Part I
Inference in first-order logic

A brief history of reasoning

450B.C. Stoics propositional logic, inference (maybe)
322B.C. Aristotle “syllogisms” (inference rules), quantifiers
1565 Cardano probability theory (propositional logic + uncertainty)
1847 Boole propositional logic (again)
1879 Frege first-order logic
1922 Wittgenstein proof by truth tables
1930 Gödel complete algorithm for FOL :-)
1930 Herbrand complete algorithm for FOL (reduce to propositional)
1931 Gödel ¬∃ complete algorithm for arithmetic :-(
1960 Davis/Putnam “practical” algorithm for propositional logic
1965 Robinson “practical” algorithm for FOL — resolution

Outline

Contents

1 Reduction to propositional inference

Universal instantiation (UI)

Definition 1. Inference rule: Every instantiation of a universally quantified sentence is entailed by it.

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

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

Example 2. \( \forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x) \) yields

\( King(John) \land Greedy(John) \Rightarrow Evil(John) \quad [g = John] \)
\( King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard) \quad [g = Richard] \)
\( King(\text{Father}(John)) \land Greedy(\text{Father}(John)) \Rightarrow Evil(\text{Father}(John)) \quad [g = \text{Father}(John)] \)

Existential instantiation (EI)

Definition 3. Inference rule: For any sentence \( \alpha \), variable \( v \), and constant symbol \( k \) that does not appear elsewhere in the knowledge base:

\[ \exists v \alpha \quad \text{SUBST}(\{v/k\}, \alpha) \]

Example 4. \( \exists x \ Crown(x) \land \text{OnHead}(x, John) \) yields

\( Crown(C1) \land \text{OnHead}(C1, John) \quad [k = C1] \)

provided \( C1 \) is a new constant symbol, called a Skolem constant.
EI and UI

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

\[ KB = \{ \forall x \ P(x) \} \quad \text{new} \ Kb = \{ \forall x \ P(x), P(\text{Richard}) \} \ldots \]

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

\[ KB = \{ \exists x \ P(x) \} \quad \text{new} \ Kb = \{ P(\text{SkolemConstant}) \} \ldots \]

Reduction to propositional inference

**Example 5.** Suppose the KB contains just the following:

\[ \forall x \ King(x) \land Greedy(x) \implies Evil(x) \quad \text{King(John)} \quad \text{Greedy(John)} \quad \text{Brother(Richard, John)} \]

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

\[ \text{King(John)} \land \text{Greedy(John)} \implies \text{Evil(John)} \quad \text{King(Richard)} \land \text{Greedy(Richard)} \implies \text{Evil(Richard)} \]

The new KB is *propositionalised*: proposition symbols are King(John), Greedy(John), Evil(John), King(Richard) \ldots

Reduction contd.

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

For \( n = 0 \) to \( \text{maxDepth} \) create a \( Kb_{\text{prop}, n} \) by reduction with depth-\( m \) terms \( (m = 1 \ldots n) \)

if \( \alpha_{\text{prop}} \) is entailed by \( Kb_{\text{prop}, n} \) (i.e. \( Kb_{\text{prop}, n} \models \alpha_{\text{prop}} \)) then STOP.

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 (\( \text{maxDepth} = \infty \)):

\[ \text{Father(Father(Father(.....Father(John).....)))} \]

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

Problems with propositionalization

Propositionalization seems to generate lots of irrelevant sentences.

**Example 6.** 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 \( \text{Evil(John)} \), but propositionalization produces lots of facts such as \( \text{Greedy(Richard)} \) 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 worse!

2 Unification and lifting

Inference rules for FOL

We can get the inference immediately on the FOL KB.

