Logic and Knowledge Representation

Propositional Logic

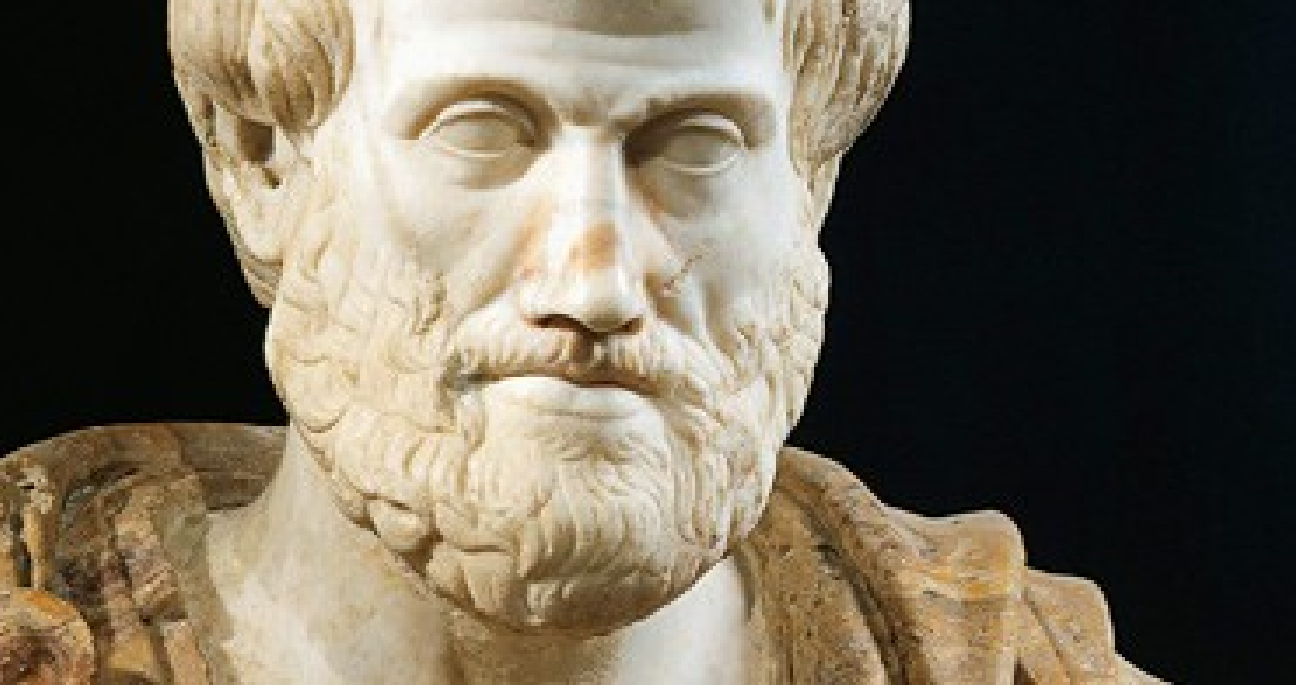
4 May 2018

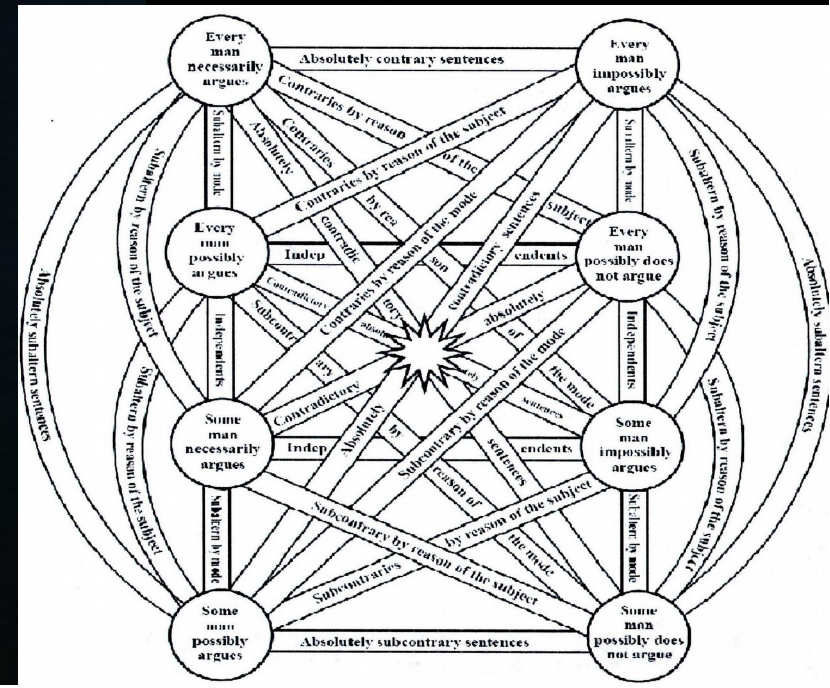
Giovanni Sileno gsileno@enst.fr

Télécom ParisTech, Paris-Dauphine University

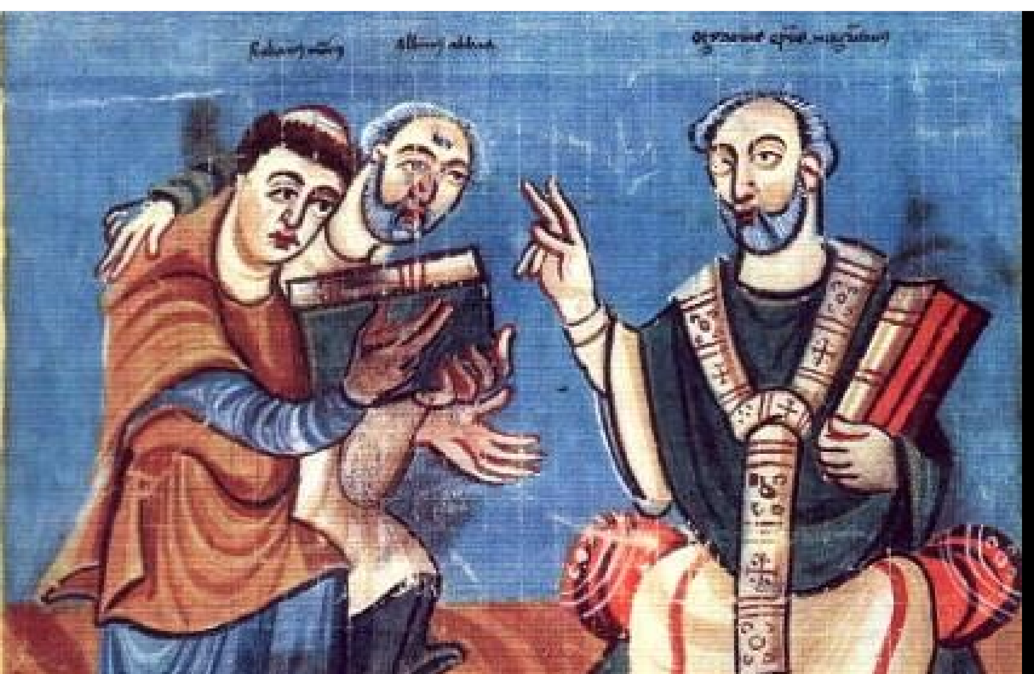

LA VÉRITÉ SORT DE LA BOUCHE DES ENFANTS
C'EST PAS VRAI !

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 (???)

