Logic and Knowledge Representation

Propositional Logic
4 May 2018

Giovanni Sileno gsileno@enst.fr
Télécom ParisTech, Paris-Dauphine University
LA VÉRITÉ
SORT DE LA BOUCHE
DES ENFANTS

C'EST PAS VRAI !
This is a nice variation of Epimenides paradox, the story of a Cretan saying “All Cretans are liars”.
Logic: a long history
Overview on (Western) logic

• Greek Logic
  – Stoics
  – Aristotle
    – *logic in argumentation*
    – *syllogism*
Overview on (Western) logic

• Greek Logic
  – Stoics
  – Aristotle
    – logic in argumentation
    – syllogism

• Medieval and traditional logic
  – Thomas Aquinas (1225-1274)
    – modal logic
  – William of Ockham (1288-1348)
    – laws of de Morgan
    – ternary logic
  – Logic of Port Royal
    • Antoine Arnauld & Pierre Nicole (1662)
All bankers are athletes.
No consultant is a banker.
Therefore....
All bankers are athletes.
No consultant is a banker. 
Therefore....

*some athlete is not a consultant (???)*
All bankers are athletes. No consultant is a banker. Therefore....

*some athlete is not a consultant!*

- Valid, and not trivial.
All bankers are athletes.
No consultant is a banker.
Therefore....

*some athlete is not a consultant.*

• Valid, and not trivial.
All bankers are athletes.
No consultant is a banker.
Therefore...

**some athlete is not a consultant.**

- **Valid**, and not trivial.
All bankers are athletes.
No consultant is a banker.
Therefore...

some athlete is not a consultant.

• Valid, and not trivial.
All bankers are athletes.
No consultant is a banker.
Therefore....

*some athlete is not a consultant.*

- Valid, and not trivial.
Overview on (Western) logic

- **Modern logic**
  - Descartes, Leibniz
  - George Boole (1848)
  - Gottlob Frege, *Begriffsschrift* (1879)
    - Quantification
  - Charles Peirce
    - Reasoning and logic
  - Giuseppe Peano
    - Logical Axiomatization of Arithmetic
  - Bertrand Russell & Alfred N. Whitehead, *Principia Mathematica* (1925)
    - Logical Axiomization of Mathematics
Propositional logic
A language consists of \textbf{symbols}, ...

- \textit{alphabet}
  - \textit{propositional symbols}
    \texttt{p1, p2, ...}
A language consists of symbols, ...

- **alphabet**
  - *propositional symbols*
    - $p_1, p_2, ...$
  - *connectives*
    - nullary: $T$, $\perp$ (top, bottom)
    - unary: $\neg$ (negation)
    - binary: $\land, \lor, \supset, \subset, \uparrow, \downarrow, \notin, \not\in, \equiv, \neq$ (and, or, implies, only-if, nand (incompatible), nor, not-implies, not-only-if, equivalent, not-equivalent)
A language consists of a **syntax** (rules to aggregate symbols), ...

- set $A$ of atomic formulas
  - $A$ contains all propositional symbols
  - $A$ contains the nullary connectives $T, \bot$
A language consists of a **syntax** (rules to aggregate symbols), ...

- **set A of atomic formulas**
  - A contains all propositional symbols
  - A contains the nullary connectives $T$, $\bot$

- **set P of (well-formed) propositional formulas**
  - P contains atomic formulas
  - if $F$ is in P, then $\neg F$ is in P
  - if $F$ and $G$ are in P, then $(F \circ G)$ is in P, where $\circ$ is a binary connective ($\land$, $\lor$, $\supset$, $\subset$, $\uparrow$, $\downarrow$, $\phi$, $\emptyset$, $\equiv$, $\neq$).
  - P is the smallest set that has these properties (equivalently, there is nothing in P that does not satisfy these properties)
A language consists of a **semantic** (rules to interpret its expressions)

- Semantics should tell us how the meaning of the *constituent parts of a discourse*, and their *mode of combination*, determine the overall meaning.
A language consists of a **semantic** (rules to interpret its expressions)

- Semantics should tell us how the meaning of the *constituent parts of a discourse*, and their *mode of combination*, determine the overall meaning.

