Propositional Logic

Propositional Logic - Syntax and Semantics and Inferences

Luciano Serafini[(1)]– Maurizio Lenzerini[(2)]

[(1)]FBK, Trento, Italy – [(2)]Sapienza Università di Roma

Logica e Informatica – A.A. 2021/22

Formal logical system

A formal logical system is constituted by several components, including:

- The language, with its
  - syntax (how to form sentences)
  - semantics (how to assign meaning to sentences)
  - pragmatics (how to use the logic for a specific objective)

  that enables forming the sentences that are acceptable in the formal system
- The inferencing apparatus, that enables studying all the relevant relationships between sentences of the language

Propositional logic - Intuition

The first formal logical system that we study is propositional logic.

- Propositional logic is the logic of propositions
- Intuitively, a proposition is anything that has a true value (i.e., can be true or false) in a state of the world (context) of interest.
- The same proposition can be expressed in different ways.
  E.g.
  - "B. Obama is drinking a beer"
  - " The U.S.A. president is drinking a beer", and
  - "B. Obama si sta facendo una birra"

  express the same proposition.
- Warning: obviously, not all sentences in natural language are propositions!
- The language of propositional logic allows us to use atomic propositions and to compose complex sentences (formulas) on the basis of atomic propositions and connectives.

Syntax of propositional logic language: the alphabet

To define the syntax of the propositional logic language, we define the alphabet (set of atomic symbols that can be used in the language), and then the set of sentences that can be formed using the alphabet.

Definition (Propositional alphabet)

Non logical symbols: A countable set $\mathcal{P}$ of symbols called propositional variables, or propositional atoms
Logical symbols (connectives): $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\equiv$
Separator symbols: "(" and ")"

The intuitive meaning of the connectives are as follows:

- $\neg$ is the negation symbol: $\neg A$ has the opposite truth value as A
- $\wedge$ is the conjunction symbol: $A \wedge B$ is true exactly when both A and B are true
- $\vee$ is the disjunction symbol: $A \vee B$ is true exactly when at least one between A and B is true
- $\rightarrow$ (sometimes written $\supset$) denotes the material implication symbol: $A \rightarrow B$ means that whenever A is true, B is true as well
- $\equiv$ is the material equivalence symbol: $A \equiv B$ is true if A and B have the same truth value.

The notion of inductive definition

In what follows, we will often define sets and functions inductively. In particular, since from a syntactic point of view a language is a set of sentences, we will often define the syntax of languages inductively.

An inductive definition of a set S specifies which are the elements of S by using

- one or more base steps, that directly specify that some elements belong to S
- one or more inductive steps, each of the form if $x \in S$, then $y \in S$, where y is related somehow with x.
- a closure condition, asserting that nothing else is in S (i.e., anything not derived from the base or inductive steps is not in S)

Whenever we indicate that a certain definition is inductive, its closure condition is implicit, and therefore we can avoid specifying such condition explicitly.

Example of inductive definition

Example

The set Nat of natural numbers is defined inductively as follows:

1. $0 \in Nat$
2. if $x \in Nat$ then $s(x) \in Nat$

Note that with the inductive definition of Nat, we can also

- define inductively any function F of the form $F : Nat \longrightarrow C$, using the following scheme:
  1. Base step We directly provide the definition of $F(0)$
  2. Inductive step Starting from $F(x)$, we provide the definition of $F(s(x))$
- prove that a certain property T holds for every natural numbers, using the following scheme:
  1. Base step We directly prove that $T(0)$ holds
  2. Inductive step Based on the assumption that $T(x)$ holds, we provide the proof that $T(s(x))$ holds as well

Syntax of propositional logic language: the formulas

Definition (Well formed formulas (or simply formulas))

A propositional logic formula on the alphabet $\mathcal{P}$ is a finite expression build according to the following inductive definition:

- every $P \in \mathcal{P}$ is an atomic formula, and every atomic formula is a formula
- if A and B are formulas, then the following are formulas:
  - (A)
  - $\neg A$
  - $A \wedge B$
  - $A \vee B$
  - $A \rightarrow B$ (also written as $A \supset B$)
  - $A \equiv B$

Formulas cont'd

Example (formulas and non formulas)

Formulas	Non formulas
$P \to Q$	PQ
$P \to (Q \to R)$	$(P \to \wedge((Q \to R)$
$P \wedge Q \to R$	$P \wedge Q \to \neg R\neg$

Reading formulas

Problem

How do we read the formula $P \wedge Q \rightarrow R$?
One might think that the formula $P \wedge Q \rightarrow R$ can be read in two ways:

1. $(P \wedge Q) \rightarrow R$
2. $P \wedge (Q \rightarrow R)$

Symbol priority

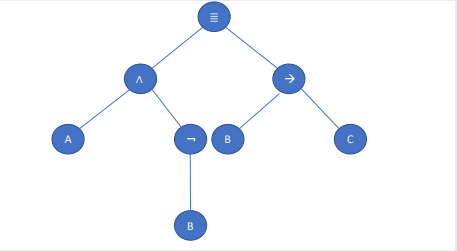
Priorities solve the problem: $\neg$ has higher priority, then $\wedge$, $\vee$, $\rightarrow$ and $\equiv$.
Parenthesis can be used around formulas to stress or change the priority.

Symbol	Priority
$\neg$	1
$\wedge$	2
$\vee$	3
$\rightarrow$	4
$\equiv$	5

Formulas as trees

Tree form of a formula

A formula can be seen as a tree. Leaf nodes are associated to propositional variables, while intermediate (non-leaf) nodes are associated to connectives. For instance the formula $(A \wedge \neg B) \equiv (B \rightarrow C)$ can be represented as the tree



Subformulas

Definition ((Proper) Subformula)

- A is a subformula of itself
- A is a subformula of $\neg A$, and A is a subformula of (A)
- A, B are subformulas of $A \wedge B$, $A \vee B$, $A \rightarrow B$, $A \equiv B$
- if A is a subformula of B and B is a subformula of C, then A is a subformula of C.
- A is a proper subformula of B if A is a subformula of B and A is different from B.

Remark

The subformulas of a formula represented as a tree correspond to all the different subtrees of the tree associated to the formula, one for each node.

