# KRR3: Inference in First-order logic

Yannick Pencolé
Yannick.Pencole@anu.edu.au

14 Mar 2005

# A brief history of reasoning

**History**

| | | |
|---|---|---|
| 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

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.

$$\frac{\forall v\ \alpha}{\textsc{Subst}(\{v/g\}, \alpha)}$$

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

## Example

$\forall x\ King(x) \land Greedy(x) \Rightarrow Evil(x)$ yields

$King(John) \land Greedy(John) \Rightarrow Evil(John)$ $[g = John]$
$King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)$ $[g = Richard]$
$King(Father(John)) \land Greedy(Father(John)) \Rightarrow Evil(Father(John))$
$[g = Father(John)]$

# 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:

$$\frac{\exists v \ \alpha}{\textsc{Subst}(\{v/k\}, \alpha)}$$

## Example

$\exists x \ Crown(x) \wedge OnHead(x, John)$ yields
$Crown(C_1) \wedge OnHead(C_1, John) \quad [k = C_1]$

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.

$KB = \{\forall x\ P(x)\}$

$newKB = \{\forall x\ P(x), P(Richard)\}$...

## EI

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

$KB = \{\exists x\ P(x)\}$

$newKB = \{P(SkolemConstant)\}$...

# Reduction to propositional inference

## Example

Suppose the KB contains just the following:

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
$King(John)$
$Greedy(John)$
$Brother(Richard, John)$

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

$King(John) \wedge Greedy(John) \implies Evil(John)$
$King(Richard) \wedge Greedy(Richard) \implies Evil(Richard)$
$King(John)$
$Greedy(John)$
$Brother(Richard, John)$

## Propositionalisation

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

# Reduction contd.

## 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 ($maxDepth = \infty$): $Father(Father(Father(.....Father(John)....)))$

## Can we solve the problem?

NO. Entailement 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 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:

$$\textsc{Unify}(p, q) = \theta$$

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

$$\textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, q)$$

# Unification

## Definition

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

$$\textsc{Unify}(p, q) = \theta$$

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

$$\textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, q)$$

## Example

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | |
| $Knows(John, x)$ | $Knows(y, Bill)$ | |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | |
| $Knows(John, x)$ | $Knows(x, Elizabeth)$ | |

# 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

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, Bill)$ | |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | |
| $Knows(John, x)$ | $Knows(x, Elizabeth)$ | |

# Unification

## Definition

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

$$\textsc{Unify}(p, q) = \theta$$

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

$$\textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, q)$$

## Example

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, Bill)$ | $\{x/Bill, y/John\}$ |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | |
| $Knows(John, x)$ | $Knows(x, Elizabeth)$ | |

# Unification

## Definition

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

$$\textsc{Unify}(p, q) = \theta$$

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

$$\textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, q)$$

## Example

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, Bill)$ | $\{x/Bill, y/John\}$ |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | $\{y/John, x/Mother(John)\}$ |
| $Knows(John, x)$ | $Knows(x, Elizabeth)$ | |

# Unification

## Definition

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

$$\textsc{Unify}(p, q) = \theta$$

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

$$\textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, q)$$

## Example

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, Bill)$ | $\{x/Bill, y/John\}$ |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | $\{y/John, x/Mother(John)\}$ |
| $Knows(John, x)$ | $Knows(x, Elizabeth)$ | *fail* |

# 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

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, Bill)$ | $\{x/Bill, y/John\}$ |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | $\{y/John, x/Mother(John)\}$ |
| $Knows(John, x)$ | $Knows(x, Elizabeth)$ | *fail* |

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

| $p$ | $q$ | $\theta$ |
|---|---|---|
| $Knows(John, x)$ | $Knows(John, Jane)$ | $\{x/Jane\}$ |
| $Knows(John, x)$ | $Knows(y, Bill)$ | $\{x/Bill, y/John\}$ |
| $Knows(John, x)$ | $Knows(y, Mother(y))$ | $\{y/John, x/Mother(John)\}$ |
| $Knows(John, x)$ | $Knows(x, Elizabeth)$ | $fail$ |

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

| | | |
|---|---|---|
| $Knows(John, x)$ | $Knows(z_{17}, Elizabeth)$ | $\{x/Elizabeth, z_{17}/John\}$ |

# Generalized Modus Ponens (GMP)

**Definition**

$$\frac{p_1{}',\quad p_2{}',\ \ldots,\ p_n{}',\quad (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q)}{\textsc{Subst}(\theta, q)}$$

where $\theta = \textsc{Unify}(p_i, p_i')$ for all $i$.
Variables are universally quantified.

**Example**

$$\frac{King(John),\quad Greedy(y)\quad (King(x) \land Greedy(x) \Rightarrow Evil(x))}{\textsc{Subst}(\theta, Evil(x))}$$

What is $\theta$? What is $\textsc{Subst}(\theta, Evil(x))$?