- But what do we mean by *meaning*?
A language consists of a **semantic** (rules to interpret its expressions)

- Semantics should tell us how the meaning of the *constituent parts of a discourse*, and their *mode of combination*, determine the overall meaning.

- But what do we mean by *meaning*? ex. “there is a dog”

*correspondence semantics* ...
...that a dog is out there
A language consists of a **semantic** (rules to interpret its expressions)

- Semantics should tell us how the meaning of the *constituent parts of a discourse*, and their *mode of combination*, determine the overall meaning.

- But what do we mean by **meaning**? ex. “there is a dog”

  *correspondence semantics*  
  ...that a dog is out there

  *truth-conditional semantics*  
  ..that that proposition is true

  “there is a dog” is true
A language consists of a **semantic** (rules to interpret its expressions)

- Semantics should tell us how the meaning of the **constituent parts of a discourse**, and their **mode of combination**, determine the overall meaning.

- But what do we mean by **meaning**?
  - ex. “there is a dog”

  **correspondence semantics**
  ...that a dog is out there

  **cognitive semantics**
  ..that the locutor believes that..

  **truth-conditional semantics**
  ..that that proposition is true

  “there is a dog” is true
Strange effects...

- “a dog is a dog”
- “a dog is a mammal”
Strange effects...

Each sentence is assigned to a truth value

- "a dog is a dog" → TRUE
- "a dog is a mammal" → TRUE
Strange effects...

Each sentence is assigned to a truth value

- “a dog is a dog” → TRUE
- “a dog is a mammal” → TRUE

Under truth-conditional semantics, they have the same “meaning”!
Strange effects...

Each sentence is assigned to a truth value

- “a dog is a dog” → TRUE
- “a dog is a mammal” → TRUE

Under truth-conditional semantics, they have the same “meaning”!

*truth-conditional semantics is prone to logic solipsism*
Truth space and functions

- Truth space: $T_v = \{T, F\}$
Truth space and functions

- Truth space: $Tv = \{T, F\}$
- Truth functions:
  - 2 Nullary functions: $T, F$
Truth space and functions

- Truth space: $Tv = \{T, F\}$
- Truth functions:
  - 2 Nullary functions: $T, F$
  - 1 Unary functions: $Tv \rightarrow Tv$

\[
\begin{array}{c|c}
\text{not} & \\
T & F \\
F & T \\
\end{array}
\]
Truth space and functions

- Truth space: \( T_v = \{T, F\} \)
- Truth functions:
  - 2 Nullary functions: \( T, F \)
  - 1 Unary functions: \( T_v \to T_v \)
  - 16 Binary functions: \( T_v \times T_v \to T_v \)

<table>
<thead>
<tr>
<th></th>
<th>( T )</th>
<th>( F )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td>( T )</td>
<td>( F )</td>
</tr>
<tr>
<td>( F )</td>
<td>( F )</td>
<td>( T )</td>
</tr>
</tbody>
</table>

or

<table>
<thead>
<tr>
<th></th>
<th>( T )</th>
<th>( F )</th>
<th>( T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td>( T )</td>
<td>( F )</td>
<td>( T )</td>
</tr>
<tr>
<td>( T )</td>
<td>( F )</td>
<td>( T )</td>
<td>( T )</td>
</tr>
<tr>
<td>( F )</td>
<td>( T )</td>
<td>( T )</td>
<td>( T )</td>
</tr>
<tr>
<td>( F )</td>
<td>( F )</td>
<td>( T )</td>
<td>( F )</td>
</tr>
</tbody>
</table>

and

<table>
<thead>
<tr>
<th></th>
<th>( T )</th>
<th>( F )</th>
<th>( T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td>( T )</td>
<td>( F )</td>
<td>( F )</td>
</tr>
<tr>
<td>( T )</td>
<td>( F )</td>
<td>( F )</td>
<td>( F )</td>
</tr>
<tr>
<td>( F )</td>
<td>( T )</td>
<td>( F )</td>
<td>( F )</td>
</tr>
<tr>
<td>( F )</td>
<td>( F )</td>
<td>( F )</td>
<td>( F )</td>
</tr>
</tbody>
</table>

imp
Connecting syntax with semantics

• **Boolean valuation**, a function mapping propositions to truth values: \( v: P \rightarrow T_v \)