Subformulas

Example

The subformulas of $(p \to (q \vee r)) \to (p \wedge \neg p)$ are

$$(p \to (q \vee r)) \to (p \wedge \neg p)$$
$$(p \to (q \vee r)), \quad (p \wedge \neg p), \quad p \to (q \vee r), \quad (q \vee r)$$
$$p \wedge \neg p, \quad q \vee r$$
$$p, \quad \neg p, \quad q, \quad r$$

Theorem

Every formula has a finite number of subformulas.

Note that the above is a meta-sentence, i.e., a sentence telling something about the formal logical system. In general, when we claim that a meta-sentence is a theorem, we should prove that such a meta-sentence is indeed true. When we do so, we use logic for proving properties of logic!

Propositional logic language: semantics

How can we specify the meaning of the sentences of propositional logic?

- The simplicity of the propositional language leads naturally to deciding that the meaning of any sentence coincides with its truth value, i.e., is either true or false.
- But, for example, a sentence like $P_1 \wedge P_2$ is either true or false depending of the truth value of P_1 and P_2. So, the meaning of a sentence depends on the truth value of the propositional variables in it!
- In turn, the truth value of a propositional variable depends on the specific intended meaning of the variable in the considered context. For example, if "piove" is a propositional variable, its truth value depends on the "world" (context, i.e., time and space) that we are considering.
- In order to capture the notion of world (or, context), propositional logic introduces the notion of interpretation.

Propositional logic language: semantics

The notion of interpretation is central to the semantics of propositional logic.

Definition (Propositional interpretation (or simply intepretation))

Given a propositional alphabet with propositional variables $\mathcal{P}$, a propositional interpretation is a function $\mathcal{I} : \mathcal{P} \to \{\text{True}, \text{False}\}$

Remark

A propositional interpretation $\mathcal{I}$ for a formula can be actually seen as the set of propositional variables such that $\mathcal{I}(P) = \text{True}$. Adopting this set theoretic representation, we will often represent an interpretation by such set.

Interpretation in propositional logic: example

Consider the following sentence over the alphabet $\{p, q, r\}$:

$$(p \wedge (\neg q \vee r)) \vee (p \wedge r)$$

Some possibile interpretations for the above sentence:

Example

	p	q	r	Set theoretic representation
$\mathcal{I}_1$	True	True	True	$\{p, q, r\}$
$\mathcal{I}_2$	True	False	True	$\{p, r\}$
$\mathcal{I}_3$	True	False	False	$\{p\}$
$\mathcal{I}_4$	False	False	False	$\{\}$

In the above picture, the table shoud be read as follows: $\mathcal{I}_1$ is the interpretation function such that $\mathcal{I}_1(p) =$ True, $\mathcal{I}_1(q) =$ True, $\mathcal{I}_1(r) =$ True, and so on.

Evaluation of a propositional formula in an interpretation

We now illustrate the rules for deciding whether the formula A is true in the intepretation $\mathcal{I}$ (or, equivalently, whether $\mathcal{I}$ satisfies A, written as $\mathcal{I} \models A$).

Definition ($\mathcal{I} \models A$)

$\mathcal{I} \models A$ is inductively defined as follows:

- If A is P, where $P \in \mathcal{P}$, then $\mathcal{I} \models A$ if and only if $\mathcal{I}(P) = \text{True}$
- $\mathcal{I} \models (A)$ if and only if $\mathcal{I} \models A$
- $\mathcal{I} \models \neg A$ if and only if it is not the case that $\mathcal{I} \models A$ (also written $\mathcal{I} \not\models A$)
- $\mathcal{I} \models A \wedge B$ if and only if both $\mathcal{I} \models A$ and $\mathcal{I} \models B$
- $\mathcal{I} \models A \vee B$ if and only if $\mathcal{I} \models A$ or $\mathcal{I} \models B$
- $\mathcal{I} \models A \rightarrow B$ if and only if it is not the case that $\mathcal{I} \models A$ and $\mathcal{I} \not\models B$
- $\mathcal{I} \models A \equiv B$ if and only if both $\mathcal{I} \models A \rightarrow B$ and $\mathcal{I} \models B \rightarrow A$

Evaluation of a propositional formula: example

Example

Let $\mathcal{I} = \{P\}$, and let us check if $\mathcal{I} \models (P \wedge Q) \vee (R \rightarrow S)$.

We replace each occurrence of each primitive propositions of the formula with the truth value assigned by $\mathcal{I}$, and apply the definition for connectives.

$$(\text{True} \wedge \text{False}) \vee (\text{False} \rightarrow \text{False}) \quad (1)$$

$$\text{False} \vee \text{True} \quad (2)$$

$$\text{True} \quad (3)$$

Observations on connectives

- $A \rightarrow B$ has the same meaning as $\neg A \vee B$
- $\neg(A \vee B)$ has the same meaning as $\neg A \wedge \neg B$ (De Morgan law 1)
- $\neg(A \wedge B)$ has the same meaning as $\neg A \vee \neg B$ (De Morgan law 2)
- $A \rightarrow B$ has the same meaning as $\neg B \rightarrow \neg A$
- $A \rightarrow \neg B$ has the same meaning as $B \rightarrow \neg A$
- A is a sufficient condition for B can be written as $A \rightarrow B$, and can read as "if A then B"
- A is a necessary condition for B can be written as $B \rightarrow A$
- A is a necessary and sufficient for B can be written as $A \equiv B$, and can be read as "A if and only if B"
- NAND is another connective that is defined as A NAND B is defined as $\neg(A \wedge B)$
- NOR is another connective that is defined as A NOR B is defined as $\neg(A \vee B)$

Exercise

Prove the following proposition (i.e., prove that the following meta-sentence is true).

Proposition

Let $\mathcal{I}$ and $\mathcal{I}'$ be two interpretations for a formula A. If for every propositional variable P appearing in A it holds that $\mathcal{I}(P) = \mathcal{I}'(P)$, then $\mathcal{I} \models A$ if and only if $\mathcal{I}' \models A$.

