Logic and Knowledge Representation

Predicate Logic
18 May 2018

Giovanni Sileno gsileno@enst.fr
Télécom ParisTech, Paris-Dauphine University
Predicate logic
All men are mortals.
Socrates is a man.
Therefore... ?
All men are mortals.  TRUE
Socrates is a man.  TRUE
Therefore... ?
All men are mortals. → TRUE
Socrates is a man. → TRUE
Therefore... ?

We cannot do a lot just working at the level of propositions.
All men are mortals. Socrates is a man. Therefore... ?

We cannot do a lot just working at the level of propositions.

→ We need to have access to individuals, their properties and their relations.
All men are mortals.  
Socrates is a man.  
Therefore... ?

We cannot do a lot just working at the level of propositions.

→ We need to have access to individuals, their properties and their relations.
Additional problems...

• 4 is preceded by 2 $\iff$ 2 precedes 4 $\rightarrow$ TRUE
Additional problems...

• 4 is preceded by 2 ⇔ 2 precedes 4 → TRUE

All instants are preceded by an instant
⇔ An instant precedes all instants → FALSE
Additional problems...

• 4 is preceded by 2 ⇔ 2 precedes 4 → TRUE

All instants are preceded by an instant
⇔ An instant precedes all instants → FALSE

• **No man** is immortal ???
**Predicate logic: Quantifiers**

- $\exists x [...]$

  The *existential quantifier* means: amongst all entities in the universe, there is *at least one entity* which satisfies what described in $[...]$
Predicate logic: Quantifiers

- $\exists x [...]

  The **existential quantifier** means: amongst all entities in the universe, there is *at least one entity* which satisfies what described in [...]

- $\forall x [...]

  The **universal quantifier** means: *all entities* in the universe satisfy what described in [...]

Gottlob Frege
Predicate logic: Quantifiers

- $\exists x \ldots$
  
  The **existential quantifier** means: amongst all entities in the universe, there is *at least one entity* which satisfies what described in $\ldots$

- $\forall x \ldots$
  
  The **universal quantifier** means: *all entities* in the universe satisfy what described in $\ldots$

Nota bene: the $x$ is a specific individual within $\ldots$, and it cannot be outside of $\ldots$
Predicate logic

- $A(x)$ is a predicate with argument $x$
  from *praedico* (~ to say something about $x$ publicly)
Predicate logic

- $A(x)$ is a predicate with argument $x$ from *praedico* (~ to say something about $x$ publicly)

- It is a “open” statement: there is no anchoring to actual elements. ex. $\text{Red(car)}$: “car is red” does not make complete sense: which car?
Predicate logic

- $A(x)$ is a **predicate** with argument $x$
- when $t$ has a defined **value** (i.e. it refers to a specific entity), $A(t)$ is a **proposition** and may be true or false.

- Example: That car is red: $Red(\text{thatcar})$
  (assuming $\text{thatcar}$ to be a shared constant)
Predicate logic

• $A(x)$ is a predicate with argument $x$

• when $t$ has a defined value (i.e. it refers to a specific entity), $A(t)$ is a proposition and may be true or false.

We can create a map with all the possible values of $x$, where $A(x)$ is true, and where is false, i.e. where $\neg A(x)$ is true
Predicate logic and Venn diagrams

• Considering just one predicate \( A(x) \), we have:

- Domain or universe of the predicate variable \( x \)
- \( A \): set of all \( x \) such as \( A(x) \) is true
- \( A^c \): set of all \( x \) such as \( A(x) \) is not true
Predicate logic: new features

- **Individual variables**: $x, y, z, \ldots$ may be considered to vary over one (or more) universes.

  Attention: letters in the propositional calculus denoted fixed statements, like "it rains", "Giovanni is teaching LKR", etc.
Predicate logic: new features

• **Individual variables**: $x, y, z, \ldots$ may be considered to vary over one (or more) universes.

  Attention: letters in the propositional calculus denoted fixed statements, like “it rains”, “Giovanni is teaching LKR”, etc.

• **Predicate symbols**: $B(\cdot), P(\cdot, \cdot)$, etc. denote fixed relations “$\cdot$ is a bird”, “$\cdot$ is parent of $\cdot$”. 
Predicate logic: new features

- **Individual variables**: $x, y, z, \ldots$ may be considered to vary over one (or more) universes.
  
  Attention: letters in the propositional calculus denoted fixed statements, like “it rains”, “Giovanni is teaching LKR”, etc.