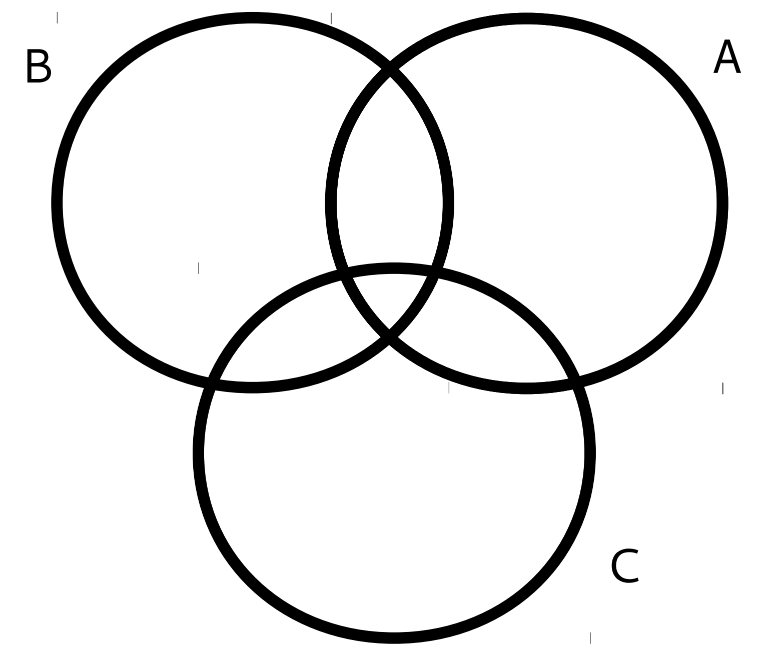
All bankers are athletes.
No consultant is a banker.
Therefore....

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.



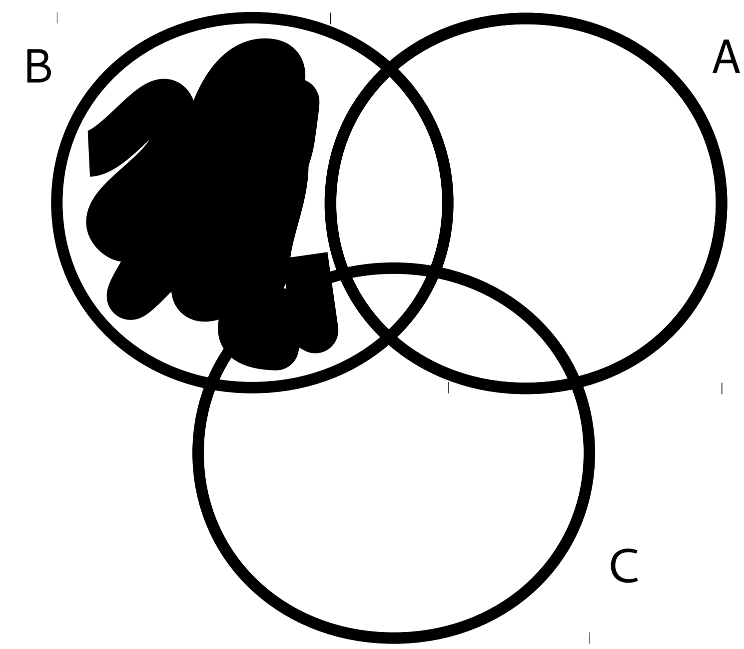
- Valid, and not trivial.

All bankers are athletes.

No consultant is a banker.

Therefore....

some athlete is not a consultant.



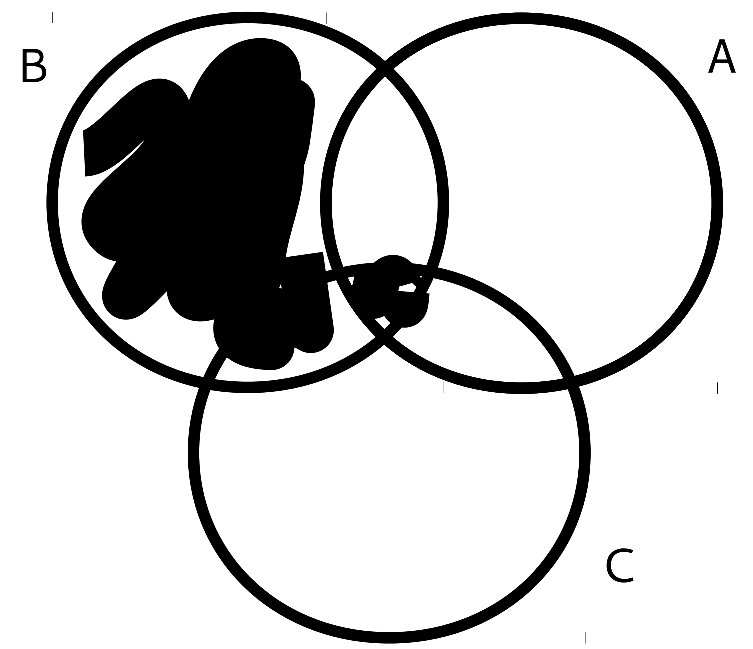
- Valid, and not trivial.

All bankers are athletes.

No consultant is a banker.

Therefore....

some athlete is not a consultant.



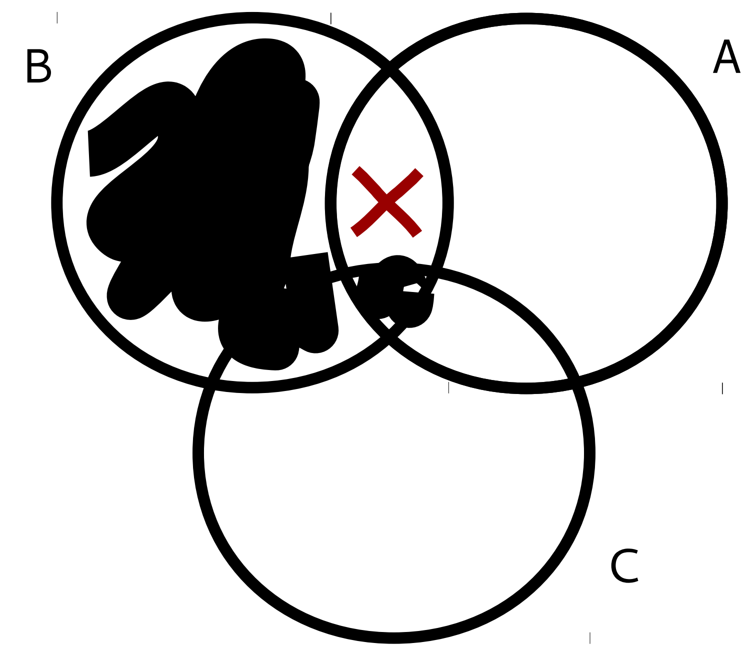
- Valid, and not trivial.

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, *Bregriffschrift* (1879)
    - Quantification
  - Charles Peirce
    - Reasoning and logic
  - Guiseppe Peano
    - Logical Axiomatization of Artihmetics
  - Bertand Russell & Alfred N. Whitehead, *Principia Mathematica* (1925)
    - Logical Axiomization of Mathematics

Propositional logic

A language consists of **symbols**, ...

- **alphabet**
  - *propositional symbols*

    p1, p2, ...

A language consists of **symbols**, ...

- **alphabet**
  - *propositional symbols*

    p1, p2, ...
  - *connectives*
    - nullary: T, ⊥ (top, bottom)
    - unary: ¬ (negation)
    - binary: ∧, ∨, ⊃, ⊂, ↑, ↓, ⊅, ⊄, ≡, ≠ (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, $\perp$

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, $\perp$
- **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 o G) is in P, where o is a binary connective ($\wedge$, $\vee$, $\supset$, $\subset$, $\uparrow$, $\downarrow$, $\not\supset$, $\not\subset$, $\equiv$, $\neq$).
  - 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***?