  - \( v(\top) = T \)
  - \( v(\bot) = F \)
  - \( v(\neg X) = \text{not } v(X) \)
  - \( v(X \circ Y) = v(X) \cdot v(Y) \)

  \( \circ \text{ syntactic connectives} \quad \neg \quad \land \quad \lor \quad \Rightarrow, \rightarrow \quad \ldots. \)

  • **semantic connectives** \quad \text{not} \quad \text{and} \quad \text{or} \quad \text{implies, } \Rightarrow \quad \ldots.
syntax vs semantics

language objects
atoms and formulas

language objects
atoms and formulas

"a"

"b"

"world" objects, here
truth values

valuations

T

F
syntax vs semantics

language objects
atoms and formulas

"a"
"b"
"a ∧ b"

"world" objects, here
truth values

valuations

→ T
→ F

?
syntax vs semantics

language objects
atoms and formulas

"a"
"b"
"a \land b"

"world" objects, here
truth values

T
F

and

F and T

valuations

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
syntax vs semantics

language objects
atoms and formulas

“a”
“b”
“a ∧ b”

“world” objects, here
truth values

T
F
F and T

valuations
syntax vs semantics 2

language objects
symbols and formulas

"3"
"5"
"8"
"3 + 5"

"world" objects, here
counting items

interpretation

plus
syntax vs semantics 2

language objects
symbols and formulas

"3"
"5"
"8"
"3 + 5"

"world" objects, here counting items

language operator

"world" operator

plus

interpretation
% symbols of DSL and priority

:- op(800, fx, if).
:- op(700, xfx, then).
:- op(300, xfy, or).
:- op(200, xfy, and).

% backward chaining rule interpreter

is_true(P) :-
    fact(P).

is_true(P) :-
    if Condition then P,
    is_true(Condition).

is_true(P1 and P2) :-
    is_true(P1), is_true(P2).

is_true(P1 or P2) :-
    is_true(P1) ; is_true(P2).

% knowledge base

If cloud then rain.
If rain then wet.
If sprinkler then wet.

fact(sprinkler).

language objects?
language operators?
"world" objects?
"world" operators?
"I know where that plane is going. To an airport."

Tautologies & co.
Tautology, satisfiability, consequence

- A propositional formula $F$ is a **tautology** if $v(F) = T$ for any Boolean valuation $v$.
Tautology, satisfiability, consequence

- A propositional formula $F$ is a **tautology** if $v(F) = T$ for any Boolean valuation $v$

inputs (associated to factors)

\[
\begin{array}{c|c|c|c}
T & F & x & F(x, y, z, \ldots) \\
T & F & y & F(x, y, z, \ldots) \\
T & F & z & F(x, y, z, \ldots) \\
\ldots & \\
\end{array}
\]

Functional view:
any configuration of inputs brings the same outcome $T$
Tautology, satisfiability, consequence

• A propositional formula $F$ is a **tautology** if $v(F) = T$ for any Boolean valuation $v$

• A set $S$ of propositional formulas is **satisfiable** if *some* valuation $v_i$ maps every member of $S$ to $T$:
  - $v_i(F) = T$ for all $F$ of $S$. 
Tautology, satisfiability, consequence

- A propositional formula $F$ is a **tautology** if $v(F) = T$ for any Boolean valuation $v$
- A set $S$ of propositional formulas is **satisfiable** if some valuation $v_i$ maps every member of $S$ to $T$:
  - $v_i(F) = T$ for all $F$ of $S$.

Functional view:
*There is a configuration of inputs making all outputs $T.*
Tautology, satisfiability, consequence

- A propositional formula $F$ is a **tautology** if $v(F) = T$ for any Boolean valuation $v$.
- A set $S$ of propositional formulas is **satisfiable** if some valuation $v_i$ maps every member of $S$ to $T$:
  - $v_i(F) = T$ for all $F$ of $S$.

$$S = \{ F_1, F_2, ... \}$$

Functional view:
*There is a configuration of inputs making $T$ the output of the conjunction of the formula in $S$.*
Tautology, satisfiability, consequence

• $S \models C$ is called **semantic consequence**: if a valuation assigns the value $T$ to all element of $S$, then it will assign $T$ to $C$. 
Tautology, satisfiability, consequence

- $S \models C$ is called **semantic consequence**: if a valuation assigns the value $T$ to all elements of $S$, then it will assign $T$ to $C$.