- **Predicate symbols**: $B(\cdot), P(\cdot, \cdot)$, etc. denote fixed relations “$\cdot$ is a bird”, “$\cdot$ is parent of $\cdot$”.

- **Quantifiers**: “For all $x$” ($\forall x$) and “there exists an $x$” ($\exists x$) *bind* the individual variable $x$. 
Predicate logic: new features

• **Individual variables**: x, y, z, . . . may be considered to vary over one (or more) universes.
  
  Attention: letters in the propositional calculus denoted fixed statements, like “it rains”, “Giovanni is teaching LKR”, etc.

• **Predicate symbols**: B(·), P(·, ·), etc. denote fixed relations “· is a bird”, “· is parent of ·”.

• **Quantifiers**: “For all x” (∀x) and “there exists an x” (∃x) *bind* the individual variable x.

**First-order logic (FOL):**

no quantifiers over predicates, e.g. ∀P [..]
no predicate applies on predicates. e.g. P(Q, R)
Predicate logic and Venn diagrams

- When a statement contains more predicates, we can map the different sets and translate the logical operators in set operators. The resulting set is where our original statement is true.

*domain* or *universe* of the predicate variable $x$
Predicate logic and syllogisms

A: All students wear uniforms.
E: No student wears uniforms.
I: Some students wear uniforms.
O: Not all students wear uniforms.
Predicate logic and syllogisms

A: All students wear uniforms.
E: No student wears uniforms.
I: Some students wear uniforms.
O: Not all students wear uniforms.

implicit universe:
∀x WearsUniform(x)

explicit universe:
∀x [Student(x) → WearsUniform(x)]
Predicate logic and syllogisms

A: All students wear uniforms.
E: No student wears uniforms.
I: Some students wear uniforms.
O: Not all students wear uniforms.

implicit universe:
∀x WearsUniform(x)

but the universe may be empty!

explicit universe:
∀x [Student(x) → WearsUniform(x)]
Predicate logic and syllogisms

A: All students wear uniforms.
E: No student wears uniforms.
I: Some students wear uniforms.
O: Not all students wear uniforms.

Syllogisms assume that in A there is always some student:

... \land \exists x \text{Student}(x)

implicit universe:
\forall x \text{WearsUniform}(x)

explicit universe:
\forall x [\text{Student}(x) \rightarrow \text{WearsUniform}(x)]
Example

any two individuals communicate, if there is an interpreter

$$(\forall x) \ (\forall y) \ (\forall l_1) \ (\forall l_2) \ (s(x, l_1) \land s(y, l_2) \land (\exists z) \ (s(z, l_1) \land s(z, l_2))) \supset c(x, y)$$
Example

any two individuals communicate, if there is an interpreter

\((\forall x) (\forall y) (\forall l_1) (\forall l_2) ((s(x, l_1) \land (s(y, l_2) \land (\exists z) (s(z, l_1) \land (s(z, l_2)))) \supset c(x, y))\)

\(\neg\)

\((\exists x) (\exists y) (\exists l_1) (\exists l_2) ((s(x, l_1) \land (s(y, l_2) \land (\exists z) (s(z, l_1) \land (s(s, l_2)))) \land \neg c(x, y))\)

there are two individuals who do not communicate despite the presence of an interpreter
A language consists of symbols, ...

• Alphabet
  – terms:
    • constants: c1, c2, ...
    • variables: v1, v2, ...
    • functors: f(t1, .., tn), where t1, .. tn are terms
A language consists of **symbols**, ...

- **Alphabet**
  - **terms:**
    - **constants:** $c_1, c_2, ...$
    - **variables:** $v_1, v_2, ...$
    - **functors:** $f(t_1, .., t_n)$, where $t_1, .., t_n$ are terms
  - **predicates:** $p(t_1, .., t_n)$, where $t_1, .., t_n$ are terms
  - **quantifiers:** $\forall, \exists$
  - **propositional logic connectives:** $T, \bot, \neg, \land, \lor, \supset, \equiv, ...$
A language consists of a **syntax** (rules to aggregate symbols), ...

- set $A$ of atomic formulas
  - predicates: $p(t_1, \ldots, t_n)$, where $t_1, \ldots, t_n$ are terms
  - nullary connectives $\top, \bot$
A language consists of a syntax (rules to aggregate symbols), ...