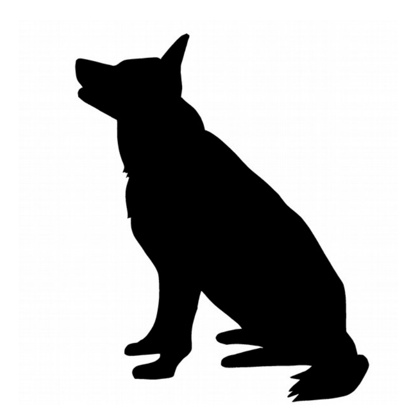
A language consists of a **semantic** (rules to interpret its expressions)

- 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

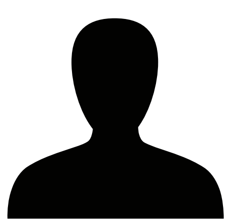
A language consists of a **semantic** (rules to interpret its expressions)

- 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“ $\longrightarrow$ TRUE
- “a dog is a mammal“ $\longrightarrow$ 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: Tv = {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 → Tv

`not`

T	F
F	T

Truth space and functions

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

not

T	F
F	T

  - 16 Binary functions: Tv x Tv → Tv

or

T	T	T
T	F	T
F	T	T
F	F	F

and

T	T	T
T	F	F
F	T	F
F	F	F

imp

T	T	T
T	F	F
F	T	T
F	F	T

Connecting syntax with semantics

- **Boolean valuation**, a function mapping propositions to truth values: v: P → Tv

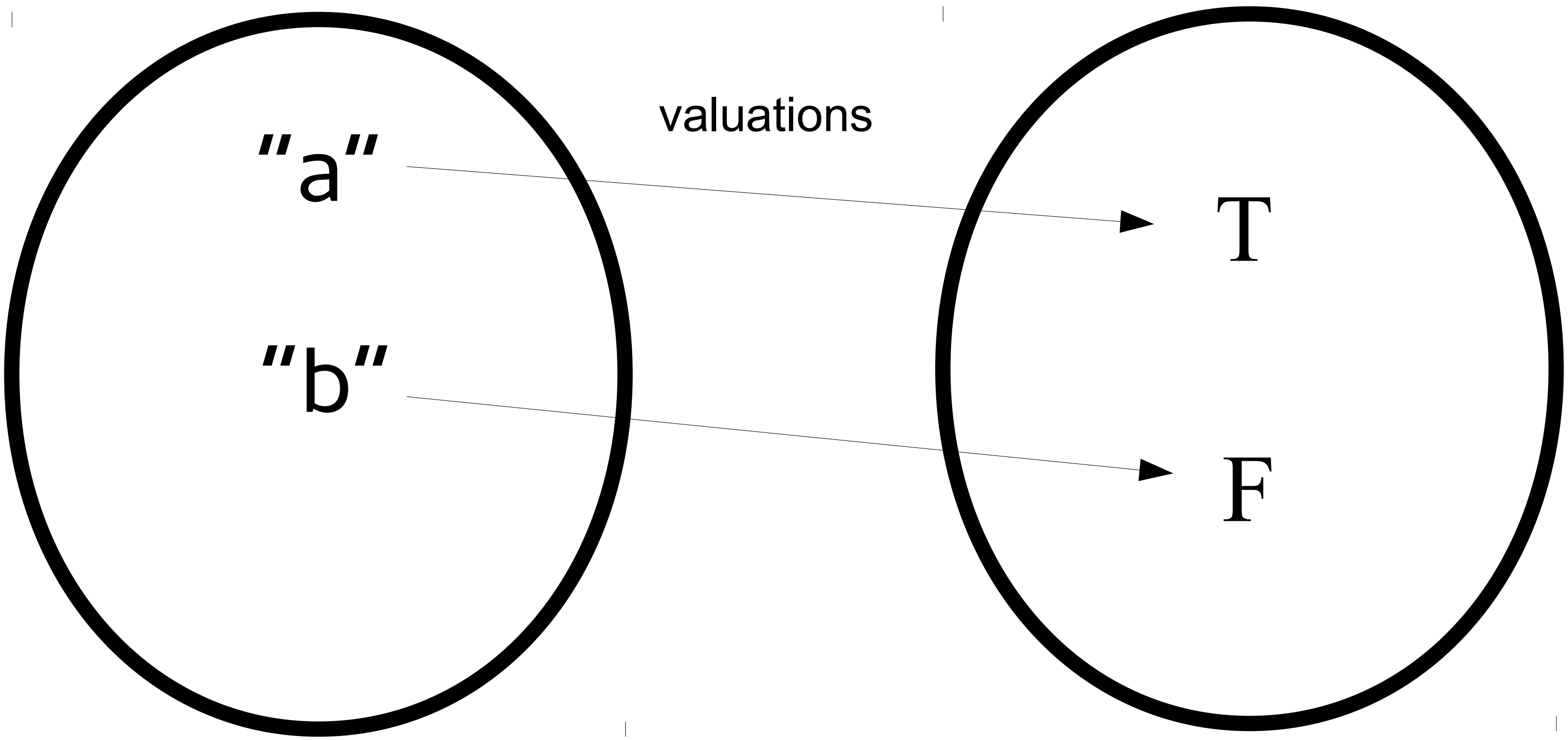
  - $v(\top) = \mathrm{T}$
  - $v(\bot) = \mathrm{F}$
  - $v(\neg X) = \text{not } v(X)$
  - $v(X \circ Y) = v(X) \bullet v(Y)$

  - o *syntactic connectives* ¬ ∧ ∨ ⊃, → ....
  - • *semantic connectives* `not` `and` `or` `implies`, ⇒ ....