Functional view:

*All configurations making $T$ the outputs of $S$ make $C$ true.*
Tautology, satisfiability, consequence

- $S \models C$ is called **semantic consequence**: if a valuation assigns the value $T$ to all element of $S$, then it will assign $T$ to $C$

Functional view:

*All configurations making $T$ the outputs of $S$ make $C$ true.*

*not the inverse!!*
Tautology, satisfiability, consequence

- $S \models C$ is called **semantic consequence**: if a valuation assigns the value $T$ to all element of $S$, then it will assign $T$ to $C$
- $\models C$ denotes the fact that $C$ is a **tautology**.

\[
\begin{array}{c}
T \quad \Rightarrow \quad T \quad \Rightarrow \quad C(x, y, z, ...) \quad \Rightarrow \quad T
\end{array}
\]
Tautology, satisfiability, consequence

• $S \models C$ if a valuation assigns the value $T$ to any element of $S$, then it will assign $T$ to $C$

• $\models C$ $C$ is a tautology.

Exercises (1):

– Show that $X$ is a tautology only if $(X \equiv T)$ is a tautology

– Show that $(\neg(X \land Y) \equiv (\neg X \lor \neg Y))$ is a tautology
Tautology, satisfiability, consequence

• $S \models C$ if a valuation assigns the value $T$ to any element of $S$, then it will assign $T$ to $C$
• $\models C$ $C$ is a *tautology*.

Exercises (2):

– *(ex falso quodlibet sequitur)*. if $A, \neg A \in S$, then for any $X : S \models X$.

– *(monotonicity)*. if $S \models X$, then $S \cup \{Y\} \models X$
Tautology, satisfiability, consequence

• \( S \models C \) if a valuation assigns the value \( T \) to any element of \( S \), then it will assign \( T \) to \( C \)
• \( \models C \) \( C \) is a tautology.

Exercises (3):

– Show that \( S \models C \) entails that \( S \cup \{\neg C\} \) is not satisfiable.
– Show the reciprocal.
Tautology, satisfiability, consequence

- Central result:

  \[ \text{C is a semantic consequence of S, i.e. } S \models C \text{ if and only if } \] 
  
  \[ S \cup \{\neg C\} \text{ is not satisfiable} \]

\[ \text{the conjunction of the formulas in } S \text{ and the } \neg C \text{ is an antilogy or contradiction (false for all inputs)} \]
Replacement theorem

- Given $F(P)$, formula with any occurrences of symbol $P$

  if $(X \equiv Y)$ is a tautology, then $(F(X) \equiv F(Y))$ is a tautology as well.
Replacement theorem

- Given $F(P)$, formula with any occurrences of symbol $P$

if $(X \equiv Y)$ is a tautology, then $(F(X) \equiv F(Y))$ is a tautology as well.

Proof. If $(X \equiv Y)$ is a tautology, then $v(X) = v(Y)$ for any evaluation $v$, but then also $v(F(X)) = v(F(Y))$. As $v(F(X)) = v(F(Y))$ for any $v$, $(F(X) \equiv F(Y))$ is a tautology.
Replacement theorem

• Given $F(P)$, formula with any occurrences of symbol $P$

If $(X \equiv Y)$ is a tautology, then $(F(X) \equiv F(Y))$ is a tautology as well.