- set A of atomic formulas
  - predicates: \( p(t_1, .., t_n) \), where \( t_1, .. t_n \) are terms
  - nullary connectives \( \top, \bot \)

- A formula is
  - an atomic formula
  - if F is a formula, then \( \neg F \) is a formula
  - if F and G are formulas, then \( (F \circ G) \) is a formula, where \( \circ \) is a binary connective.
  - if F is a formula, \( (\forall x) F \) and \( (\exists x) F \) are formulas, where x is a variable.
A language consists of a **syntax** (rules to aggregate symbols), ...

- **set A of atomic formulas**
  - predicates: $p(t_1, .., t_n)$, where $t_1, .. t_n$ are terms
  - nullary connectives $\top, \bot$

- **A formula is**
  - an atomic formula
  - if $F$ is a formula, then $\neg F$ is a formula
  - if $F$ and $G$ are formulas, then $(F \circ G)$ is a formula, where $\circ$ is a binary connective.
  - if $F$ is a formula, $(\forall x) F$ and $(\exists x) F$ are formulas, where $x$ is a variable.

A formula is **ground** if there are no variables.
A language consists of a **syntax** (rules to aggregate symbols), ...

- A term or a formula is **ground** if there are no variables.
A language consists of a syntax (rules to aggregate symbols), ...

- A term or a formula is ground if there are no variables.
- A variable is either free or bound in a formula.
  - x is bound by the smallest $\forall x \ F$ or $\exists x \ F$ in the formula where the $F$ contains an occurrence of $x$;
  - unbound variables are free.
A language consists of a syntax (rules to aggregate symbols), ...

- A term or a formula is **ground** if there are no variables.
- A variable is either **free** or **bound** in a formula.
  - x is **bound** by the smallest $\forall x \, F$ or $\exists x \, F$ in the formula where the $F$ contains an occurrence of x;
  - unbound variables are **free**.
- A **closed formula** or **statement** is a formula with no free variables.
Another example: Peano arithmetics

- Alphabet
  - terms:
    - constants: 1, 2, 3, 4, 5, 6, 7, 8, 9
    - variables: x, y, z, ...
    - functors (arity): 0/0, +/2, */2
  - predicates (arity): </2, ≤/2, ≈/2, ...
  - quantifiers: ∀, ∃
  - propositional logic connectives: ⊤, ⊥, ¬, ∧, ∨, ⊃, ≡, ...

- Examples:
  \[(∀x) (∀y) (x ≤ y ≡ (∃z) (x+z≈y))\]
  \[(∃x) (∀y) (x+y≈y)\]
Another example: Peano arithmetics

• Alphabet
  – terms:
    • constants: 1, 2, 3, 4, 5, 6, 7, 8, 9
    • variables: x, y, z, ...
    • functors (arity): 0/0, +/2, */2
  – predicates (arity): </2, ≤/2, ≈/2, ...
  – quantifiers: ∀, ∃
  – propositional logic connectives: ⊤, ⊥, ¬, ∧, ∨, ⊃, ≡, ...

• Examples:
  \[(\forall x) (\forall y) (x \leq y \equiv (\exists z) (x+z\approx y)) \quad \text{Definition of} \leq\]
  \[(\exists x) (\forall y) (x+y\approx y) \quad \text{Definition of} \ 0\]
Another example: Peano arithmetics

- **Alphabet**
  - **terms:**
    - **constants:** 1, 2, 3, 4, 5, 6, 7, 8, 9
    - **variables:** x, y, z, ...
    - **functors (arity):** 0/0, +/2, */2
  - **predicates (arity):** </2, ≤/2, ≈/2, ...
  - **quantifiers:** ∀, ∃
  - **propositional logic connectives:** ⊤, ⊥, ¬, ∧, ∨, ⊃, ≡, ...

- **Examples:**
  - $(∀x) (∀y) (x ≤ y ≡ (∃z) (x+z ≈ y))$  \[ \text{Definition of } ≤ \]
  - $(∃x) (∀y) (x+y ≈ y)$  \[ \text{Definition of } 0 \]

*trade-off between complexity of signature and of quantification*
A language consists of a **semantic** (rules to interpret it)

- A non-empty set D called **domain**.
A language consists of a **semantic** (rules to interpret it)

- A non-empty set $D$ called **domain**.
- An **interpretation** $I$ associates:
  - each constant $c$ of the language with an element $c^I$ of $D$,
  - each functor $f$ of arity $n$ to a function $f^I : D^n \to D$
  - each predicate $P$ of arity $n$ to a $n$-ary relation $P^I$ dans $D$. 