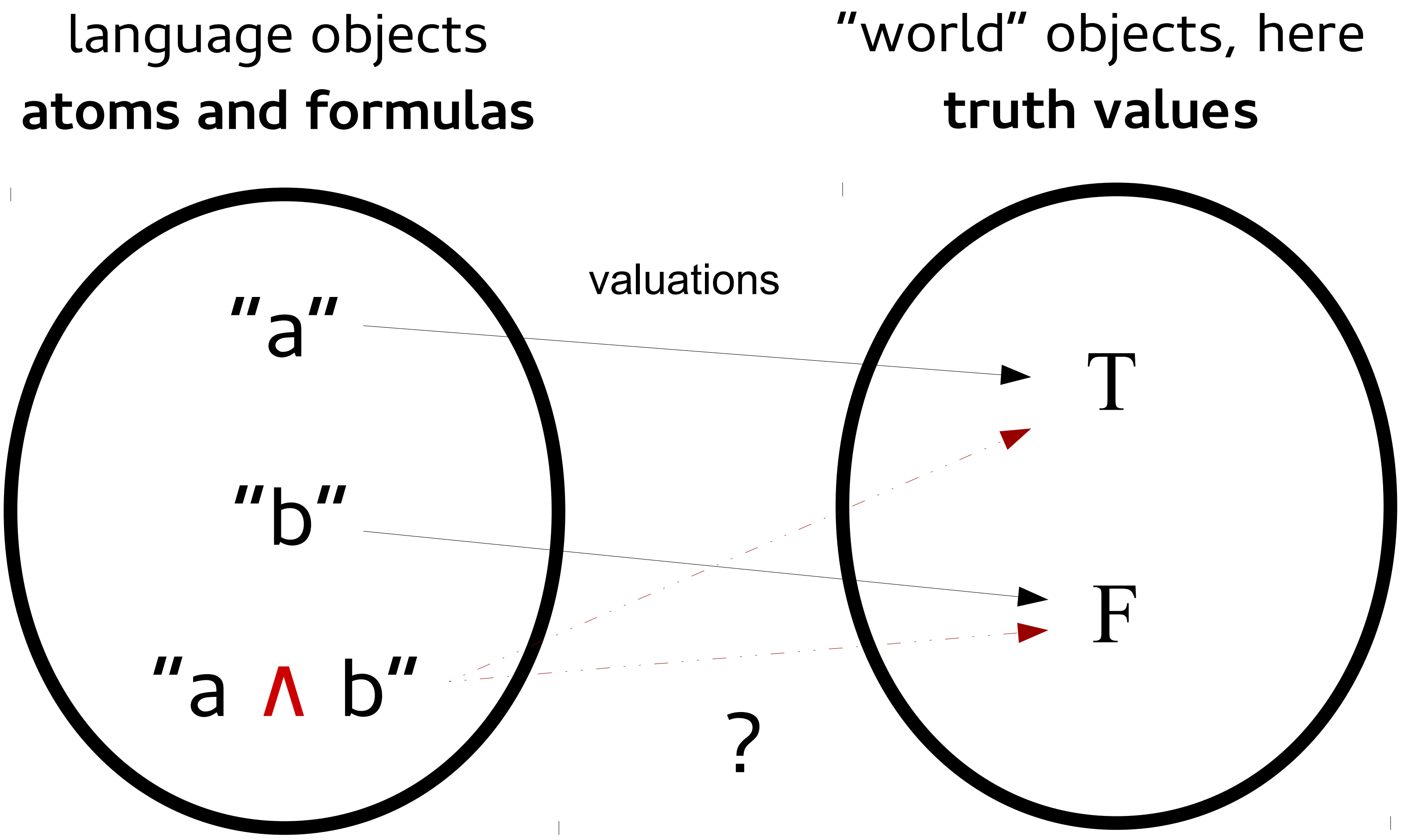
syntax vs semantics

language objects
atoms and formulas

"world" objects, here
truth values



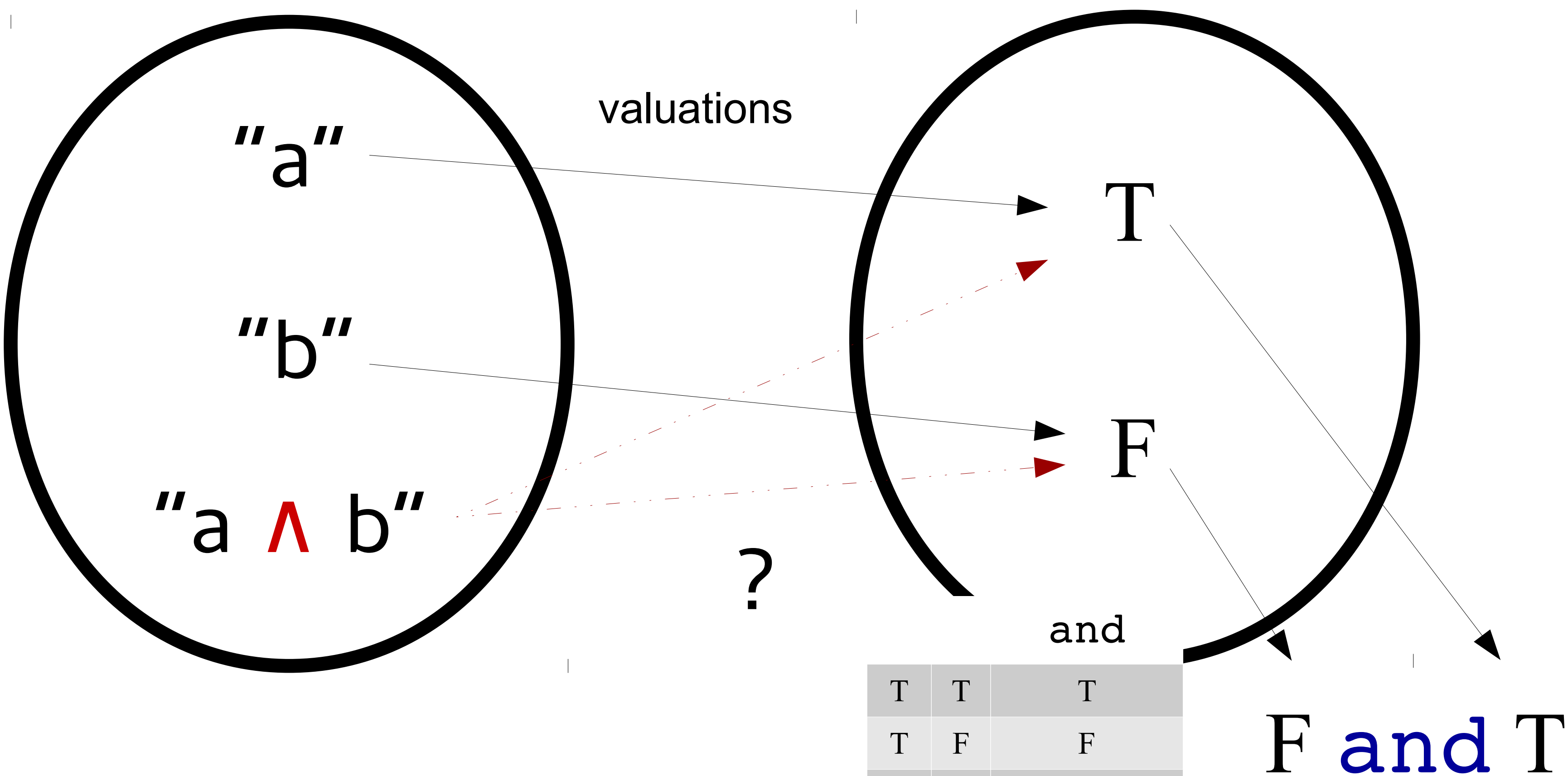
syntax vs semantics



syntax vs semantics

language objects
atoms and formulas

“world” objects, here
truth values

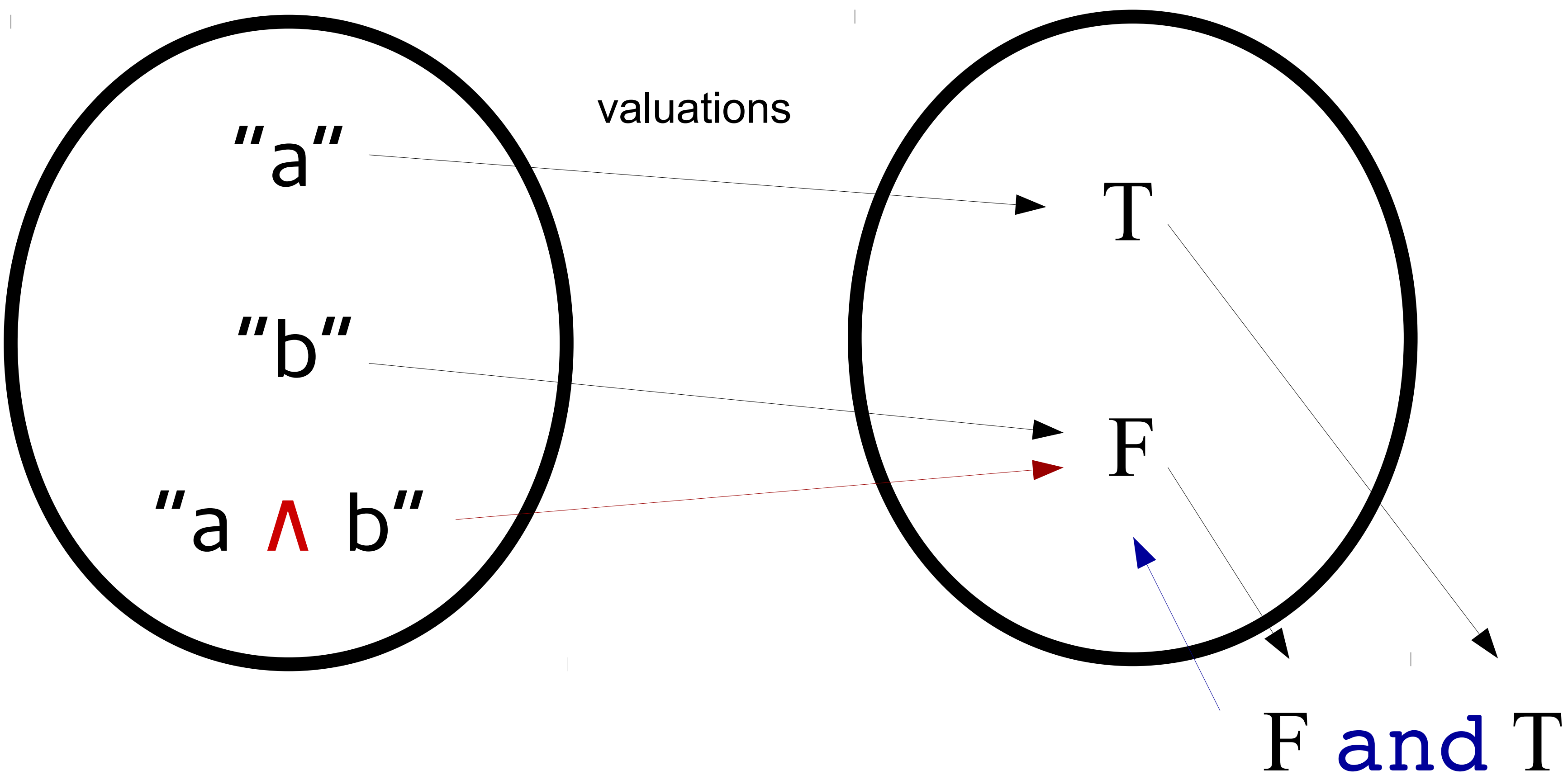


T	T	T
T	F	F
F	**T**	**F**
F	F	F

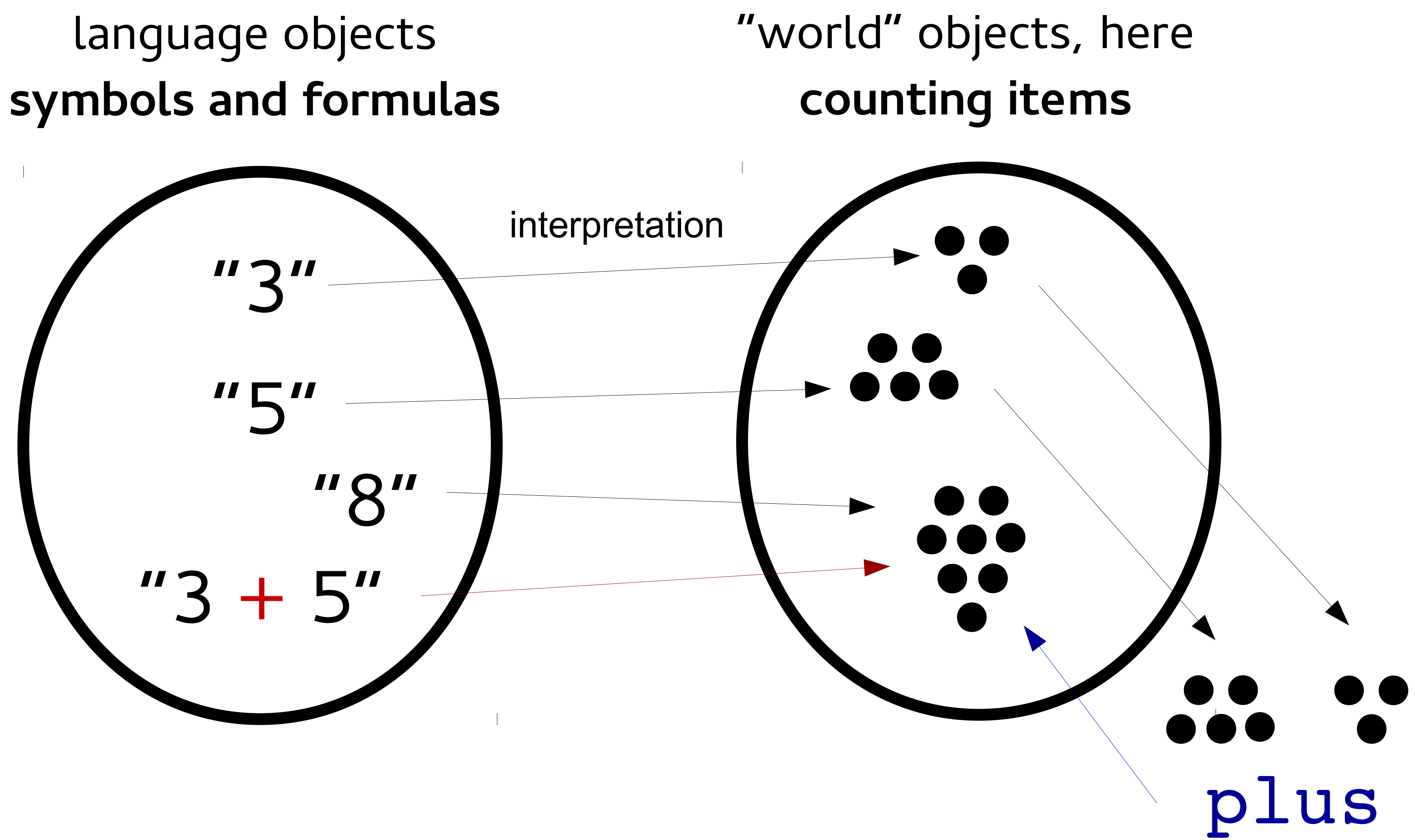
syntax vs semantics

language objects
atoms and formulas