Algorithm for checking if $\mathcal{I} \models A$

Lazy evaluation algorithm (1/2)

(A is p)

check($\mathcal{I} \models p$):
  if $\mathcal{I}(p) = true$
    then return YES
  else return NO

(A is $B \wedge C$)

check($\mathcal{I} \models B \wedge C$):
  if check($\mathcal{I} \models B$)
    then return check($\mathcal{I} \models C$)
  else return NO

(A is $B \vee C$)

check($\mathcal{I} \models B \vee C$):
  if check($\mathcal{I} \models B$)
    then return YES
  else return check($\mathcal{I} \models C$)

Checking if $\mathcal{I} \models A$

Lazy evaluation algorithm (2/2)

(A is $B \rightarrow C$)	check($\mathcal{I} \models B \rightarrow C$): if check($\mathcal{I} \models B$) then return check($\mathcal{I} \models C$) else return YES
(A is $B \equiv C$)	check($\mathcal{I} \models B \equiv C$): if check($\mathcal{I} \models B$) then return check($\mathcal{I} \models C$) else return not(check($\mathcal{I} \models C$)

Question: which is the computational complexity of the algorithm?

Formalizing natural language sentences

Exercise

Let's consider a propositional language where p means *"Paola is happy"*, q means *"Paola paints a picture"*, and r means *"Renzo is happy"*. Formalize the following sentences:

1. *"if Paola is happy and paints a picture then Renzo isn't happy"*
2. *"if Paola is happy, then she paints a picture"*
3. *"Paola is happy only if she paints a picture"*

The precision of formal languages avoids the ambiguities of natural languages.

Formalizing natural language sentences

Exercise

Let's consider a propositional language where p means *"Paola is happy"*, q means *"Paola paints a picture"*, and r means *"Renzo is happy"*. Formalize the following sentences:

1. *"if Paola is happy and paints a picture then Renzo isn't happy"*
   $p \wedge q \rightarrow \neg r$
2. *"if Paola is happy, then she paints a picture"*
3. *"Paola is happy only if she paints a picture"*

The precision of formal languages avoid the ambiguities of natural languages.

Formalizing natural language sentences

Exercise

Let's consider a propositional language where p means *"Paola is happy"*, q means *"Paola paints a picture"*, and r means *"Renzo is happy"*. Formalize the following sentences:

1. *"if Paola is happy and paints a picture then Renzo isn't happy"*
   $p \wedge q \rightarrow \neg r$
2. *"if Paola is happy, then she paints a picture"*
   $p \rightarrow q$
3. *"Paola is happy only if she paints a picture"*

The precision of formal languages avoid the ambiguities of natural languages.

Formalizing natural language sentences

Exercise

Let's consider a propositional language where p means *"Paola is happy"*, q means *"Paola paints a picture"*, and r means *"Renzo is happy"*. Formalize the following sentences:

1. *"if Paola is happy and paints a picture then Renzo isn't happy"*
   $p \wedge q \rightarrow \neg r$
2. *"if Paola is happy, then she paints a picture"*
   $p \rightarrow q$
3. *"Paola is happy only if she paints a picture"*
   $\neg(p \wedge \neg q)$ which is equivalent to $p \rightarrow q$

The precision of formal languages avoids the ambiguities of natural languages.

Formalizing natural language sentences

Exercise

Let's consider a propositional language where r means *"rigore nei conti pubblici"*, e means *"sviluppo dell'economia"*, s means *"stagnazione"* and f means *"spesa facile"*. Formalize the following sentences:

1. *"il rigore nei conti pubblici contraddice lo sviluppo economico"*
2. *"se c'è stagnazione, allora non c'è sviluppo economico"*
3. *"se c'è sviluppo economico, allora non c'è stagnazione"*
4. *"la spesa facile è il contrario del rigore nei conti pubblici"*

Formalizing natural language sentences

Exercise

Let's consider a propositional language where r means *"rigore nei conti pubblici"*, e means *"sviluppo dell'economia"*, s means *"stagnazione"* and f means *"spesa facile"*. Formalize the following sentences:

1. *"il rigore nei conti pubblici contraddice lo sviluppo economico"*
   $r \rightarrow \neg e$
2. *"se c'è stagnazione, allora non c'è sviluppo economico"*
3. *"se c'è sviluppo economico, allora non c'è stagnazione"*
4. *"la spesa facile è il contrario del rigore nei conti pubblici"*

Formalizing natural language sentences

Exercise

Let's consider a propositional language where r means *"rigore nei conti pubblici"*, e means *"sviluppo dell'economia"*, s means *"stagnazione"* and f means *"spesa facile"*. Formalize the following sentences:

1. *"il rigore nei conti pubblici contraddice lo sviluppo economico"*
   $r \rightarrow \neg e$
2. *"se c'è stagnazione, allora non c'è sviluppo economico"*
   $s \rightarrow \neg e$
3. *"se c'è sviluppo economico, allora non c'è stagnazione"*
4. *"la spesa facile è il contrario del rigore nei conti pubblici"*

Formalizing natural language sentences

Exercise

Let's consider a propositional language where r means *"rigore nei conti pubblici"*, e means *"sviluppo dell'economia"*, s means *"stagnazione"* and f means *"spesa facile"*. Formalize the following sentences:

1. *"il rigore nei conti pubblici contraddice lo sviluppo economico"*
   $r \rightarrow \neg e$
2. *"se c'è stagnazione, allora non c'è sviluppo economico"*
   $s \rightarrow \neg e$
3. *"se c'è sviluppo economico, allora non c'è stagnazione"*
   $e \rightarrow \neg s$
4. *"la spesa facile è il contrario del rigore nei conti pubblici"*

Formalizing natural language sentences

Exercise

Let's consider a propositional language where r means *"rigore nei conti pubblici"*, e means *"sviluppo dell'economia"*, s means *"stagnazione"* and f means *"spesa facile"*. Formalize the following sentences:

1. *"il rigore nei conti pubblici contraddice lo sviluppo economico"*
   $r \rightarrow \neg e$
2. *"se c'è stagnazione, allora non c'è sviluppo economico"*
   $s \rightarrow \neg e$
3. *"se c'è sviluppo economico, allora non c'è stagnazione"*
   $e \rightarrow \neg s$
4. *"la spesa facile è il contrario del rigore nei conti pubblici"*
   $f \equiv \neg r$

Evaluation of a formula as a function

The algorithm for checking whether a formula A is satisfied by an interpretation $\mathcal{I}$ makes it clear the relationship between an interpretation and a formula, and shows that an interpretation can be extended to a function from the set of formulas to the set $\{\text{True}, \text{False}\}$:

- $\mathcal{I}(A) = \text{True}$ if $\mathcal{I} \models A$
- $\mathcal{I}(A) = \text{False}$ if $\mathcal{I} \not\models A$

For example, it is easy to see that, if $\mathcal{I}(p) = \text{True}$ and $\mathcal{I}(q) = \text{False}$, then $\mathcal{I}(p \vee q) = \text{True}$.
Note that it is common to regard the truth values as numbers, usually with the following convention:

- True corresponds to 1
- False corresponds to 0

Thus, an intepretation can also be seen as a function from the set of formulas to the set $\{0, 1\}$. This is the basis for the development of Boolean algebra.