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

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

2
1. **Substitution (SUBST)**

2. **Unification (UNIFY)**

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

**Unification**

**Definition 7.** 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 8.** 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:

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

<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>

**Unification**

**Example 9.**

\[
\text{UNIFY}(p, q) = \{x/Jane\}
\]

<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>

**Unification**

**Definition 10.** 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:

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

<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 12.** 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 14. 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 15.

\[
\begin{array}{c|c|c}
  p & q & \theta \\
  \text{Knows(John, x)} & \text{Knows(John, Jane)} & \{x/Jane\} \\
  \text{Knows(John, x)} & \text{Knows(y, Bill)} & \{x/Bill, y/John\} \\
  \text{Knows(John, x)} & \text{Knows(y, Mother(y))} & \{y/John, x/Mother(John)\} \\
  \text{Knows(John, x)} & \text{Knows(x, Elizabeth)} & \text{fail} \\
\end{array}
\]

Unification

Definition 16. 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 17.

\[
\begin{array}{c|c|c}
  p & q & \theta \\
  \text{Knows(John, x)} & \text{Knows(John, Jane)} & \{x/Jane\} \\
  \text{Knows(John, x)} & \text{Knows(y, Bill)} & \{x/Bill, y/John\} \\
  \text{Knows(John, x)} & \text{Knows(y, Mother(y))} & \{y/John, x/Mother(John)\} \\
  \text{Knows(John, x)} & \text{Knows(x, Elizabeth)} & \text{fail} \\
\end{array}
\]

Unification

Definition 18. 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 19.

\[
\begin{array}{c|c|c}
  p & q & \theta \\
  \text{Knows(John, x)} & \text{Knows(John, Jane)} & \{x/Jane\} \\
  \text{Knows(John, x)} & \text{Knows(y, Bill)} & \{x/Bill, y/John\} \\
  \text{Knows(John, x)} & \text{Knows(y, Mother(y))} & \{y/John, x/Mother(John)\} \\
  \text{Knows(John, x)} & \text{Knows(x, Elizabeth)} & \text{fail} \\
\end{array}
\]

Standardizing apart eliminates overlap of variables (renaming of variables) $\text{Knows}(z_{17}, Elizabeth)$
Unification

**Definition 20.** 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:

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

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( \theta )</th>
<th>Example 21.</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{Knows}(John, x)</td>
<td>\text{Knows}(John, Jane)</td>
<td>{x/Jane}</td>
<td>\text{Knows}(John, x) \text{Knows}(y, Bill) {x/Bill, y/John}</td>
</tr>
<tr>
<td>\text{Knows}(John, x)</td>
<td>\text{Knows}(y, Mother(y))</td>
<td>{y/John, x/Mother(John)}</td>
<td>fail</td>
</tr>
<tr>
<td>\text{Knows}(John, x)</td>
<td>\text{Knows}(x, Elizabeth)</td>
<td>fail</td>
<td></td>
</tr>
</tbody>
</table>

*Standardizing apart* eliminates overlap of variables (renaming of variables) \( \text{Knows}(z_{17}, Elizabeth) \text{Knows}(John, x) \text{Knows}(z_{17}, Elizabeth) \)

\( \theta = \text{UNIFY}(\text{Knows}(John, x), \text{Knows}(z_{17}, Elizabeth)) = \text{UNIFY}(\text{Knows}(y, Bill), \text{Knows}(y, Mother(y))) = \{x/Bill, y/John\} \)

Generalized Modus Ponens (GMP)

**Definition 22.**

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

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

Variables are universally quantified.

Example 23.

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

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

Generalized Modus Ponens (GMP)

**Definition 24.**

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

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

Variables are universally quantified.

Example 25.

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

What is \( \theta \)? What is \( \text{SUBST}(\theta, \text{Evil}(x)) \)? \( \theta = \text{UNIFY}(\text{King}(John), \text{King}(x)) = \text{UNIFY}(\text{Greedy}(y), \text{Greedy}(x)) = \{x/John, y/John\} \)

Generalized Modus Ponens (GMP)