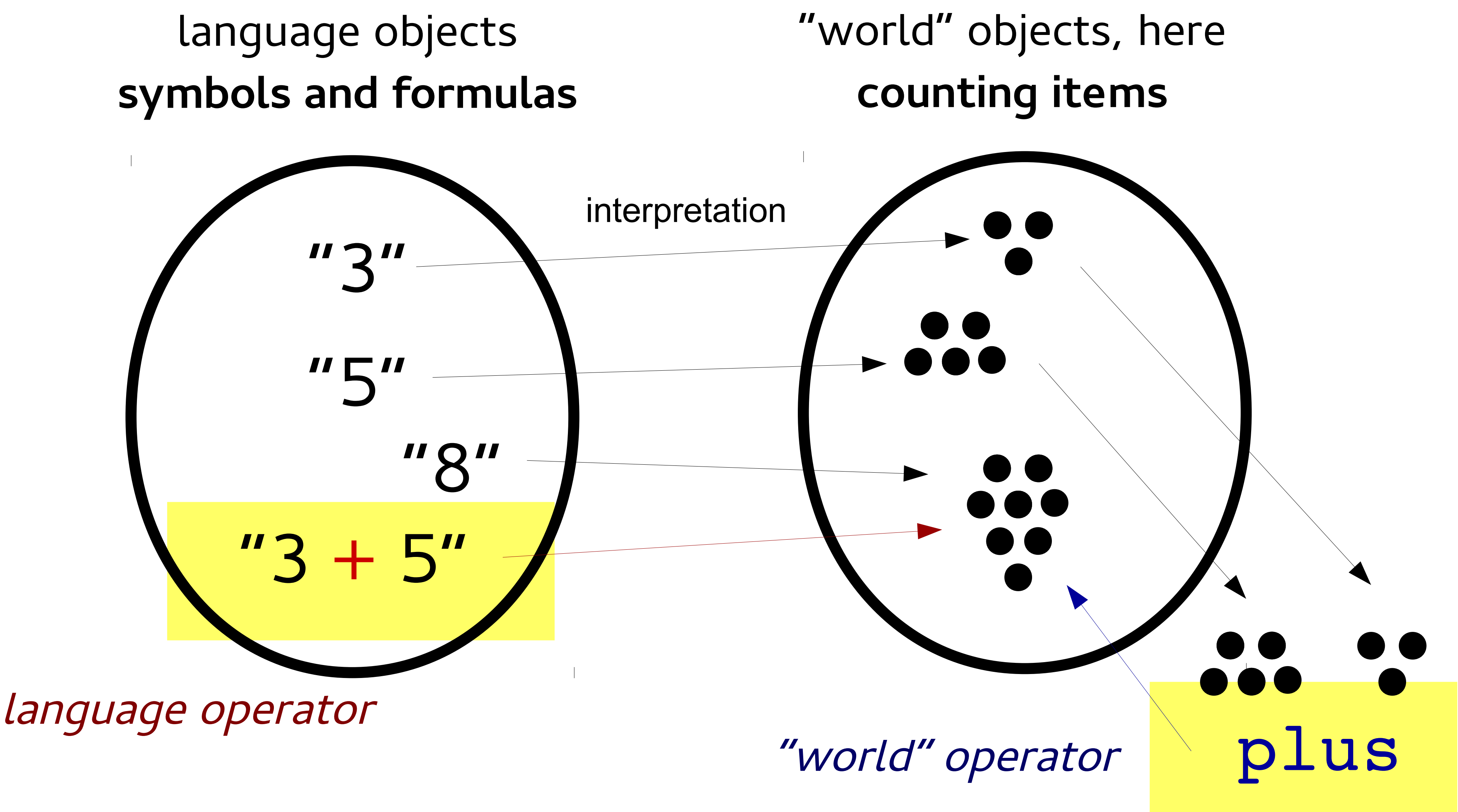
“world” objects, here
truth values



syntax vs semantics 2



syntax vs semantics 2



syntax vs semantics 3

```
% symbols of DSL and priority

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

% knowledge base

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

fact(sprinkler).
```

language objects?

language operators?

"world" objects?

"world" operators?

```
% 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).