Truth tables

The meaning of the connectives we have considered in propositional logic ($\neg$, $\vee$, $\wedge$, $\rightarrow$, $\equiv$) can be represented as a table, called truth table. For example:

Example: truth table for the $\wedge$ binary connective ($n = 2$)

A	B	$A \wedge B$
False	False	False
False	True	False
True	False	False
True	True	True

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

In principles, we could think of other connectives whose semantics is specified through truth tables: if a connective is n-ary (i.e., it applies to n operands), the truth table has as many rows as the possibile vectors of n values, each value being True or False, and $n + 1$ columns, one for each operand plus one for representing the truth value that the formula has if the truth values of the various components are the ones specified by the vector fixing the truth value of the operands.

Boolean functions

More generally, any Boolean function can be specified through a truth table, where a Boolean function is a function of type

$$\{\text{False}, \text{True}\}^n \to \{\text{False}, \text{True}\}$$

or $\{0,1\}^n \to \{0,1\}$, and the corresponding truth table has again as many rows as the possible vectors of n values, and $n+1$ columns.

Example of Boolean function f ($n = 3$)

A	B	C	$f(A,B,C)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Boolean functions and connectives

It follows that for any connective, there is a corresponding Boolean function capturing its semantics, and viceversa.

- Boolean functions with 0 argument:
  - $f_T :\rightarrow 1$ corresponds to the new connective T (for every $\mathcal{I}$, $I(\mathsf{T}) = \text{True}$)
  - $f_F :\rightarrow 0$ corresponds to the new connective F (for every $\mathcal{I}$, $I(\mathsf{F}) = \text{False}$)
- Boolean functions with 1 argument:

A	f_{id}
0	0
1	1

A	$f_{\neg}$
0	1
1	0

A	$f_{1,1}$
0	1
1	1

A	$f_{0,1}$
0	0
1	0

Note that $f_{\neg}$ correspond to the connective $\neg$. The other 1-argument functions do not correspond to interesting connectives.

Semantics of unary connectives expressed through Boolean functions

If $\circ$ is a 1-argument connective, and $f_{\circ}$ is the corresponding Boolean function, we have that, for every interpretation $\mathcal{I}$, and every formula A,

$$\mathcal{I}(\circ A) = f_{\circ}(\mathcal{I}(A)).$$

Boolean functions and connectives

Obviously, for every binary connective, there is a corresponding Boolean function, and viceversa.

- Boolean functions with 2 arguments corresponding to our connectives:

A	B	$f_\wedge$
0	0	0
0	1	0
1	0	0
1	1	1

A	B	$f_\vee$
0	0	0
0	1	1
1	0	1
1	1	1

A	B	$f_\rightarrow$
0	0	1
0	1	1
1	0	0
1	1	1

A	B	$f_\equiv$
0	0	1
0	1	0
1	0	0
1	1	1

Semantics of binary connectives expressed through Boolean functions

If $\circ$ is a 2-argument connective, and $f_\circ$ is the corresponding Boolean function, then for every interpretation $\mathcal{I}$, and every formulas A, B,

$$\mathcal{I}(A \circ B) = f_\circ(\mathcal{I}(A), \mathcal{I}(B)).$$

How many Boolean functions?

How many Boolean functions (or connectives) with n arguments we have?

- $n = 0 \longrightarrow 2$

f_T
1

f_F
0

- $n = 1 \longrightarrow 4$

A	f_{id}
0	0
1	1

A	$f_{\neg}$
0	1
1	0

A	$f_{1,1}$
0	1
1	1

A	$f_{0,1}$
0	0
1	0

How many Boolean functions?

How many Boolean functions (or connectives) with n arguments we have? In general, all the truth tables of the various Boolean functions with the same number n of arguments have a number of rows equal to the number of interpretations that we can define on n propositional variables, i.e., 2^n. Two such truth tables differ only in the last columns. Now, since there are 2^{2^n} possible columns of values 0,1 with 2^n rows, it follows that there are 2^{2^n} possible Boolean functions with n arguments.

- $n = 2 \longrightarrow 16$
  the number of tables with all possible combinations of $\{0, 1\}$ in 2 columns, i.e., number of tables with $2^2 = 4$ rows, and $2 + 1$ columns, i.e., the number of $\{0, 1\}$ vectors of 4 elements, i.e., $2^4 = 16$.
- n generic $\longrightarrow 2^{2^n}$
  the number of tables with all possible combinations of $\{0, 1\}$ in n columns, i.e., number of tables with 2^n rows, and $n + 1$ columns, i.e., the number of $\{0, 1\}$ vectors of 2^n elements, i.e., 2^{2^n}.

Truth tables for formulas

It is immediate to build a truth table for a formula: if the formula has n propositional variables, the table has one row for each interpretation of the formula (assigning truth values to all the propositional variables), and $n+1$ columns, where for a row corresponding to an interpretation $\mathcal{I}$, the last column will contain $\mathcal{I}(A)$.

Example

Truth table for $(F \vee G) \wedge \neg(F \wedge G)$.

F	G	$(F \vee G) \wedge \neg(F \wedge G)$
True	True	False
True	False	True
False	True	True
False	False	False

Note: intuitively, what does the formula in the example represent?

Truth tables for formulas: exercises

Suggestion: to simplify the process of building a truth table for a formula, we can add columns to the table corresponding to the various subformulas

Example: Truth table for $(F \vee G) \wedge \neg(F \wedge G)$

F	G	$F \vee G$	$F \wedge G$	$\neg(F \wedge G)$	$(F \vee G) \wedge \neg(F \wedge G)$
True	True	True	True	False	False
True	False	True	False	True	True
False	True	True	False	True	True
False	False	False	False	True	False

Exercise: build the truth tables for the following propositional formulas:

- $(p \to p) \to p$
- $p \to (p \to p)$
- $p \vee q \to p \wedge q$
- $p \vee (q \wedge r) \to (p \wedge r) \vee q$
- $p \to (q \to p)$
- $(p \wedge \neg q) \vee \neg(p \leftrightarrow q)$

From formulas to Boolean functions

It can be shown that there is a total correspondence between Boolean functions and formulas. One direction is easy to prove.