Exercises:

- (double negation) Show that $(X \equiv \neg\neg X)$ is a tautology

- (modus ponens) Show that $Y$ is a tautology if $X$ and $(X \supset Y)$ are tautologies
Normal Forms (CNF and DNF)
Rewriting of formulas

• Any number may be computed as
  – product of sums, e.g.
    $$8 = (1 + 1) \times (2 + 2)$$
  – sums of products, e.g.
    $$8 = (2 \times 1) + (2 \times 1) + (2 \times 1) + (2 \times 1)$$
Rewriting of formulas

• Any number may be computed as
  - product of sums, e.g.
    \[ 8 = (1 + 1) \times (2 + 2) \]
  - sums of products, e.g.
    \[ 8 = (2 \times 1) + (2 \times 1) + (2 \times 1) + (2 \times 1) \]

• In the algebraic interpretation of boolean logic,
  - \textit{conjunction } \land \textit{ stands for product } \times
  - \textit{disjunction } \lor \textit{ stands for addition } +
Rewriting of formulas

- Any formula may be rewritten as
  - conjunction of disjunctions (CNF)
  - disjunction of conjunctions (DNF)

- In the algebraic interpretation of boolean logic,
  - conjunction $\land$ stands for product $\ast$
  - disjunction $\lor$ stands for addition $+$
Rewriting of formulas

- Any formula may be rewritten as
  - conjunction of disjunctions (CNF)
  - disjunction of conjunctions (DNF)

\[
\begin{array}{|c|c|c|c|}
\hline
a & b & F & \neg F \\
\hline
T & T & T & F \\
T & F & F & T \\
F & T & T & F \\
F & F & T & F \\
\hline
\end{array}
\]

For instance from truth tables:

\[
\begin{align*}
\text{DNF}_F &= (a \land b) \lor (a \land \neg b) \lor (\neg a \land \neg b) \\
\text{CNF}_F &= \neg \text{DNF}_F = \neg (a \land \neg b) = \neg a \lor b
\end{align*}
\]
Conjunctive normal form

- The conjunctive normal form (CNF) rewrites any propositional formula as a conjunction of clauses.
Conjunctive normal form

• The conjunctive normal form (CNF) rewrites any propositional formula as a conjunction of **clauses**.

• A clause is a disjunction of propositional symbols possibly with negation. It is noted as \([a,b,c]\).
Conjunctive normal form

• The conjunctive normal form (CNF) rewrites any propositional formula as a conjunction of clauses.

• A clause is a disjunction of propositional symbols possibly with negation. It is noted as \([a, b, c]\).

• A conjunction of clauses is noted \(<C1, C2, C3>\).
Conjunctive normal form

- The conjunctive normal form (CNF) rewrites any propositional formula as a conjunction of clauses.
- A clause is a disjunction of propositional symbols possibly with negation. It is noted as \([ a, b, c ]\).
- A conjunction of clauses is noted \(< C_1, C_2, C_3 >\).

Propositional formula \(\neg(a \supset \neg(b \supset c))\) rewrites to CNF as \(< [\neg a, b], [\neg a, \neg c] >\).
Evaluation with CNF

• Evaluations are performed as follows:

  - \( v([X_1, X_2, \ldots, X_n]) = F \) if and only if \( v(X_i) = F \) for all \( i \)
  
  - \( v(<C_1, C_2, \ldots, C_m>) = T \) if and only if \( v(C_i) = T \) for all \( i \)

  - empty clause: \( v([ ]) = F \)
  
  - empty conjunction: \( v(<>) = T \).
Transforming a formula to CNF

<table>
<thead>
<tr>
<th>α-formula</th>
<th>α1</th>
<th>α2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X ∧ Y)</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>(¬(X ∨ Y))</td>
<td>¬X</td>
<td>¬Y</td>
</tr>
<tr>
<td>(¬(X ⊃ Y))</td>
<td>X</td>
<td>¬Y</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>β-formula</th>
<th>β1</th>
<th>β2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X ∨ Y)</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>(¬(X ∧ Y))</td>
<td>¬X</td>
<td>¬Y</td>
</tr>
<tr>
<td>(X ⊃ Y)</td>
<td>¬X</td>
<td>Y</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- The algorithm that converts a propositional formula into CNF proceeds sequentially with these steps:

  replace < ... [ ... β ...]... > by < ... [ ... β1, β2 ...] ... >
  replace < ... [ ... α ...]... > by < ... [ ... α1 ...], [ ... α2 ...] ... >
  replace < ... [ ... ¬¬α ...]... > by < ... [ ... α ...] ... >
Example