# Generalized Modus Ponens (GMP)

## Definition

$$\frac{p_1{}',\ \ p_2{}',\ \ldots,\ p_n{}',\ \ (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q)}{\textsc{Subst}(\theta, q)}$$

where $\theta = \textsc{Unify}(p_i, p_i')$ for all $i$.
Variables are universally quantified.

## Example

$$\frac{King(John),\ \ Greedy(y)\ \ (King(x) \land Greedy(x) \Rightarrow Evil(x))}{\textsc{Subst}(\theta, Evil(x))}$$

What is $\theta$? What is $\textsc{Subst}(\theta, Evil(x))$?
$\theta = \textsc{Unify}(King(John), King(x)) = \textsc{Unify}(Greedy(y), Greedy(x))$
$\theta = \{x/John, y/John\}$

# Generalized Modus Ponens (GMP)

## Definition

$$\frac{p_1{}',\ \ p_2{}',\ \ldots,\ p_n{}',\ \ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{\text{SUBST}(\theta, q)}$$

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

## Example

$$\frac{King(John),\ \ Greedy(y)\ \ (King(x) \wedge Greedy(x) \Rightarrow Evil(x))}{\text{SUBST}(\theta, Evil(x))}$$

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

# GMP is sound

## Property

GMP is sound. We have:

$$p_1{}',\ \ p_2{}',\ \ldots,\ p_n{}'\ \ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q) \models \textsc{Subst}(\theta, q)$$

provided $\theta = \textsc{Unify}(p_i, p_i')$ for all $i$.

## Proof idea

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

$$p \models \textsc{Subst}(\theta, p)$$

2. because of 1 and UI, from $p_1{}',\ \ p_2{}',\ \ldots,\ p_n{}'$ we infer

$$\textsc{Subst}(\theta, p_1') \wedge \cdots \wedge \textsc{Subst}(\theta, p_n{}')$$

3. because of 1 and UI, from $p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q$ we infer

$$\textsc{Subst}(\theta, p_1) \wedge \cdots \wedge \textsc{Subst}(\theta, p_n) \Rightarrow \textsc{Subst}(\theta, q)$$

4. If $\theta = \textsc{Unify}(p_i, p_i')$ for all $i$ then from 2 and 3 we infer $\textsc{Subst}(\theta, q)$ $\square$

# Outline

# Knowledge base: an example

**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...”:

# Knowledge base: an example

## Example

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

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

# 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...":

$$\forall x \forall y \forall z \; Person(x) \wedge Gives(x, y, z) \wedge GoodLecturesFOL(y) \wedge Students(z) \Rightarrow GoodTeacher(x)$$

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

$$\exists x \; Have(People, x) \wedge GoodLecturesLogic(x)$$

# Knowledge base: an example

**Example**

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

$\forall x \forall y \forall z \; Person(x) \wedge Gives(x, y, z) \wedge GoodLecturesFOL(y) \wedge Students(z) \Rightarrow GoodTeacher(x)$

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

$\exists x \; Have(People, x) \wedge GoodLecturesLogic(x)$

"...all of their lectures are given by Yannick...":

# Knowledge base: an example

## Example

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

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

"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...":

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

# Knowledge base: an example

**Example**

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

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

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

$\exists x\ Have(People, x) \wedge GoodLecturesLogic(x)$

"...all of their lectures are given by Yannick...":

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

"..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...":

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

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

$$\exists x\ Have(People, x) \wedge GoodLecturesLogic(x)$$

"...all of their lectures are given by Yannick...":

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

"..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...":

$\forall x \forall y \forall z \; Person(x) \wedge Gives(x, y, z) \wedge GoodLecturesFOL(y) \wedge Students(z) \Rightarrow GoodTeacher(x)$

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

$\exists x \; Have(People, x) \wedge GoodLecturesLogic(x)$

"...all of their lectures are given by Yannick...":

$\forall x \; GoodLecturesLogic(x) \wedge Have(People, x) \Rightarrow Gives(Yannick, x, People)$

"..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...":

$\forall x \forall y \forall z \; Person(x) \wedge Gives(x, y, z) \wedge GoodLecturesFOL(y) \wedge Students(z) \Rightarrow GoodTeacher(x)$

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

$\exists x \; Have(People, x) \wedge GoodLecturesLogic(x)$

"...all of their lectures are given by Yannick...":

$\forall x \; GoodLecturesLogic(x) \wedge Have(People, x) \Rightarrow Gives(Yannick, x, People)$

"..This group of people, studying at the ANU":

$Study(People, ANU)$

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

$Person(Yannick)$

# Knowledge base: an example

## Example

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

$$\forall x \forall y \forall z \; Person(x) \wedge Gives(x, y, z) \wedge GoodLecturesFOL(y) \wedge Students(z) \Rightarrow GoodTeacher(x)$$

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

$$\exists x \; Have(People, x) \wedge GoodLecturesLogic(x)$$

"...all of their lectures are given by Yannick...":

$$\forall x \; GoodLecturesLogic(x) \wedge Have(People, x) \Rightarrow Gives(Yannick, x, People)$$