Proposition

For every formula A with n propositional variables $p_1, \ldots, p_n$, one can define a Boolean function f_A with n arguments such that for every combination of truth values for $p_1, \ldots, p_n$ corresponding to the interpretation $\mathcal{I}$, $f(\mathcal{I}(p_1), \ldots, \mathcal{I}(p_n)) = \mathcal{I}(A_f)$.

The proof of the above proposition is based on the fact that the truth table for a formula directly specifies the corresponding Boolean formula.

From Boolean functions to formulas

It can be shown that there is a total correspondence between Boolean functions and formulas. The other direction is represented by the following proposition.

Proposition

For every Boolean function f with n arguments, there is a corresponding formula A_f with n propositional variables $p_1, \ldots, p_n$, such that for every interpretation $\mathcal{I}$, $f(\mathcal{I}(p_1), \ldots, \mathcal{I}(p_n)) = \mathcal{I}(A_f)$.

The above proposition is more difficult to prove.

From Boolean functions to formulas: first formulation

Given the truth table of a Boolean function ϕ with n arguments ($n > 0$), we can build the formula with $n+1$ propositional variables $p_0, p_1, \ldots, p_n$ of the form

$$(p_0 \wedge \neg p_0) \vee \alpha_1 \vee \cdots \vee \alpha_k$$

with one α_i for each row with the value 1 for ϕ, where each α_i has the form

$$\gamma_1 \wedge \cdots \wedge \gamma_n$$

where γ_j is p_j if the row has 1 in column j, and is $\neg p_j$ if the row has the value 0 in column j. Note that $(p_0 \wedge \neg p_0)$ takes care of the case where all rows for ϕ have no 1.

Example: formula corresponding to the Boolean function ϕ

p_1	p_2	p_3	$\phi(p_1, p_2, p_3)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

$$\Longrightarrow \quad (p_0 \wedge \neg p_0) \vee (\neg p_1 \wedge \neg p_2 \wedge \neg p_3) \vee (\neg p_1 \wedge p_2 \wedge p_3)$$

From Boolean functions to formulas: second formulation

Given the truth table of a Boolean function ψ with n arguments ($n > 0$), we can build the formula with $n+1$ propositional variables $p_0, p_1, \ldots, p_n$ of the form

$$\neg((p_0 \wedge \neg p_0) \vee \beta_1 \vee \cdots \vee \beta_k)$$

with one β_i for each row with the value 0 for ψ, where each β_i has the form

$$\gamma_1 \wedge \cdots \wedge \gamma_n$$

where γ_j is p_j if the row has 1 in column j, and is $\neg p_j$ if the row has the value 0 in column j. Note that $(p_0 \wedge \neg p_0)$ takes care of the case where all rows for ϕ have no 0.

Example: formula corresponding to the Boolean function ψ

p_1	p_2	p_3	$\psi(p_1, p_2, p_3)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$\Longrightarrow$

$$\neg((p_0 \wedge \neg p_0) \vee (\neg p_1 \wedge \neg p_2 \wedge p_3) \vee (p_1 \wedge \neg p_2 \wedge \neg p_3))$$

by De Morgan, equivalent to

$$(\neg p_0 \vee p_0) \wedge (p_1 \vee p_2 \vee \neg p_3) \wedge (\neg p1 \vee p_2 \vee p_3)$$

Complete sets of connectives

The two formulations above are at the basis of the proof that every Boolean function can be represented by a propositional formula. Such proof shows that the following proposition is true.

Proposition

Every formula of propositional logic can be written using only the connectives $\neg$, $\vee$ *and* $\wedge$.

In other words, the set of connectives $\{\neg, \vee, \wedge\}$ is complete, because they allow to express any formula that corresponds to a Boolean function. Note that this implies that the set of our connectives, i.e., $\{\neg, \vee, \wedge, \rightarrow, \equiv\}$ is also complete.

Even more specifically, it can be shown that at least two normal forms exist for propositional formulas.

Normal form for formulas

Proposition

Every Boolean function can be expressed by a formula of propositional logic in the so-called disjunctive normal form, i.e., as

$$\alpha_1 \vee \cdots \vee \alpha_k$$

where each α_i is a minterm, i.e., has the form

$$\gamma_1 \wedge \cdots \wedge \gamma_n$$

where each γ_j is a literal, i.e., either a propositional variable or the negation of a propositional variable.

Proposition

Every Boolean function can be expressed by a formula of propositional logic in the so-called conjunctive normal form, i.e., as

$$\beta_1 \wedge \cdots \wedge \beta_k$$

where each β_i is a clause, i.e., has the form

$$\gamma_1 \vee \cdots \vee \gamma_n$$

where each γ_j is a literal.

Are all our connectives necessary?

We leave as an exercise to prove the following propositions, regarding other complete sets of connectives.

Proposition

Every formula of propositional logic can be written using only the connectives $\neg$ *and* $\vee$.

Proposition

Every formulas of propositional logic can be written using only the connectives $\neg$ *and* $\wedge$.

Also, if we assume we can use also T and F in formulas, the following proposition holds.

Proposition

Every formulas of propositional logic can be written using only the connective NAND *or only the connective* NOR.

Valid, satisfiable, and unsatisfiable formulas

Definition (Model of a formula)

An interpretation $\mathcal{I}$ that satisfies a formula A, i.e., such that $\mathcal{I} \models A$, is called a model of A.