A language consists of a semantic (rules to interpret it)

- A non-empty set $D$ called **domain**.
- An **interpretation $I$** associates:
  - each constant $c$ of the language with an element $c^I$ of $D$,
  - each functor $f$ of arity $n$ to a function $f^I : D^n \rightarrow D$
  - each predicate $P$ of arity $n$ to a $n$-ary relation $P^I$ dans $D$.
- An **assignment $A$** instantiates each variable $v$ by giving it a value $v^A$ taken from $D$. 
A language consists of a semantic (rules to interpret it)

- Terms are interpreted recursively from the interpretation of their elements: for each term $t$, $t^{I,A}$ is defined as:
  - $c^I$ for a constant $c$,
  - $v^A$ for a variable $v$,
  - $f^I(t_1^{I,A}, t_2^{I,A}, \ldots, t_n^{I,A})$ for a functional term $f(t_1, \ldots, t_n)$.

- A model $M(D, I)$ is defined by the domain $D$ and the interpretation $I$. 

A language consists of a **semantic** (rules to interpret it)

- How to evaluate truth?
  - Start by considering the domain as a *database* made with many tables (one for each $P^1$), then..
A language consists of a **semantic** (rules to interpret it)

- How to evaluate truth?
  - Start by considering the domain as a *database* made with many tables (one for each $P^I$), then..
  
  $P(t_1, \ldots, t_n)^{I,A} = \top$ *if and only if* $(t_1^{I,A}, \ldots, t_n^{I,A}) \in P^I$
A language consists of a **semantic** (rules to interpret it)

- How to evaluate truth?
  - Start by considering the domain as a *database* made with many tables (one for each $P^I$), then..
    $$P(t_1, ..., t_n)^{I,A} = \text{V if and only if } (t_1^{I,A}, ..., t_n^{I,A}) \in P^I$$

- $T^{I,A} = \text{V}; \perp^{I,A} = \text{F}$
- $(\neg X^{I,A}) = \neg X^{I,A}$
- $(X \circ Y)^{I,A} = X^{I,A} \bullet Y^{I,A}$ for coupled operators $\circ$ and $\bullet$
- $((\forall x)\ F)^{I,A} = \text{ V if and only if } F^{I,B} = \text{ V}$ for all assignation $B$ equal to $A$ save for $x$.
- $((\exists x)\ F)^{I,A} = \text{ V if and only if } F^{I,B} = \text{ V}$ for (at least) one assignation $B$ equal to $A$ save for $x$. 
Semantics of formulas

• A formula $F$ is *true* in $M(D,I)$
  if $F^I_A = V$ for all assignments $A$. 
Semantics of formulas

- A formula F is **true** in $M(D,I)$
  if $F^{I,A} = V$ for all assignments $A$.

- A formula F is **valid**
  if F is true in any $M(D,I)$.
Semantics of formulas

• A formula $F$ is **true** in $M(D,I)$
  if $F^{I,A} = V$ for all assignments $A$.

• A formula $F$ is **valid**
  if $F$ is true in any $M(D,I)$.

• A set $S$ of formulas is **satisfiable** in $M(D,I)$
  if there exists (at least) an assignment $A$ such
  that $F^{I,A} = V$ for all $F$ belonging to $S$.

**NOTA BENE:** A formula $F$ is **valid**
if and only if $\{ \neg F \}$ is not satisfiable.
Example

• Consider
  – the domain $D = \{"France", "Vatican", "Japan", "to have diplomatic relations with"\}$
  – in the interpretation $I$ which matches $f$ to $f^I$ as "France", ...
    $D$ to the only relation of $D$,

• We can then evaluate the truth and validity of formula as $D(x, y)$, $D(\text{france, vatican})$, and all their combinations
Herbrand model

- A model $M(D, I)$ for a first-order language $L$ is an Herbrand model, if and only if:
  - $D$ contains only closed terms of $L$
  - For each closed term $t$, $t^I = t$

Noting with $F\{v/d\}$ the outcome of a substitution of $v$ by $d$ in $F$ (note that $F\{v/d\}$ is still a formula!)