• Transform \(((A \implies (B \implies C)) \implies ((A \implies B) \implies (A \implies C)))\) to CNF
• Show that it is a tautology.
Automatic proof methods
Resolution method

- A **sequence** is the *conjunction* of lines.
- A **line** is a *disjunction* of propositional formulas.
Resolution method

• A sequence is the conjunction of lines.
• A line is a disjunction of propositional formulas.
• For a propositional formula X composed on the line L, the growth of the sequence consists of:
  – If X is of type $\beta$, replace it with $\beta_1$, $\beta_2$.
  – If X is of type $\alpha$, create two new lines L1 and L2, recopy the line L replacing $\alpha$ with $\alpha_1$ and $\alpha_2$ respectively.
Resolution method

• A sequence is the conjunction of lines.
• A line is a disjunction of propositional formulas.

• For a propositional formula X composed on the line L, the growth of the sequence consists of:
  – If X is of type $\beta$, replace it with $\beta_1, \beta_2$.
  – If X is of type $\alpha$, create two new lines L1 and L2, recopy the line L replacing $\alpha$ with $\alpha_1$ and $\alpha_2$ respectively.

• A resolution consists of concatenating two lines where X and $\neg X$ are separated, omitting all occurrences of these last two formulas. The new line is called the resolving clause of the other two.
Resolution method

• A sequence is the conjunction of lines.
• A line is a disjunction of propositional formulas.

• For a propositional formula X composed on the line L, the growth of the sequence consists of:
  – If X is of type $\beta$, replace it with $\beta_1$, $\beta_2$.
  – If X is of type $\alpha$, create two new lines L1 and L2, recopy the line L replacing $\alpha$ with $\alpha_1$ and $\alpha_2$ respectively.

• A resolution consists of concatenating two lines where X and $\neg X$ are separated, omitting all occurrences of these last two formulas. The new line is called the resolving clause of the other two.

• A proof by resolution of F is a sequence derived from $\neg\neg F$ and containing an empty clause $\square$. 
Example

Proof using the resolution method that:

\(((A \supset B) \land (B \supset C)) \supset \neg(\neg C \land A)\)

a. \[\neg(((A \supset B) \land (B \supset C)) \supset \neg(\neg C \land A))] \quad \text{negation of target}
Example

Proof using the resolution method that:

\( ((A \supset B) \land (B \supset C)) \supset \neg(\neg C \land A) \)

\[ \neg(((A \supset B) \land (B \supset C)) \supset \neg(\neg C \land A)) \]  \hspace{2cm} \text{negation of target}

b. \[ ((A \supset B) \land (B \supset C)) \]  \hspace{2cm} \text{development of a}

c. \[ (\neg C \land A) \]  \hspace{2cm} \text{development of a}
Example

Proof using the resolution method that:

\(((A \supset B) \land (B \supset C)) \supset \neg (\neg C \land A)\)

\[\neg(((A \supset B) \land (B \supset C)) \supset \neg (\neg C \land A))\] \hspace{1cm} \text{negation of target}

\[((A \supset B) \land (B \supset C))] \hspace{1cm} \text{development of a}

c. \[(\neg C \land A)] \hspace{1cm} \text{development of a}

d. \[(A \supset B)] \hspace{1cm} \text{development of b}

e. \[(B \supset C)] \hspace{1cm} \text{development of b}
Example

Proof using the resolution method that:

\(((A \supset B) \land (B \supset C)) \supset \neg(\neg C \land A)\)

- a. \([\neg(((A \supset B) \land (B \supset C)) \supset (\neg C \land A))]\) __negation of target__
- b. \([((A \supset B) \land (B \supset C))]\) __development of a__
- c. \([(\neg C \land A)]\) __development of a__
- d. \([(A \supset B)]\) __development of b__
- e. \([(B \supset C)]\) __development of b__
- f. \([\neg A, B]\) __rewriting d__
Example

Proof using the resolution method that:

\((A \supset B) \land (B \supset C) \supset \neg (\neg C \land A)\)

\(\neg(((A \supset B) \land (B \supset C)) \supset (\neg C \land A))\) \hspace{1cm} \text{negation of target}

