ANU”:

$$\text{ Study}(\text{ People}, \text{ ANU})$$

“..Yannick who is a person..”:

$$\text{ Person}(\text{ Yannick})$$

People that study at the ANU, are students

$$\forall x \ \text{ Study}(x, \text{ ANU}) \Rightarrow \text{ Students}(x)$$

If the lectures about Logic are good, the lectures about FOL are good:
Knowledge base: an example

Example

“...a person who gives good lectures about FOL to students is a good teacher...”:
\( \forall x \forall y \forall z \text{Person}(x) \land \text{Gives}(x, y, z) \land \text{GoodLecturesFOL}(y) \land \text{Students}(z) \Rightarrow \text{GoodTeacher}(x) \)

“This group of people.... have very good lectures about Logic”:
\( \exists x \text{Have}(\text{People}, x) \land \text{GoodLecturesLogic}(x) \)

“...all of their lectures are given by Yannick...”:
\( \forall x \text{GoodLecturesLogic}(x) \land \text{Have}(\text{People}, x) \Rightarrow \text{Gives}(\text{Yannick}, x, \text{People}) \)

“.This group of people, studying at the ANU”:
\( \text{Study} (\text{People, ANU}) \)

“.Yannick who is a person.”:
\( \text{Person}(\text{Yannick}) \)

People that study at the ANU, are students
\( \forall x \text{Study}(x, \text{ANU}) \Rightarrow \text{Students}(x) \)

If the lectures about Logic are good, the lectures about FOL are good:
\( \forall x \text{GoodLecturesLogic}(x) \Rightarrow \text{GoodLecturesFOL}(x) \)
Outline

1. Reduction to propositional inference
2. Unification and lifting
3. Knowledge base: an example
4. Forward chaining
Restriction: Definite clauses

Definition
As said previously, a Horn clause is
- a predicate $P(\cdot)$, or
- something like $P_1(\cdot) \cdots P_n(\cdot) \Rightarrow C(\cdot)$
- $\equiv \neg P_1(\cdot) \lor \cdots \lor \neg P_n(\cdot) \lor C(\cdot)$

($P_i(\cdot)$ is a premise and $C(\cdot)$ the conclusion)
The conclusion $C(\cdot)$ can simply be “True” (i.e. no positive literal).
$\neg P_1(\cdot) \lor \cdots \lor \neg P_n(\cdot)$ is also a Horn clause.

Definition
A definite clause is a Horn Clause with exactly one positive literal.
$\neg P_1(\cdot) \lor \cdots \lor \neg P_n(\cdot) \lor C(\cdot)$.

Restriction for FC
FC applies GMP and only works with Definite Clauses. The presented KB is a set of Definite Clauses.
function FOL-FC-Ask(KB, α) returns a substitution or false

repeat until new is empty
    new ← {}
    for each sentence $r$ in KB do
        $(p_1 \land \ldots \land p_n \Rightarrow q) \leftarrow \text{STANDARDIZE-APART}(r)$
        for each $\theta$ such that $(p_1 \land \ldots \land p_n)\theta = (p_1' \land \ldots \land p_n')\theta$
            for some $p_1', \ldots, p_n'$ in KB
                $q' \leftarrow \text{SUBST}(\theta, q)$
                if $q'$ is not a renaming of a sentence already in KB or new then do
                    add $q'$ to new
                    $\phi \leftarrow \text{UNIFY}(q', \alpha)$
                    if $\phi$ is not fail then return $\phi$
                add new to KB
        return false
Example

S1: 
\( \forall x \forall y \forall z \ Person(x) \wedge Gives(x, y, z) \wedge GoodLecturesFOL(y) \wedge Students(z) \Rightarrow GoodTeacher(x) \)

S2: \( \exists x \ Have(People, x) \wedge GoodLecturesLogic(x) \)
We replace S2 thanks to EI, we introduce a new symbol \( M1 \)
S2: \( Have(People, M1) \wedge GoodLecturesLogic(M1) \)

S3: 
\( \forall x \ GoodLecturesLogic(x) \wedge Have(People, x) \Rightarrow Gives(Yannick, x, People) \)

S4: \( Study(People, ANU) \)

S5: \( Person(Yannick) \)

S6: \( \forall x \ Study(x, ANU) \Rightarrow Students(x) \)

S7: \( \forall x \ GoodLecturesLogic(x) \Rightarrow GoodLecturesFOL(x) \)
Forward chaining: example

Example

Person(Yannick)  GoodLecturesLogic(M1)  Have(People,M1)  Study(People, ANU)  Iteration 1

S5  S2  S2  S4
Forward chaining: example

Example

Iteration 1
- Person(Yannick)
- GoodLecturesLogic(M1)
- Have(People,M1)

Iteration 2
- GoodLecturesFOL(M1)
- Gives(Yannick,M1,People)
- Students(People)
- Study(People, ANU)

S5
S2
S2
S4
S7
{ x / M1 }
S3
{ x / M1 }
S6
{ x / People}
Forward chaining: example

Example

GoodTeacher(Yannick)

Iteration 3
{ x / Yannick, y / M1, z / People }

Iteration 2
{ x / M1 }

Iteration 1
{ x / M1 }
{ x / People}

Person(Yannick)

GoodLecturesFOL(M1)

Gives(Yannick,M1,People)

Students(People)

Have(People,M1)

Study(People, ANU)

S5

S7

{ x / M1 }

S1

S3

S2

S2

S6

S4
Properties of forward chaining

Soundness
FC is sound for KB with Definite Clauses. (Use of GMP)

Completeness
FC is complete for KB with Definite Clauses. (Brute force)

Termination
FC always terminates for KB = Datalog (no functions). FC may not terminate for KB with Definite Clauses.

Decidability
Even on Definite Clauses, the problem is semidecidable.
Efficiency of FC

Efficiency
The algorithm is brute-force. We can optimise.

Idea
- no need to match a rule on iteration $k$ if a premise wasn’t added on iteration $k - 1$
- match each rule whose premise contains a newly added literal

Algorithm RETE (management of a working memory...