Definition

A formula A is

Valid (or a tautology) if for every interpretation $\mathcal{I}$, $\mathcal{I} \models A$
i.e., every interpretation for A is a model of A

Satisfiable if there is an interpretation $\mathcal{I}$ s.t. $\mathcal{I} \models A$
i.e., A has at least one interpretation that is a model, i.e.. A has at least one model

Unsatisfiable if for no interpretation $\mathcal{I}$, $\mathcal{I} \models A$
i.e., A has no interpretation that is a model, i.e., A does not have any model

Valid, satisfiable, and unsatisfiable formulas

Proposition

A valid $\longrightarrow$ *A satisfiable* $\longleftrightarrow$ *A not unsatisfiable*
A unsatisfiable $\longleftrightarrow$ *A not satisfiable* $\longrightarrow$ *A not Valid*

Proposition

if A is	*then* $\neg A$ *is*
Valid	*Unsatisfiable*
Satisfiable	*not Valid*
not Valid	*Satisfiable*
Unsatisfiable	*Valid*

Cheking validity and (un)satisfiability of a formula

Truth table

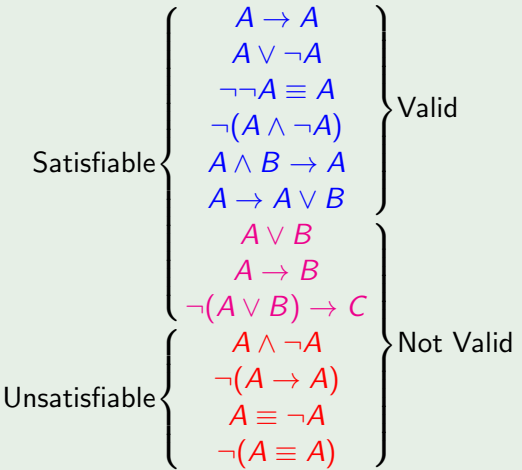
Checking (un)satisfiability and validity of a formula A can be done by enumerating all the interpretations which are relevant for S, and for each interpretation $\mathcal{I}$ checking if $\mathcal{I} \models A$.

Example (using truth table)

A	B	C	$A \rightarrow (B \vee \neg C)$
True	True	True	True
True	True	False	True
True	False	True	False
True	False	False	True
False	True	True	True
False	True	False	True
False	False	True	True
False	False	False	True

Valid, satisfiable, and unsatisfiable formulas

Example



Prove that the blue formulas are valid, that the magenta formulas are satisfiable but not valid, and that the red formulas are unsatisfiable.

Valid, satisfiable, and unsatisfiable sets of formulas

Definition (Model of a set of formula)

An interpretation $\mathcal{I}$ that satisfies all the formula of a set Γ, i.e., such that $\mathcal{I} \models A$ for all formulas $A \in \Gamma$, is called a model of Γ (written $\mathcal{I} \models \Gamma$).

Definition

A set of formulas Γ is

Valid if for all $\mathcal{I}$, $\mathcal{I} \models \Gamma$, i.e., every interpretation is a model of Γ

Satisfiable if there is an $\mathcal{I}$ s.t. $\mathcal{I} \models \Gamma$, i.e., Γ has at least one model

Unsatisfiable if for no $\mathcal{I}$, $\mathcal{I} \models \Gamma$, i.e., Γ has no model

Proposition

For any finite set of formulas Γ, (i.e., $\Gamma = \{A_1, \ldots, A_n\}$ for some $n \geq 1$), Γ is valid (resp., satisfiable and unsatisfiable) if and only if $A_1 \wedge \cdots \wedge A_n$ is valid (resp., satisfiable and unsatisfiable).

Validity check by truth tables: Example

We use the truth tables method to determine whether $(p \to q) \vee (p \to \neg q)$ is valid.

p	q	$p \to q$	$\neg q$	$p \to \neg q$	$(p \to q) \vee (p \to \neg q)$
True	True	True	False	False	**True**
True	False	False	True	True	**True**
False	True	True	False	True	**True**
False	False	True	True	True	**True**

The formula is valid since it is satisfied by every interpretation.

Satisfiablity check by truth tables: Example

We use the truth tables method to determine whether $(\neg p \vee q) \wedge (q \rightarrow \neg r \wedge \neg p) \wedge (p \vee r)$ (denoted with ϕ) is satisfiable.

p	q	r	$\neg p \vee q$	$\neg r \wedge \neg p$	$q \rightarrow \neg r \wedge \neg p$	$(p \vee r)$	ϕ
True	True	True	True	False	False	True	False
True	True	False	True	False	False	True	False
True	False	True	False	False	True	True	False
True	False	False	False	False	True	True	False
False	True	True	True	False	False	True	False
False	True	False	True	True	True	False	False
False	False	True	True	False	True	True	**True**
False	False	False	True	True	True	False	False

There exists an interpretation satisfying ϕ, thus ϕ is satisfiable.

Truth tables: Exercises

Use the truth table method to verify whether the following formulas are valid, satisfiable or unsatisfiable:

- $(p \to q) \land \neg q \to \neg p$
- $(p \to q) \to (p \to \neg q)$
- $(p \lor q \to r) \lor p \lor q$
- $(p \lor q) \land (p \to r \land q) \land (q \to \neg r \land p)$
- $(p \to (q \to r)) \to ((p \to q) \to (p \to r))$
- $(p \lor q) \land (\neg q \land \neg p)$
- $(\neg p \to q) \lor ((p \land \neg r) \leftrightarrow q)$
- $(p \to q) \land (p \to \neg q)$
- $(p \to (q \lor r)) \lor (r \to \neg p)$

Logical consequence (or logical implication)

Definition (Logical consequence or logically implication)

A formula A is a logical consequence of (or, is logically implied by) a set of formulas Γ, denoted by $\Gamma \models A$ if every model of Γ is also a model of A, i.e.,

$$\text{for all } \mathcal{I},\ \mathcal{I} \models \Gamma \text{ implies } \mathcal{I} \models A.$$

We write $\models A$ to mean $\emptyset \models A$, i.e., to mean that A is valid. We write $\Gamma \not\models A$ to mean that A is not logically implied by Γ. Thus, $\not\models A$ means A not valid.

Observation

$\Gamma \not\models A$ means that there exists at least one model of Γ where A is false.

If Γ_1 and Γ_2 are sets of formulas, we also define $\Gamma_1 \models \Gamma_2$ simply as $\Gamma_1 \models A$ for every $A \in \Gamma_2$. Thus, $\emptyset \models \Gamma$, $\not\models \Gamma$, and $\Gamma_1 \not\models \Gamma_2$ are defined accordingly.

Overloading of $\models$

Note that the symbol $\models$ is overloaded:

- when $\models$ has an intepretation $\mathcal{I}$ on the left, then it means satisfaction, i.e., $\mathcal{I} \models \alpha$ means that $\mathcal{I}$ satisfies α;
- when $\models$ has a formula or a set of formulas Γ on the left, then it means logical implication, i.e., $\Gamma \models \alpha$ means that Γ logically implies α.