In an Herbrand model:
- For any formula $F$, $(\forall v) \ F \text{ is true}$ if and only if $F\{v/d\}$ is true for any $d \in D$
- For any formula $F$, $(\exists v) \ F \text{ is true}$ if and only if $F\{v/d\}$ it true for at least a $d \in D$
Replacing quantifiers

• $M(D, I)$ is an Herbrand Model for a first order language $L$:

• If $\gamma$ is a formula of $L$, $\gamma$ is true in $M$ if and only if $\gamma(d)$ is true for any $d \in D$;

• If $\delta$ is a formula of $L$, $\delta$ is true in $M$ if and only if $\delta(d)$ is true for (at least a) $d \in D$

\[
\begin{align*}
\gamma\text{-rule} & : & \gamma(d) \\
(\forall x) F & : & F\{x/d\} \\
\neg(\exists x) F & : & \neg F\{x/d\}
\end{align*}
\]

\[
\begin{align*}
\delta\text{-rule} & : & \delta(d) \\
(\exists x) F & : & F\{x/d\} \\
\neg(\forall x) F & : & \neg F\{x/d\}
\end{align*}
\]
Example of resolution

$((\forall x) \ (P(x) \lor Q(x)) \supset ((\exists x) \ P(x) \lor (\forall x) \ Q(x)))$

start from the negated formula (refutation):
1. $[(\lnot((\forall x) \ (P(x) \lor Q(x)) \supset ((\exists x) \ P(x) \lor (\forall x) \ Q(x))))]$


Example of resolution

\[((\forall x) \ (P(x) \lor Q(x)) \supset ((\exists x) \ P(x) \lor (\forall x) \ Q(x)))\]

start from the negated formula (refutation):

1.  \[\neg((\forall x) \ (P(x) \lor Q(x)) \supset ((\exists x) \ P(x) \lor (\forall x) \ Q(x))))\]
2.  \[((\forall x) \ (P(x) \lor Q(x)))\] development of 1. (α-rule)
3.  \[\neg((\exists x) \ P(x) \lor (\forall x) \ Q(x)))\] development of 1. (α-rule)
Example of resolution

$$((\forall x) \ (P(x) \lor Q(x)) \supset ((\exists x) \ P(x) \lor (\forall x) \ Q(x)))$$

start from the negated formula (refutation):

1. $$\neg(((\forall x) \ (P(x) \lor Q(x)) \supset ((\exists x) \ P(x) \lor (\forall x) \ Q(x))))$$

2. $$((\forall x) \ (P(x) \lor Q(x)))$$ development of 1. ($\alpha$-rule)

3. $$\neg(((\exists x) \ P(x) \lor (\forall x) \ Q(x)))$$ development of 1. ($\alpha$-rule)

4. $$\neg((\exists x) \ P(x))$$ development of 3. ($\alpha$-rule)

5. $$\neg((\forall x) \ Q(x))$$ development of 3. ($\alpha$-rule)
Example of resolution

\[ ((\forall x) (P(x) \lor Q(x)) \Rightarrow ((\exists x) P(x) \lor (\forall x) Q(x))) \]

start from the negated formula (refutation):

1. \[ \neg((\forall x) (P(x) \lor Q(x)) \Rightarrow ((\exists x) P(x) \lor (\forall x) Q(x))) \]
2. \[ ((\forall x) (P(x) \lor Q(x))) \] development of 1. (\(\alpha\)-rule)
3. \[ \neg((\exists x) P(x) \lor (\forall x) Q(x))) \] development of 1. (\(\alpha\)-rule)
4. \[ \neg(\exists x) P(x) \] development of 3. (\(\alpha\)-rule)
5. \[ \neg(\forall x) Q(x) \] development of 3. (\(\alpha\)-rule)
6. \[ \neg Q(c) \] a version of 5. (\(\delta\)-rule)
Example of resolution

\((\forall x) (P(x) \lor Q(x)) \supset ((\exists x) P(x) \lor (\forall x) Q(x)))\)

start from the negated formula (refutation):

1. \([\neg((\forall x) (P(x) \lor Q(x)) \supset ((\exists x) P(x) \lor (\forall x) Q(x)))]\)  
2. \([(\forall x) (P(x) \lor Q(x))] \text{ development of 1. (\(\alpha\)-rule)}\)  
3. \([\neg((\exists x) P(x) \lor (\forall x) Q(x))] \text{ development of 1. (\(\alpha\)-rule)}\)  
4. \([\neg(\exists x) P(x)] \text{ development of 3. (\(\alpha\)-rule)}\)  
5. \([\neg(\forall x) Q(x)] \text{ development of 3. (\(\alpha\)-rule)}\)  
6. \([\neg Q(c)] \text{ a version of 5. (\(\delta\)-rule)}\)  
7. \([\neg P(c)] \text{ a version of 4. (\(\gamma\)-rule, taking the same c)}\)
Example of resolution