```

"I know where that plane is going.
To an airport."

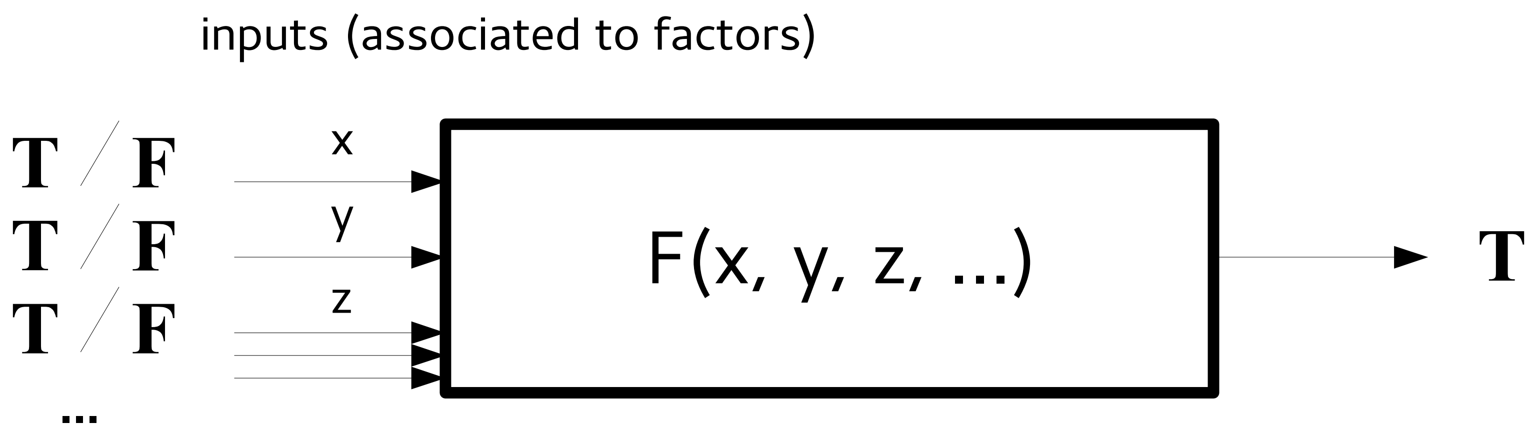
Tautologies & co.

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



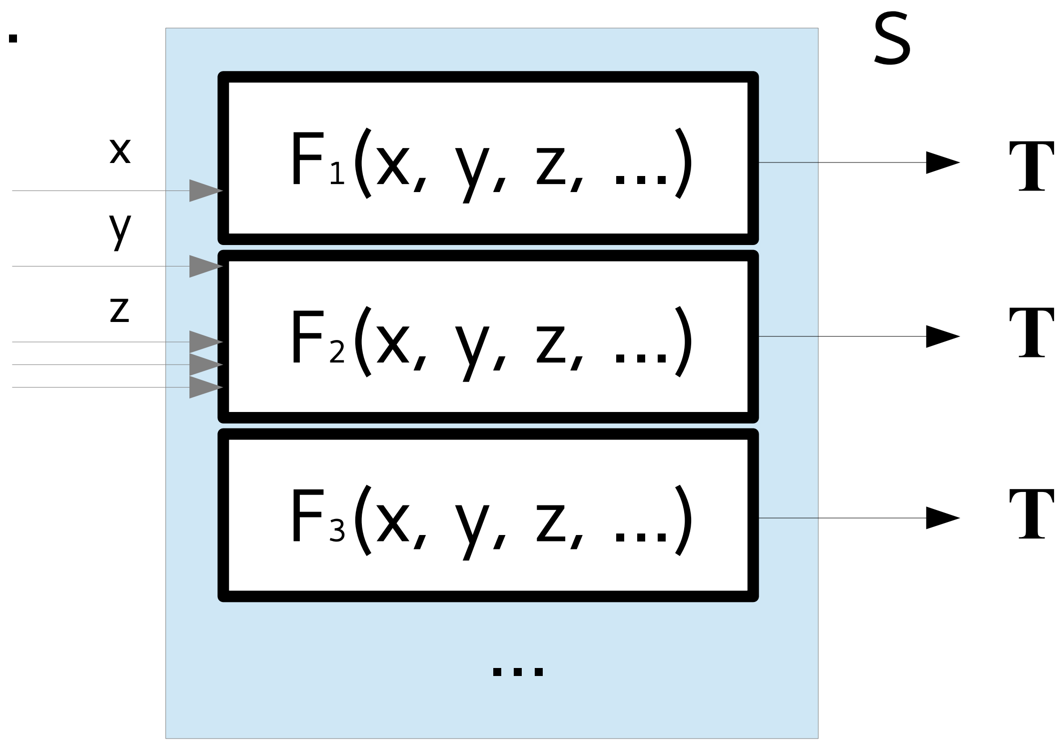
Functional view:
any configuration of inputs brings the same outcome **T**

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 $v(F) = \mathbf{T}$ 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:
  - $v_i(F) = \mathbf{T}$ for all F of S.