**Definition 26.**

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

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

Variables are universally quantified.
Example 27. 
\[
\begin{align*}
\text{King}(John), \text{Greedy}(y) & \quad (\text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)) \\
\text{SUBST}(\theta, \text{Evil}(x)) \quad \text{What is } \theta? \quad \text{What is } \text{SUBST}(\theta, \text{Evil}(x))? \quad \theta = \text{UNIFY}(\text{King}(John), \text{King}(x)) = \text{UNIFY}(\text{Greedy}(y), \text{Greedy}(x)) \quad \theta = \{x/John, y/John\} \quad \text{SUBST}(\theta, \text{Evil}(x)) = \text{Evil}(John)
\end{align*}
\]

GMP is sound

GMP is sound. We have:
\[
p_1', p_2', \ldots, p_n' \quad (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \models \text{SUBST}(\theta, q)
\]
provided \(\theta = \text{UNIFY}(p_i, p'_i)\) for all \(i\).

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
\[
\text{SUBST}(\theta, p_1') \land \cdots \land \text{SUBST}(\theta, p_n')
\]
3. because of 1 and UI, from \(p_1 \land p_2 \land \ldots \land p_n \Rightarrow q\) we infer
\[
\text{SUBST}(\theta, p_1) \land \cdots \land \text{SUBST}(\theta, p_n) \Rightarrow \text{SUBST}(\theta, q)
\]
4. If \(\theta = \text{UNIFY}(p_i, p'_i)\) for all \(i\) then from 2 and 3 we infer \(\text{SUBST}(\theta, q)\) \(\square\)

3 Knowledge base: an example

Knowledge base: an example

Example 28. “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 29. “...a person who gives good lectures about FOL to students is a good teacher...”:

Knowledge base: an example

Example 30. “...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 31. “...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”:

Knowledge base: an example

Example 32. “...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”:
\[
\exists x \text{Have}(\text{People}, x) \land \text{GoodLecturesLogic}(x)
\]

6
Knowledge base: an example

Example 33. "...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...":

Knowledge base: an example

Example 34. "...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...": \( \forall x \text{GoodLecturesLogic}(x) \land \text{Have}(\text{People}, x) \Rightarrow \text{Gives}(\text{Yannick}, x, \text{People}) \)

Knowledge base: an example

Example 35. "...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":

Knowledge base: an example

Example 36. "...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": \( \text{Study}(\text{People}, \text{ANU}) \)

Knowledge base: an example

Example 37. "...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...": \( \forall x \text{GoodLecturesLogic}(x) \land \text{Have}(\text{People}, x) \Rightarrow \text{Gives}(\text{Yannick}, x, \text{People}) \)

"...Yannick who is a person...":

Knowledge base: an example

Example 38. "...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... ∀x GoodLecturesLogic(x) ∧ Have(People, x) ⇒ Gives(Yannick, x, People)

This group of people... have very good lectures about Logic: ∃x Have(People, x) ∧ GoodLecturesLogic(x)

Knowledge base: an example

Example 39. "...a person who gives good lectures about FOL to students is a good teacher...": ∀x∀y∀z Person(x) ∧ Gives(x, y, z) ∧ GoodLecturesFOL(y) ∧ Students(z) ⇒ GoodTeacher(x)

This group of people... have very good lectures about Logic: ∃x Have(People, x) ∧ GoodLecturesLogic(x)

Knowledge base: an example

Example 40. "...a person who gives good lectures about FOL to students is a good teacher...": ∀x∀y∀z Person(x) ∧ Gives(x, y, z) ∧ GoodLecturesFOL(y) ∧ Students(z) ⇒ GoodTeacher(x)

This group of people... have very good lectures about Logic: ∃x Have(People, x) ∧ GoodLecturesLogic(x)

Knowledge base: an example

Example 41. "...a person who gives good lectures about FOL to students is a good teacher...": ∀x∀y∀z Person(x) ∧ Gives(x, y, z) ∧ GoodLecturesFOL(y) ∧ Students(z) ⇒ GoodTeacher(x)

This group of people... have very good lectures about Logic: ∃x Have(People, x) ∧ GoodLecturesLogic(x)