"..This group of people, studying at the ANU":

$$Study(People, ANU)$$

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

$$Person(Yannick)$$

People that study at the ANU, are students

# Knowledge base: an example

## Example

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

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

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

$\exists x\ Have(People, x) \wedge GoodLecturesLogic(x)$

"...all of their lectures are given by Yannick...":

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

"..This group of people, studying at the ANU":

$Study(People, ANU)$

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

$Person(Yannick)$

People that study at the ANU, are students

$\forall x\ Study(x, ANU) \Rightarrow Students(x)$

# Knowledge base: an example

## Example

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

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

"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..":

$$Person(Yannick)$$

People that study at the ANU, are students

$$\forall x\ Study(x, ANU) \Rightarrow Students(x)$$

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...":

$\forall x \forall y \forall z \; Person(x) \wedge Gives(x, y, z) \wedge GoodLecturesFOL(y) \wedge Students(z) \Rightarrow GoodTeacher(x)$

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

$\exists x \; Have(People, x) \wedge GoodLecturesLogic(x)$

"...all of their lectures are given by Yannick...":

$\forall x \; GoodLecturesLogic(x) \wedge Have(People, x) \Rightarrow Gives(Yannick, x, People)$

"..This group of people, studying at the ANU":

$Study(People, ANU)$

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

$Person(Yannick)$

People that study at the ANU, are students

$\forall x \; Study(x, ANU) \Rightarrow Students(x)$

If the lectures about Logic are good, the lectures about FOL are good:

$\forall x \; GoodLecturesLogic(x) \Rightarrow GoodLecturesFOL(x)$

# Outline

# Restriction: Definite clauses

## Definition

As said previously, a Horn clause is

- a predicate $P(..)$, or
- something like $P_1(..) \cdots P_n(..) \Rightarrow C(..)$
- $\equiv \neg P_1(..) \vee \cdots \vee \neg P_n(..) \vee C(..)$

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

## Definition

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

## Restriction for FC

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($KB, \alpha$) **returns** a substitution or *false*

 **repeat until** *new* is empty
  *new* $\leftarrow \{\}$
  **for each** sentence $r$ **in** $KB$ **do**
   $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-APART($r$)
   **for each** $\theta$ such that $(p_1 \wedge \ldots \wedge p_n)\theta = (p'_1 \wedge \ldots \wedge p'_n)\theta$
      for some $p'_1, \ldots, p'_n$ in $KB$
    $q' \leftarrow$ SUBST($\theta, q$)
    **if** $q'$ is not a renaming of a sentence already in $KB$ or *new* **then do**
     add $q'$ to *new*
     $\phi \leftarrow$ UNIFY($q', \alpha$)
     **if** $\phi$ is not *fail* **then return** $\phi$
  add *new* to $KB$
 **return** *false*

# Example

## 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: $\forall x\ GoodLecturesLogic(x) \Rightarrow GoodLecturesFOL(x)$

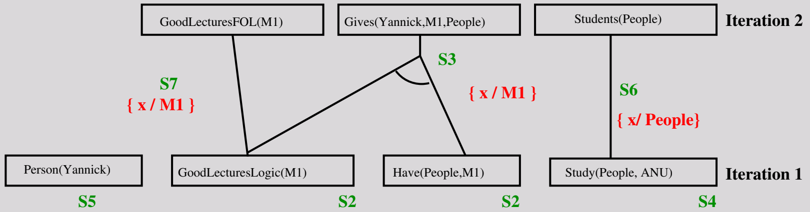
# Forward chaining: example

## Example

| Person(Yannick) | GoodLecturesLogic(M1) | Have(People,M1) | Study(People, ANU) | **Iteration 1** |
| --- | --- | --- | --- | --- |
| **S5** | **S2** | **S2** | **S4** | |

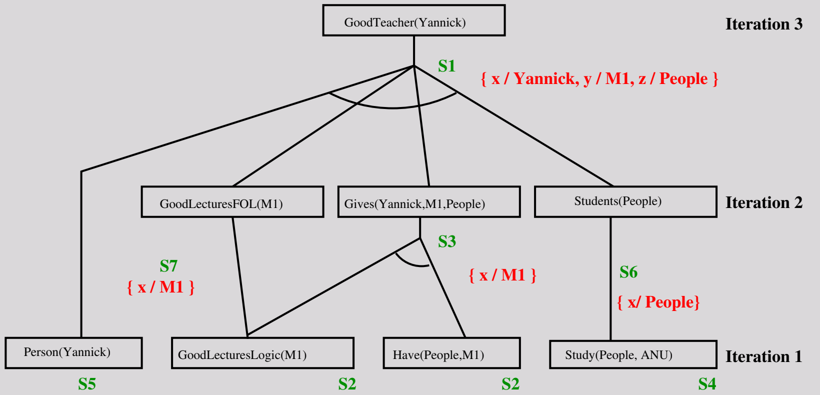
# Forward chaining: example

## Example



# Forward chaining: example

## Example



# 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 (managment of a working memory...)