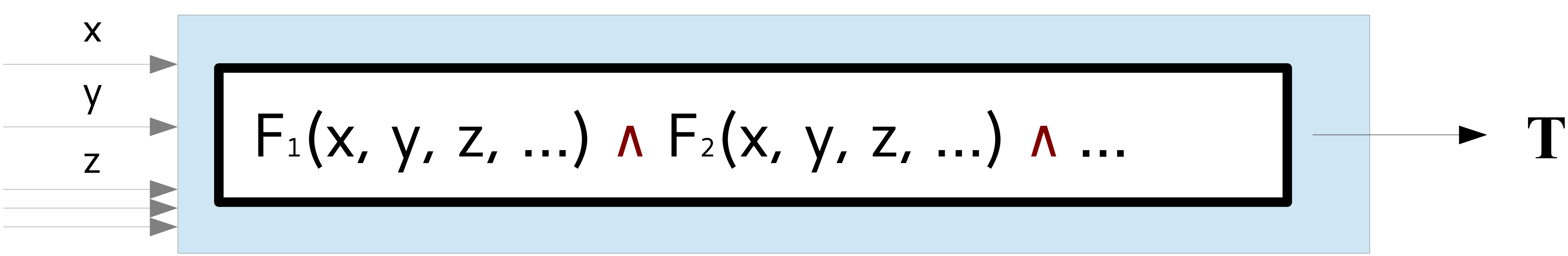


Functional view:
There is a configuration of inputs making all outputs $\boldsymbol{T}$.

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:
  - $v_i(F) = \mathbf{T}$ for all F of S.

$S = \{ F_{1,}, F_2 , ...\}$



Functional view:

There is a configuration of

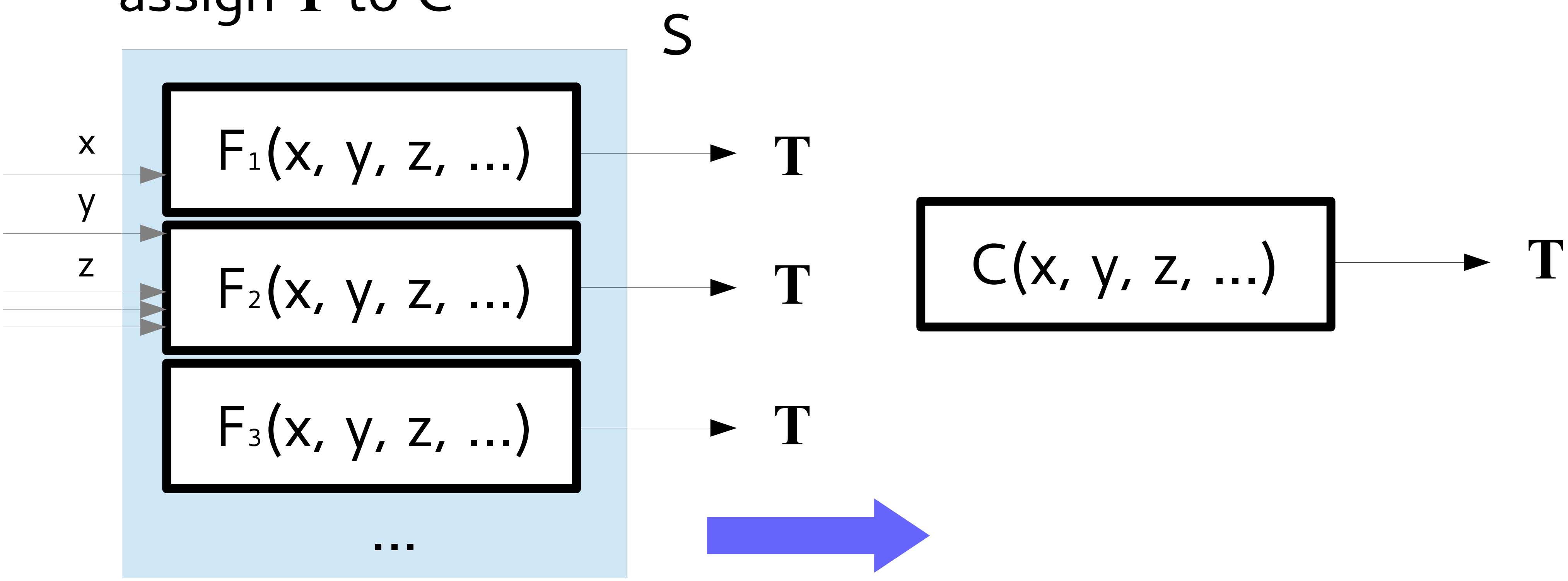
inputs making ***T*** *the output of the conjunction of the formula in S.*

Tautology, satisfiability, consequence

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

Tautology, satisfiability, consequence

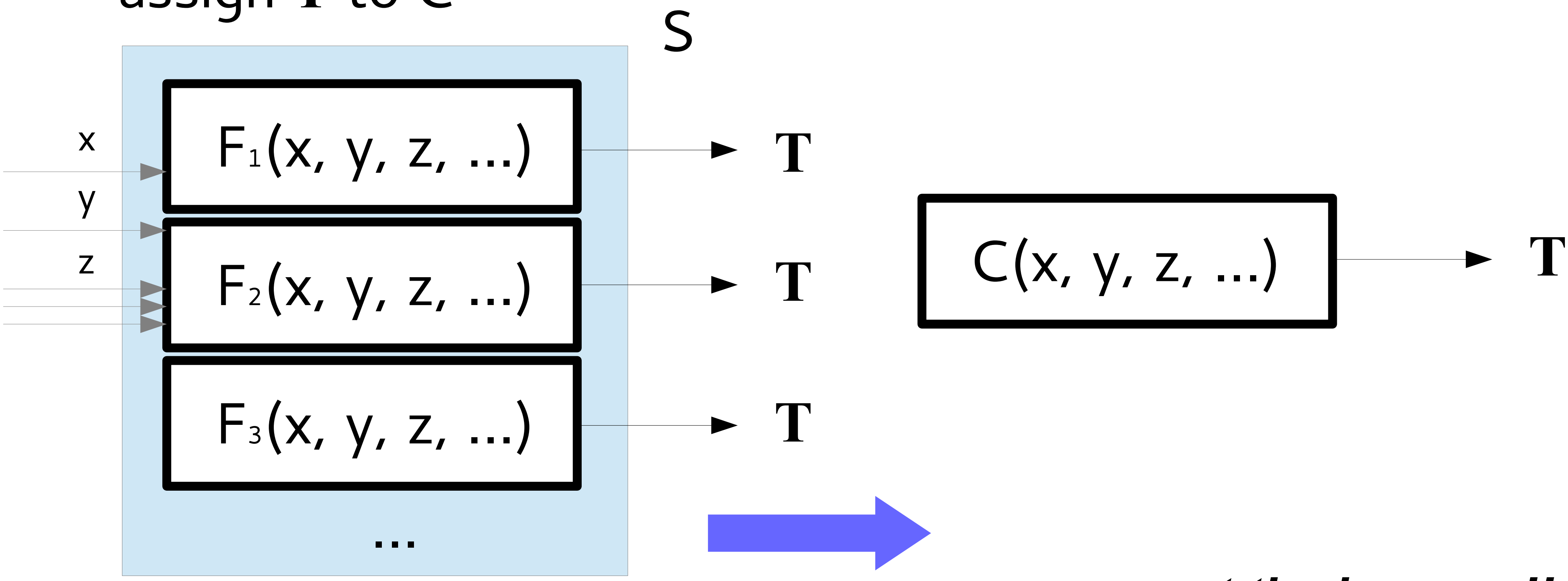
- S ⊨ C is called **semantic consequence**: if a valuation assigns the value **T** to all element of S, then it will assign **T** to C



Functional view:
All configurations making **T** *the outputs of S make C true.*

Tautology, satisfiability, consequence

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



not the inverse !!