Logical equivalence

Definition (Logical equivalence)

Two formulas F and G are logically equivalent (denoted with $F \models\models G$) if both $F \models G$ and $G \models F$, i.e., for every interpretation $\mathcal{I}$, $\mathcal{I}(F) = \mathcal{I}(G)$ (in other words, if the set of models of F and G coincide).

Logical equivalence is naturally extended to sets of formulas: $\Gamma_1 \models\models \Gamma_2$ if both $\Gamma_1 \models \Gamma_2$ and $\Gamma_2 \models \Gamma_2$.

Sometimes, logical equivalence is denoted by the same symbol used for "material equivalence", i.e., $\equiv$. However, it is important not to confuse the two notions:

- material implication is in the language of propositonal logic
- logical implication is in the meta-language, i.e., it is outside the language of propositonal logic.

Exercises on logical implication

Check that the following logical implications hold

- $p \models p \vee q$
- $q \vee p \models p \vee q$
- $\{p \vee q, p \rightarrow r, q \rightarrow r\} \models r$
- $\{p \rightarrow q, p\} \models q$

Hint: base your check on the definition of "satisfaction", i.e., check that every model of the formulas on the left is also a model of the formula on the right.

Solution: checking logical consequence in a direct manner

Verify that $p \models p \vee q$ Suppose that $\mathcal{I} \models p$, then by definition $\mathcal{I} \models p \vee q$.

Verify that $q \vee p \models p \vee q$ Suppose that $\mathcal{I} \models q \vee p$, then either $\mathcal{I} \models q$ or $\mathcal{I} \models p$. In both cases we have that $\mathcal{I} \models p \vee q$.

Verify that $\{p \vee q, p \rightarrow r, q \rightarrow r\} \models r$ Suppose that $\mathcal{I} \models p \vee q$ and $\mathcal{I} \models p \rightarrow r$ and $\mathcal{I} \models q \rightarrow r$. Then either $\mathcal{I} \models p$ or $\mathcal{I} \models q$. In the first case, since $\mathcal{I} \models p \rightarrow r$, then $\mathcal{I} \models r$, In the second case, since $\mathcal{I} \models q \rightarrow r$, then $\mathcal{I} \models r$.

Verify that $\{p \rightarrow q, p\} \models q$ Suppose that $\mathcal{I} \models p \rightarrow q$ and $\mathcal{I} \models p$. By the semantics of $\rightarrow$, $\mathcal{I} \models p \rightarrow q$ means that $\mathcal{I} \models \neg p \vee q$. Therefore, $\mathcal{I} \models q$.

Logical consequence and equivalence using truth tables

Use the truth tables method to determine whether $\neg p \models p \wedge \neg q \rightarrow p \wedge q$.

p	q	$\neg p$	$p \wedge \neg q$	$p \wedge q$	$p \wedge \neg q \rightarrow p \wedge q$
True	True	False	False	True	True
True	False	False	True	False	False
False	True	**True**	False	False	**True**
False	False	**True**	False	False	**True**

Use the truth tables method to determine whether $p \rightarrow (q \wedge \neg q) \models\!\!\!| \neg p$.

p	q	$q \wedge \neg q$	$p \rightarrow (q \wedge \neg q)$	$\neg p$
True	True	False	**False**	**False**
True	False	False	**False**	**False**
False	True	False	**True**	**True**
False	False	False	**True**	**True**

Truth tables: exercises

Use the truth table method to determine whether the following logical consequences and logical equivalences hold:

- $(p \to q) \models \neg p \to \neg q$
- $(p \to q) \wedge \neg q \models \neg p$
- $p \to q \wedge r \models (p \to q) \to r$
- $p \vee (\neg q \wedge r) \models q \vee \neg r \to p$
- $\neg(p \wedge q) \models\models \neg p \vee \neg q$
- $(p \vee q) \wedge (\neg p \to \neg q) \models\models q$
- $(p \wedge q) \vee r \models\models (p \to \neg q) \to r$
- $(p \vee q) \wedge (\neg p \to \neg q) \models\models p$
- $((p \to q) \to q) \to q \models\models p \to q$

Properties of propositional logical consequence

Proposition

If Γ and Σ are two sets of propositional formulas, and A and B are two formulas, then the following properties hold:

Reflexivity *If $A \in \Gamma$, then $\Gamma \models A$*

Ex falso sequitur quodlibet *If Γ is unsatisfiable, then $\Gamma \models A$ for all A*

Monotonicity *If $\Gamma \models A$ then $\Gamma \cup \Sigma \models A$*

Cut *If $\Gamma \models A$ and $\Sigma \cup \{A\} \models B$ then $\Gamma \cup \Sigma \models B$*

Compactness *If $\Gamma \models A$, then there is a finite subset $\Gamma_0 \subseteq \Gamma$, such that $\Gamma_0 \models A$*

Deduction theorem *If $\Gamma \cup \{A\} \models B$ then $\Gamma \models A \rightarrow B$*

Deduction principle *$\Gamma \cup \{A\} \models B$ if and only if $\Gamma \models A \rightarrow B$*

Refutation principle *$\Gamma \models A$ if and only if $\Gamma \cup \{\neg A\}$ is unsatisfiable*

Properties of propositional logical consequence

Reflexivity If $A \in \Gamma$, then $\Gamma \models A$.
PROOF: If $A \in \Gamma$, and $\mathcal{I} \models \Gamma$, then obviously $\mathcal{I} \models A$.

Ex falso sequitur quodlibet If Γ is unsatisfiable, then $\Gamma \models A$ for all A
PROOF: Recall that $\Gamma \not\models A$ means that there is a model of Γ where A is false. But if Γ is unsatisfiable there is no model of Γ, and therefore it cannot be that $\Gamma \not\models A$. Thus we conclude that $\Gamma \models A$.

Monotonicity If $\Gamma \models A$ then $\Gamma \cup \Sigma \models A$
PROOF: Every model of $\Gamma \cup \Sigma$ is clearly a model Γ. Therefore, If $\Gamma \models A$, then every model of $\Gamma \cup \Sigma$ is also a model of A, i.e., $\Gamma \cup \Sigma \models A$.