\(((\forall x) (P(x) \lor Q(x))) \supset ((\exists x) P(x) \lor (\forall x) Q(x)))\)

start from the negated formula (refutation):

1. \([-((\forall x) (P(x) \lor Q(x))) \supset ((\exists x) P(x) \lor (\forall x) Q(x)))\]
2. \[(\forall x) (P(x) \lor Q(x))\] development of 1. (α-rule)
3. \([-((\exists x) P(x) \lor (\forall x) Q(x))\] development of 1. (α-rule)
4. \([-((\exists x) P(x))\] development of 3. (α-rule)
5. \([-((\forall x) Q(x))\] development of 3. (α-rule)
6. \([-Q(c)]\) a version of 5. (δ-rule)
7. \([-P(c)]\) a version of 4. (γ-rule, taking the same c)
8. \[(P(c) \lor Q(c))]\] a version of 2. (γ-rule, taking the same c)
Example of resolution

\[(\forall x) (P(x) \lor Q(x)) \supset ((\exists x) P(x) \lor (\forall x) Q(x)))\]

start from the negated formula (refutation):

1. \[\neg((\forall x) (P(x) \lor Q(x)) \supset ((\exists x) P(x) \lor (\forall x) Q(x))))\]

2. \[(\forall x) (P(x) \lor Q(x))\] development of 1. (\(\alpha\)-rule)

3. \[\neg((\exists x) P(x) \lor (\forall x) Q(x))\] development of 1. (\(\alpha\)-rule)

4. \[\neg(\exists x) P(x)\] development of 3. (\(\alpha\)-rule)

5. \[\neg(\forall x) Q(x)\] development of 3. (\(\alpha\)-rule)

6. \[\neg Q(c)\] a version of 5. (\(\delta\)-rule)

7. \[\neg P(c)\] a version of 4. (\(\gamma\)-rule, taking the same \(c\))

8. \[(P(c) \lor Q(c))\] a version of 2. (\(\gamma\)-rule, taking the same \(c\))

9. \[P(c), Q(c)\] modification of 9. (\(\beta\)-rule)
Example of resolution

\[(\forall x) (P(x) \lor Q(x)) \supset ((\exists x) P(x) \lor (\forall x) Q(x))\]

Start from the negated formula (refutation):

1. \[\neg((\forall x) (P(x) \lor Q(x)) \supset ((\exists x) P(x) \lor (\forall x) Q(x)))\]
2. \[((\forall x) (P(x) \lor Q(x)))\] development of 1. (\(\alpha\)-rule)
3. \[\neg((\exists x) P(x) \lor (\forall x) Q(x)))\] development of 1. (\(\alpha\)-rule)
4. \[\neg(\exists x) P(x)\] development of 3. (\(\alpha\)-rule)
5. \[\neg(\forall x) Q(x)\] development of 3. (\(\alpha\)-rule)
6. \[\neg\neg Q(c)\] a version of 5. (\(\delta\)-rule)
7. \[\neg\neg P(c)\] a version of 4. (\(\gamma\)-rule, taking the same \(c\))
8. \[(P(c) \lor Q(c))\] a version of 2. (\(\gamma\)-rule, taking the same \(c\))
9. \[P(c), Q(c)\] modification of 9. (\(\beta\)-rule)
10. \[Q(c)\] resolving clause of 7. and 9.
Example of resolution

\(((\forall x) (P(x) \lor Q(x)) \supset ((\exists x) P(x) \lor (\forall x) Q(x)))\)

start from the negated formula (refutation):

1. \[\neg((\forall x) (P(x) \lor Q(x)) \supset ((\exists x) P(x) \lor (\forall x) Q(x)))\]

2. \[((\forall x) (P(x) \lor Q(x)))\] development of 1. (\(\alpha\)-rule)

3. \[\neg((\exists x) P(x) \lor (\forall x) Q(x))\] development of 1. (\(\alpha\)-rule)

4. \[\neg(\exists x) P(x)\] development of 3. (\(\alpha\)-rule)

5. \[\neg(\forall x) Q(x)\] development of 3. (\(\alpha\)-rule)

6. \[\neg Q(c)\] a version of 5. (\(\delta\)-rule)

7. \[\neg P(c)\] a version of 4. (\(\gamma\)-rule, taking the same \(c\))

8. \[((P(c) \lor Q(c)))\] a version of 2. (\(\gamma\)-rule, taking the same \(c\))

9. \[P(c), Q(c)\] modification of 9. (\(\beta\)-rule)

10. \[Q(c)\] resolving clause of 7. and 9.