\(((A \supset B) \land (B \supset C))\) \hspace{1cm} \text{development of a}

c. \((\neg C \land A)\) \hspace{1cm} \text{development of a}

d. \((A \supset B)\) \hspace{1cm} \text{development of b}

e. \((B \supset C)\) \hspace{1cm} \text{development of b}

f. \(\neg A, B\) \hspace{1cm} \text{rewriting d}

g. \(\neg B, C\) \hspace{1cm} \text{rewriting e}
Example

Proof using the resolution method that:

\((A \supset B) \land (B \supset C) \supset \neg (\neg C \land A)\)

a. \[\neg (((A \supset B) \land (B \supset C)) \supset \neg (\neg C \land A))\] \hspace{1cm} \text{negation of target}

b. \[((A \supset B) \land (B \supset C))] \hspace{1cm} \text{development of a}

c. \[(\neg C \land A)] \hspace{1cm} \text{development of a}

d. \[(A \supset B)] \hspace{1cm} \text{development of b}

e. \[(B \supset C)] \hspace{1cm} \text{development of b}

f. \[(\neg A, B] \hspace{1cm} \text{rewriting d}

g. \[(\neg B, C] \hspace{1cm} \text{rewriting e}

h. \[(\neg C] \hspace{1cm} \text{development of c}

i. \[A] \hspace{1cm} \text{development of c}
Example

Proof using the resolution method that:

\(((A \supset B) \land (B \supset C)) \supset \neg (\neg C \land A)\)

a. \[\neg(((A \supset B) \land (B \supset C)) \supset \neg (\neg C \land A))\] negation of target

b. \[((A \supset B) \land (B \supset C))\] development of a

c. \[\neg (\neg C \land A)\] development of a

d. \[\neg (\neg C \land A)\] development of b

e. \[(A \supset B)\] development of b

f. \[\neg A, B\] rewriting d

g. \[\neg B, C\] rewriting e

h. \[\neg C\] development of c

i. \[A\] development of c

j. \[B\] resolving f and i
Example

Proof using the resolution method that:

$$((A \supset B) \land (B \supset C)) \supset \neg(\neg C \land A)$$

\begin{align*}
a. \neg(((A \supset B) \land (B \supset C)) \supset \neg(\neg C \land A)) & \quad \text{negation of target} \\
b. ((A \supset B) \land (B \supset C)) & \quad \text{development of a} \\
c. (\neg C \land A) & \quad \text{development of a} \\
d. (A \supset B) & \quad \text{development of b} \\
e. (B \supset C) & \quad \text{development of b} \\
f. \neg A, B & \quad \text{rewriting d} \\
g. \neg B, C & \quad \text{rewriting e} \\
h. \neg C & \quad \text{development of c} \\
i. A & \quad \text{development of c} \\
j. B & \quad \text{resolving f and i} \\
k. C & \quad \text{resolving g and j}
\end{align*}
Example

Proof using the resolution method that:

\(((A \supset B) \land (B \supset C)) \supset \neg(\neg C \land A)\)

\begin{align*}
\text{a. } & \neg(((A \supset B) \land (B \supset C)) \supset \neg(\neg C \land A)) \text{ \textit{negation of target}} \\
\text{b. } & (((A \supset B) \land (B \supset C))) \text{ \textit{development of a}} \\
\text{c. } & (\neg C \land A) \text{ \textit{development of a}} \\
\text{d. } & (A \supset B) \text{ \textit{development of b}} \\
\text{e. } & (B \supset C) \text{ \textit{development of b}} \\
\text{f. } & \neg A, B \text{ \textit{rewriting d}} \\
\text{g. } & \neg B, C \text{ \textit{rewriting e}} \\
\text{h. } & \neg C \text{ \textit{development of c}} \\
\text{i. } & A \text{ \textit{development of c}} \\
\text{j. } & B \text{ \textit{resolving f and i}} \\
\text{k. } & C \text{ \textit{resolving g and j}} \\
\text{l. } & \text{ } \text{ \textit{resolving h and k}}
\end{align*}
Prolog and resolution

a. < [a], [b], [c, ¬a, ¬b],

b. CNF

c :- a, b. < [a], [b], [c, ¬a, ¬b],

?− c. [¬c] >
Complexity (time or space)

- SAT problem
  (check whether a boolean expression is satisfiable)
  - general case: NP-complete
  - with Horn clauses: $P$

$P \neq NP$
Tableaux method

- A **tree** represents the *disjunction* of branches.
- A **branch** represents a *conjunction* of propositional formulas.
Tableaux method

- A tree represents the disjunction of branches.
- A branch represents a conjunction of propositional formulas.