Cut If $\Gamma \models A$ and $\Sigma \cup \{A\} \models B$ then $\Sigma \cup \Gamma \models B$.
PROOF: For all $\mathcal{I}$, if $\mathcal{I} \models \Gamma \cup \Sigma$, then $\mathcal{I} \models \Gamma$ and $\mathcal{I} \models \Sigma$. The hypothesis $\Gamma \models A$ implies that $\mathcal{I} \models A$. Since $\mathcal{I} \models \Sigma$, then $\mathcal{I} \models \Sigma \cup \{A\}$. The hypothesis $\Sigma \cup \{A\} \models B$, implies that $\mathcal{I} \models B$. We can therefore conclude that $\Sigma \cup \Gamma \models B$.

Properties of propositional logical consequence

Compactness If $\Gamma \models A$, then there is a finite subset $\Gamma_0 \subseteq \Gamma$, such that $\Gamma_0 \models A$.

PROOF: Let $\mathcal{P}_A$ be the propositional variables occurring in A. Let $\mathcal{I}_1, \ldots, \mathcal{I}_n$ (with $n \leq 2^{|\mathcal{P}_A|}$), be all the interpretations of the language $\mathcal{P}_A$ that do not satisfy A. Since $\Gamma \models A$, then there should be $\mathcal{I}'_1, \ldots, \mathcal{I}'_n$ interpretations of the language of Γ, which are extensions of $\mathcal{I}_1, \ldots, \mathcal{I}_n$, and such that $\mathcal{I}'_k \not\models \gamma_k$ for some $\gamma_k \in \Gamma$. Let $\Gamma_0 = \{\gamma_1, \ldots, \gamma_k\}$. Then $\Gamma_0 \models A$. Indeed if $\mathcal{I} \models \Gamma_0$ then $\mathcal{I}$ is an extension of an interpretation J of $\mathcal{P}_A$ that satisfies A, and therefore $\mathcal{I} \models A$.

Deduction theorem If $\Gamma \cup \{A\} \models B$ then $\Gamma \models A \rightarrow B$

PROOF: Assume $\Gamma \cup \{A\} \models B$. Consider an interpretation $\mathcal{I}$, and suppose that $\mathcal{I} \models \Gamma$. If $\mathcal{I} \not\models A$, then $\mathcal{I} \models A \rightarrow B$. If instead $\mathcal{I} \models A$, then $\mathcal{I} \models \Gamma \vee \{A\}$. But since $\Gamma \cup \{A\} \models B$, we conclude that $\mathcal{I} \models B$. We can therefore conclude that $\Gamma \models A \rightarrow B$.

Properties of propositional logical consequence

Deduction principle $\Gamma \cup \{A\} \models B$ if and only if $\Gamma \models A \rightarrow B$

PROOF: One direction coincides with the deduction theorem. The other direction amount to prove that if $\Gamma \models A \rightarrow B$, then $\Gamma \cup \{A\} \models B$. But $\Gamma \models A \rightarrow B$ means that every model of Γ is a model of $\neg A \vee B$. This means that every model of Γ where A is true must have B true, and thus every model of $\Gamma \vee A$ is a model of B, which means $\Gamma \cup \{A\} \models B$.

Refutation principle $\Gamma \models A$ if and only if $\Gamma \cup \{\neg A\}$ is unsatisfiable

PROOF:

($\Longrightarrow$) Suppose by contradiction that $\Gamma \cup \{\neg A\}$ is satisfiable. This implies that there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \Gamma$ and $\mathcal{I} \models \neg A$, i.e., $\mathcal{I} \not\models A$. This contradicts that fact that for all interpretations that satisfies Γ, they satisfy A

($\Longleftarrow$) Let $\mathcal{I} \models \Gamma$, then by the fact that $\Gamma \cup \{\neg A\}$ is unsatisfiable, we have that $\mathcal{I} \not\models \neg A$, and therefore $\mathcal{I} \models A$. We can conclude that $\Gamma \models A$.

Logic and questions

- When we pose a dichotomic question Q and we implicitely refer to a specific intepretation $\mathcal{I}$, the formalization in logic corresponds to checking whether $\mathcal{I} \models Q$. It follows that we are dealing with evaluating a formula (representing a Boolean questions) in an interpretation, and the possible answers are:
  - yes, if $\mathcal{I} \models Q$
  - no, if $\mathcal{I} \not\models Q$

  Example: "Did Juventus win the Champions League last year"? no
- When we pose a dichotomic question Q and we refer to a set of more than one interpretations, denoted by a set of formulas Γ, the formalization in logic corresponds to checking whether $\Gamma \models Q$. It follows that we are dealing with logical implication, and the possible answers are:
  - yes, if $\Gamma \models Q$
  - no, if $\Gamma \models \neg Q$
  - don't know, if $\Gamma \not\models Q$ and $\Gamma \not\models \neg Q$

  Example: "Will A.S. Roma win the Champions League next year"? don't know

Logical implication and formal reasoning

We discuss two basic forms of reasoning based on semantics, i.e., based on logical implication.

- Deductive reasoning (deduction) via logical implication
  Deductive reasoning allows us to conclude A on the basis of a set of formuals Γ, by checking whether a formula A can be deduced from a set of formulas Γ, i.e., by checking whether
  $$\Gamma \models A$$
- Abductive reasoning (abduction) via logical implication
  Abductive reasoning allows us to conclude that a formula can be used to deduce another formula A in the context of a background knowledge represented by a set of formulas Γ (that does not logically implies A), by checking whether there esists a formula B such that:
  - $\Gamma \cup \{B\}$ is satisfiable, and
  - $\Gamma \cup \{B\} \models A$

  In other words, we are checking whether B can be abduced from A (or, for explaining A) in the context of Γ. Usually, we additionally impose that the formula B belongs to a certain class.

Logical implication and argumentation

We discuss two basic forms of argumentation and its formalization using logical implication.

- Support

  To support a sentence S while arguing, one can exhibit a set of sentences Γ (accepted as valid) that logically implies S

  $$\Gamma \text{ supports } S \text{ if } \Gamma \models S$$

  Example: "Paolo è maggiorenne" is supported by "Paolo ha la patente" and "patente $\rightarrow$ maggiorenne".

- Refutation

  To refute a sentence S while arguing, one can exhibit a set of sentences Γ (accepted as valid) that logically implies the negation of S

  $$\Gamma \text{ refutes } S \text{ if } \Gamma \models \neg S$$

  Example: "Paolo è minorenne" is refuted by "Paolo ha la patente", "patente $\rightarrow$ maggiorenne", "maggiorenne $\rightarrow \neg$ minorenne".