11. \[\] resolving clause of 6. and 10.
Prenex form

- A formula is in *prenex* form if it is written as a sequence of quantifiers (prefix) followed by a quantifier-free part (matrix).

\[(Q_1 x_1) \ldots (Q_n x_n) M\]

\[Q_i \in \{\forall, \exists\}\]
Prenex form

- A formula is in **prenex** form if it is written as a sequence of quantifiers (prefix) followed by a quantifier-free part (matrix).

$$(Q_1 \ x_1) \ldots (Q_n \ x_n) \ M$$

$Q_i \in \{\forall, \exists\}$
Prenex form

- A formula is in **prenex** form if it is written as a sequence of quantifiers (prefix) followed by a quantifier-free part (matrix).

\[(Q_1 x_1) \ldots (Q_n x_n) M\]

\[Q_i \in \{\forall, \exists\}\]

**non-deterministic process**

a good practice: existential quantifiers in the *leftmost* possible positions.
Example

\[(\forall x) \ (\exists y) \ \neg (A(x) \supset A(y))\]

\[\neg (\exists x) \ A(x) \equiv (\forall x) \ \neg A(x)\]

\[\neg (\forall x) \ A(x) \equiv (\exists x) \ \neg A(x)\]
Example

\[(\forall x) \ (\exists y) \ \neg (A(x) \supset A(y))\]

1. \[(\forall x) \ \neg (\forall y) \ (A(x) \supset A(y))\]

\[\neg (\exists x) \ A(x) \equiv (\forall x) \ \neg A(x)\]
\[\neg (\forall x) \ A(x) \equiv (\exists x) \ \neg A(x)\]
Example

\((\forall x)(\exists y)\neg(A(x) \supset A(y))\)

1. \((\forall x)\neg(\forall y)(A(x) \supset A(y))\)
2. \((\forall x)\neg(A(x) \supset (\forall y)A(y))\)

\(\neg(\exists x)A(x) \equiv (\forall x)\neg A(x)\)
\(\neg(\forall x)A(x) \equiv (\exists x)\neg A(x)\)
Example

\((\forall x) \ (\exists y) \ \neg (A(x) \supset A(y))\)

1. \((\forall x) \ \neg (\forall y) \ (A(x) \supset A(y))\)
2. \((\forall x) \ \neg (A(x) \supset (\forall y) A(y))\)
3. \(\neg (\exists x) \ (A(x) \supset (\forall y) A(y))\)

\[
\neg (\exists x) A(x) \equiv (\forall x) \neg A(x)
\]

\[
\neg (\forall x) A(x) \equiv (\exists x) \neg A(x)
\]
Example

\[(\forall x) \ (\exists y) \ \neg (A(x) \supset A(y))\]

1. \[(\forall x) \ \neg (\forall y) \ (A(x) \supset A(y))\]
2. \[(\forall x) \ \neg (A(x) \supset (\forall y) \ A(y))\]
3. \[\neg (\exists x) \ (A(x) \supset (\forall y) \ A(y))\]
4. \[\neg ((\forall x) \ A(x) \supset (\forall y) \ A(y))\]

\[\neg (\exists x) \ A(x) \equiv (\forall x) \ \neg A(x)\]

\[\neg (\forall x) \ A(x) \equiv (\exists x) \ \neg A(x)\]
Example

\((\forall x) (\exists y) \neg(A(x) \supset A(y))\)

1. \((\forall x) \neg(\forall y) (A(x) \supset A(y))\)
2. \((\forall x) \neg(A(x) \supset (\forall y) A(y))\)
3. \(\neg(\exists x) (A(x) \supset (\forall y) A(y))\)
4. \(\neg((\forall x) A(x) \supset (\forall y) A(y))\)
5. \(\neg(\forall y) ((\exists x) A(x) \supset A(y))\)

\(\neg(\exists x) A(x) \equiv (\forall x) \neg A(x)\)
\(\neg(\forall x) A(x) \equiv (\exists x) \neg A(x)\)
Example