- For a propositional formula $X$ composed on the branch $B$, the growth of the tree consists of:
  - If $X$ is of type $\alpha$, add $\alpha_1$ then $\alpha_2$ at the end of $B$.
  - If $X$ is of type $\beta$, create a node and two new branches $B_1$, $B_2$ at the end of $B$, add $\beta_1$ and $\beta_2$ respectively.
Tableaux method

- A **tree** represents the *disjunction* of branches.
- A **branch** represents a *conjunction* of propositional formulas.

For a propositional formula $X$ composed on the branch $B$, the *growth* of the tree consists of:

- If $X$ is of type $\alpha$, add $\alpha_1$ then $\alpha_2$ at the end of $B$.
- If $X$ is of type $\beta$, create a node and two new branches $B_1$, $B_2$ at the end of $B$, add $\beta_1$ and $\beta_2$ respectively.

- A **branch** is **closed** if $X$ and $\neg X$ appear.
- A **tree** is **closed** if all its branches are closed.
- A proof tree for $F$ is a **closed** tree grew from $\{\neg F\}$. 
Syntaxic consequence

- $S \vdash X$, if $X$ can be proven from $S$.
- $\vdash X$, if $X$ admits a proof.

- **deduction theorem**:

  $S \cup \{X\} \vdash Y$ if and only if $S \vdash (X \supset Y)$
Syntaxic consequence

• $S \vdash X$, if $X$ can be proven from $S$.
• $\vdash X$, if $X$ admits a proof. ($X$ is said theorem)

• Proof the *modus ponens*: $\{ P, (P \Rightarrow Q) \} \vdash Q$
  
  $\{ (P \Rightarrow Q) \} \vdash (P \Rightarrow Q)$  trivial
  
  $\{ (P \Rightarrow Q) \} \cup \{ P \} \vdash Q$  for deduction theorem
Syntaxic consequence

• $S \vdash X$, if $X$ can be proven from $S$.
• $\vdash X$, if $X$ admits a proof. ($X$ is said theorem)

• $(((P \supset (Q \supset R)) \supset (Q \supset (P \supset R)))$ is a theorem:
  
  \[
  \begin{align*}
  \{ (P \supset (Q \supset R), P, Q \} & \vdash R \quad \text{applying twice modus ponens} \\
  \{ (P \supset (Q \supset R), Q \} & \vdash (P \supset R) \quad \text{deduction theorem} \\
  \{ (P \supset (Q \supset R) \} & \vdash (Q \supset (P \supset R)) \quad \text{deduction theorem} \\
  \vdash (((P \supset (Q \supset R) \supset (Q \supset (P \supset R)))) & \quad \text{deduction theorem}
  \end{align*}
  \]
Soundness, completeness

Let $F$ be any propositional formula and $S$ a set of propositional formulas (also called *axioms*),

- The logical system is **Sound**: if $S \vdash F$, then $S \models F$
  (all that can be proven is true, but there may be true propositions unproven)

- The logical system is **Complete**: if $S \models F$, then $S \vdash F$
  (whatever is true can be proven, but there may proof returning false propositions)
Consistency

Let $F$ be any propositional formula and $S$ a set of propositional formulas (also called axioms),

- A logical system is **Consistent** : if $S \vdash F$, then $S \not\vdash \neg F$
Gödel's incompleteness theorems

if $S$ is a logical system which contain elementary arithmetic, then $S$ is **incomplete**

*there are propositions that can be neither proved, neither disproved*
Gödel's incompleteness theorems

if $S$ is a logical system which contain elementary arithmetic, then $S$ is **incomplete**

*there are propositions that can be neither proved, neither disproved*

if $S$ is a logical system which contain elementary arithmetic, then $S \nvDash \text{Consistent}(S)$

*a system cannot proof its own consistency*