Knowledge base: an example

Example 42. "...a person who gives good lectures about FOL to students is a good teacher...": ∀x∀y∀z Person(x) ∧ Gives(x, y, z) ∧ GoodLecturesFOL(y) ∧ Students(z) ⇒ GoodTeacher(x)

This group of people... have very good lectures about Logic: ∃x Have(People, x) ∧ GoodLecturesLogic(x)

Knowledge base: an example

Example 43. "...a person who gives good lectures about FOL to students is a good teacher...": ∀x∀y∀z Person(x) ∧ Gives(x, y, z) ∧ GoodLecturesFOL(y) ∧ Students(z) ⇒ GoodTeacher(x)

This group of people... have very good lectures about Logic: ∃x Have(People, x) ∧ GoodLecturesLogic(x)
4 Forward chaining

Restriction: Definite clauses

Definition 43. As said previously, a Horn clause is
- a predicate $P(...)$, or
- something like $P_1(...) \cdots P_n(\ldots) \Rightarrow C(...)$
- $\equiv \neg P_1(...) \lor \cdots \lor \neg P_n(...) \lor C(...)$

($P_i(\ldots)$ is a premise and $C(\ldots)$ the conclusion)

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

Definition 44. A definite clause is a Horn Clause with exactly one positive literal. $\neg P_1(...) \lor \cdots \lor \neg P_n(...) \lor C(...)$. FC applies GMP and only works with Definite Clauses. The presented KB is a set of Definite Clauses.

Forward chaining: algorithm

```
function FOL-FC-Ask(KB, a) returns a substitution or false
    repeat until new is empty
        new ← \{
        for each sentence $\phi$ in KB do
            \{ $p_1 \land \ldots \land p_n \Rightarrow q$ \} ← STANDARDIZE-APART($\phi$)
            for each $\theta$ such that $(p_1 \land \ldots \land p_n)$
                $\equiv (p'_1 \land \ldots \land p'_n)$
                for some $p'_1, \ldots, p'_n$ in KB
                    $q' ← \text{SUBST}(\theta, q)$
                    if $q'$ is not a renaming of a sentence already in KB or new then do
                        add $q'$ to new
                        $\phi ← \text{UNIFY}(q', a)$
                        if $\phi$ is not fail then return $\phi$
        return false
```

Example

Example 45. S1: $\forall x\forall y\forall z \ \text{Person}(x) \land \text{Gives}(x, y, z) \land \text{GoodLecturesFOL}(y) \land \text{Students}(z) \Rightarrow \text{GoodTeacher}(x)$ \quad S2: $\exists x \ \text{Have}(\text{People}, x) \land \text{GoodLecturesLogic}(x)$ \quad We replace S2 thanks to EI, we introduce a new symbol $M1$ \quad S2: $\text{Have}(\text{People}, M1) \land \text{GoodLecturesLogic}(M1)$

S3: $\forall x \text{GoodLecturesLogic}(x) \land \text{Have}(\text{People}, x) \Rightarrow \text{Gives}(Yannick, x, \text{People})$
S4: $\text{Study}(\text{People}, \text{ANU})$
S5: $\text{Person}(Yannick)$
S6: $\forall x \ \text{Study}(x, \text{ANU}) \Rightarrow \text{Students}(x)$
S7: $\forall x \text{GoodLecturesLogic}(x) \Rightarrow \text{GoodLecturesFOL}(x)$

Forward chaining: example

Example 46. 

<table>
<thead>
<tr>
<th>Person(Yannick)</th>
<th>GoodLecturesLogic(M1)</th>
<th>Have(People, M1)</th>
<th>Study(People, ANU)</th>
<th>Iteration 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>S5</td>
<td>S2</td>
<td>S2</td>
<td>S4</td>
<td></td>
</tr>
</tbody>
</table>
Forward chaining: example

Example 47.

Forward chaining: example

Example 48.

Properties of forward chaining

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

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

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

Even on Definite Clauses, the problem is semidecidable.

Efficiency of FC

The algorithm is brute-force. We can optimise.

- 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...)