$(\forall x) \ (\exists y) \ \neg (A(x) \supset A(y))$

1. $(\forall x) \ \neg (\forall y) \ (A(x) \supset A(y))$
2. $(\forall x) \ \neg (A(x) \supset (\forall y) \ A(y))$
3. $\neg (\exists x) \ (A(x) \supset (\forall y) \ A(y))$
4. $\neg ((\forall x) \ A(x) \supset (\forall y) \ A(y))$
5. $\neg (\forall y) \ ((\exists x) \ A(x) \supset A(y))$
6. $\neg (\forall y) \ (\exists x) \ (A(x) \supset A(y))$

$\neg (\exists x) \ A(x) \equiv (\forall x) \ \neg A(x)$

$\neg (\forall x) \ A(x) \equiv (\exists x) \ \neg A(x)$
Example

$$(\forall x) \ (\exists y) \ \neg(A(x) \supset A(y))$$

1. $$(\forall x) \ \neg(\forall y) \ (A(x) \supset A(y))$$
2. $$(\forall x) \ \neg(A(x) \supset (\forall y) \ A(y))$$
3. $$\neg(\exists x) \ (A(x) \supset (\forall y) \ A(y))$$
4. $$\neg((\forall x) \ A(x) \supset (\forall y) \ A(y))$$
5. $$\neg(\forall y) \ ((\exists x) \ A(x) \supset A(y))$$
6. $$\neg(\forall y) \ (\exists x) \ (A(x) \supset A(y))$$
7. $$(\exists y) \ (\forall x) \ \neg(A(x) \supset A(y))$$

$$\neg(\exists x) \ A(x) \equiv (\forall x) \ \neg A(x)$$
$$\neg(\forall x) \ A(x) \equiv (\exists x) \ \neg A(x)$$
Skolemization

- Skolemization is a transformation in which a formula in prenex form, as

\[(Q_1x_1) (Q_2x_2) \ldots (\exists x_k) \ldots (Q_{nx_n}) F\]

is transformed in

\[(Q_1x_1) (Q_2x_2) \ldots (Q_{nx_n}) F\{x_k/f(x_1, x_2 \ldots x_{k-1})\}\]

where \(f\) is a new functor (called \textit{Skolem function}) that does not belong to the language.
Skolemization

- Skolemization is a transformation in which a formula in prenex form, as
  
  $$(Q_1 x_1) (Q_2 x_2) \ldots (\exists x_k) \ldots (Q_n x_n) F$$

  is transformed in

  $$(Q_1 x_1) (Q_2 x_2) \ldots (Q_n x_n) F\{x_k/f(x_1, x_2 \ldots x_{k-1})\}$$

  where f is a new functor (called *Skolem function*) that does not belong to the language.

  (Also free variables should be put in f.)
Skolemization

• Skolemization is a transformation in which a formula in prenex form, as

\[(Q_1x_1) (Q_2x_2) ...(\exists x_k)...(Q_{nxn}) F\]

is transformed in

\[(Q_1x_1) (Q_2x_2) ...(Q_{nxn}) F{xk/f(x_1, x_2 ...x_{k-1})}\]

where f is a new functor (called *Skolem function*) that does not belong to the language.

(Also free variables should be put in f.)

• The two formulas are not equivalent, but they have the same satisfiability.
Skolemization

- Skolemization is a transformation in which a formula in prenex form, as

$$(Q_1x_1) (Q_2x_2) ... (\exists x_k) ...(Q_{nxn}) F$$

is transformed in

$$(Q_1x_1) (Q_2x_2) ... (Q_{nxn}) F\{x_k/f(x_1, x_2 ... x_{k-1})\}$$

where $f$ is a new functor (called **Skolem function**) that does not belong to the language.

**non-deterministic process**

a good practice: start from external quantifiers
Herbrand model lemma

- Let $S$ be a set of statements in Skolem form

$S$ has a model
if and only if
$S$ has a Herbrand model
To prove the validity of a formula F:

- rename variables if necessary,
  e.g. \(((\forall x) \ p(x) \supset (\forall x) \ r(x))\) in \(((\forall x) \ p(x) \supset (\forall y) \ r(y))\)
- transform \(\neg F\) in prenex form
- skolemize
- remove the quantifiers
- transform in CNF
- use the resolution method
- apply unification
Si... alors...

Quoi, «si alors» ?...

Logique formelle... t'as plus qu'à remplir.