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 $\mathbf{T}$ to all element of S, then it will assign $\mathbf{T}$ to C
- $\models$ C denotes the fact that C is a *tautology*.

$$\top \longrightarrow \mathbf{T} \Longrightarrow \boxed{C(x, y, z, \ldots)} \longrightarrow \mathbf{T}$$

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*.

Exercices (1):

- Show that X is a tautology only if $(X \equiv \top)$ is a tautology
- Show that $(\neg(X \wedge Y) \equiv (\neg X \vee \neg Y))$ is a tautology

Tautology, satisfiability, consequence

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

Exercices (2):

- (*ex falso quodlibet sequitur*). if $A, \neg A \in S$, then for any $X : S \vDash X$.
- (monotonicity). if $S \vDash X$, then $S \cup \{Y\} \vDash X$

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*.

Exercices (3):

- Show that $S \models C$ entails that $S \cup \{\neg C\}$ is not satisfiable.
- Show the reciprocal.

Tautology, satisfiability, consequence

- Central result:

C is a semantic consequence of S, i.e. $S \models C$

if and only if

$S \cup \{\neg C\}$ is not satisfiable

$\updownarrow$

*the conjunction of the formulas in S and the $\neg C$ 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.

  Exercices:

  - (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.

    $8 = (1 + 1) * (2 + 2)$
  - sums of products, e.g.

    $8 = (2 * 1) + (2 * 1) + (2 * 1) + (2 * 1)$

Rewriting of formulas

- Any number may be computed as
  - product of sums, e.g.

    8 = (1 + 1) * (2 + 2)
  - sums of products, e.g.

    8 = (2 * 1) + (2 * 1) + (2 * 1) + (2 * 1)

- In the algebraic interpretation of boolean logic,
  - **conjunction ∧** *stands for* **product ***
  - **disjunction ∨** *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** $\wedge$ *stands for* **product** $*$
  - **disjunction** $\vee$ *stands for* **addition** $+$

Rewriting of formulas

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

For instance from truth tables:

a	b	F	¬F
T	T	T	F
T	F	F	T
F	T	T	F
F	F	T	F

$$DNF_F = (a \wedge b) \vee (a \wedge \neg b) \vee (\neg a \wedge \neg b)$$

$$CNF_F = \neg DNF_{\neg F} = \neg(a \wedge \neg b) = \neg a \vee b$$

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 `<C1,C2,C3>`.

propositional formula	CNF
$(a \supset \neg(b \supset c))$	$< [\neg a, b], [\neg a, \neg c] >$

Evaluation with CNF

- Evaluations are performed as follows :
  - v(`[X1,X2,…,Xn]`) = F
    if and only if v(Xi) = F for all I
  - v(<`C1,C2,…,Cm`>) = T
    if and only if v(Ci) = T for all I
  - empty clause: v([ ]) = F
  - empty conjunction: v(< >) = T.

Transforming a formula to CNF

α-formula	α1	α2
$(X \wedge Y)$	X	Y
$\neg(X \vee Y)$	$\neg X$	$\neg Y$
$\neg(X \supset Y)$	X	$\neg Y$
...		

β-formula	β1	β2
$(X \vee Y)$	X	Y
$\neg(X \wedge Y)$	$\neg X$	$\neg Y$
$(X \supset Y)$	$\neg X$	Y
...		

- 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 $((A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C)))$ 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 β, replace it with $\beta 1$, $\beta 2$.
  - If X is of type α, create two new lines L1 and L2, recopy the line L replacing α 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 β, replace it with $\beta1$, $\beta2$.
  - If X is of type α, create two new lines L1 and L2, recopy the line L replacing α with $\alpha1$ and $\alpha2$ 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 β, replace it with $\beta 1$, $\beta 2$.
  - If X is of type α, create two new lines L1 and L2, recopy the line L replacing α with $\alpha 1$ and $\alpha 2$ respectively.
- A resolution consists of concatenating two lines where X and ¬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 **<¬F>** and containing an empty clause [].

Example

Proof using the resolution method that:

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

a. $[\neg(((A \supset B) \wedge (B \supset C)) \supset \neg(\neg C \wedge A))]$ **negation of target**

Example

Proof using the resolution method that:

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

a. $[\neg(((A \supset B) \wedge (B \supset C)) \supset \neg(\neg C \wedge A))]$	**negation of target**
b. $[((A \supset B) \wedge (B \supset C))]$	development of a
c. $[(\neg C \wedge A)]$	development of a

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

d. $[(A \supset B)]$ — development of b

e. $[(B \supset C)]$ — development of b

Example

Proof using the resolution method that:

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

~~a. [$\neg(((A \supset B) \wedge (B \supset C)) \supset \neg(\neg C \wedge A))$]~~	~~**negation of target**~~
~~b. [$((A \supset B) \wedge (B \supset C))$]~~	~~development of a~~
c. [$(\neg C \wedge A)$]	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:

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

~~a. $[\neg(((A \supset B) \wedge (B \supset C)) \supset \neg(\neg C \wedge A))]$~~	~~**negation of target**~~
~~b. $[((A \supset B) \wedge (B \supset C))]$~~	~~development of a~~
c. $[(\neg C \wedge A)]$	development of a
~~d. $[(A \supset B)]$~~	~~development of b~~
~~e. $[(B \supset C)]$~~	~~development of b~~
f. $[\neg A, B]$	rewriting d
g. $[\neg B, C]$	rewriting e

Example

Proof using the resolution method that:

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

~~a. $[\neg(((A \supset B) \wedge (B \supset C)) \supset \neg(\neg C \wedge A))]$~~	~~**negation of target**~~
~~b. $[((A \supset B) \wedge (B \supset C))]$~~	~~development of a~~
~~c. $[(\neg C \wedge A)]$~~	~~development of a~~
~~d. $[(A \supset B)]$~~	~~development of b~~
~~e. $[(B \supset C)]$~~	~~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

Example

Proof using the resolution method that:

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

~~a. $[\neg(((A \supset B) \wedge (B \supset C)) \supset \neg(\neg C \wedge A))]$~~	~~**negation of target**~~
~~b. $[((A \supset B) \wedge (B \supset C))]$~~	~~development of a~~
~~c. $[(\neg C \wedge A)]$~~	~~development of a~~
~~d. $[(A \supset B)]$~~	~~development of b~~
~~e. $[(B \supset C)]$~~	~~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:

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

a. $[\neg(((A \supset B) \wedge (B \supset C)) \supset \neg(\neg C \wedge A))]$	**negation of target**
b. $[((A \supset B) \wedge (B \supset C))]$	development of a
c. $[(\neg C \wedge A)]$	development of a
d. $[(A \supset B)]$	development of b
e. $[(B \supset C)]$	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
k. $[C]$	resolving g and j

Example

Proof using the resolution method that:

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

a. $[\neg(((A \supset B) \wedge (B \supset C)) \supset \neg(\neg C \wedge A))]$	**negation of target**
b. $[((A \supset B) \wedge (B \supset C))]$	development of a
c. $[(\neg C \wedge A)]$	development of a
d. $[(A \supset B)]$	development of b
e. $[(B \supset C)]$	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
k. $[C]$	resolving g and j
l. $[\ ]$	resolving h and k

Prolog and resolution

```
a.
b.
c :- a, b.


?- c.
```

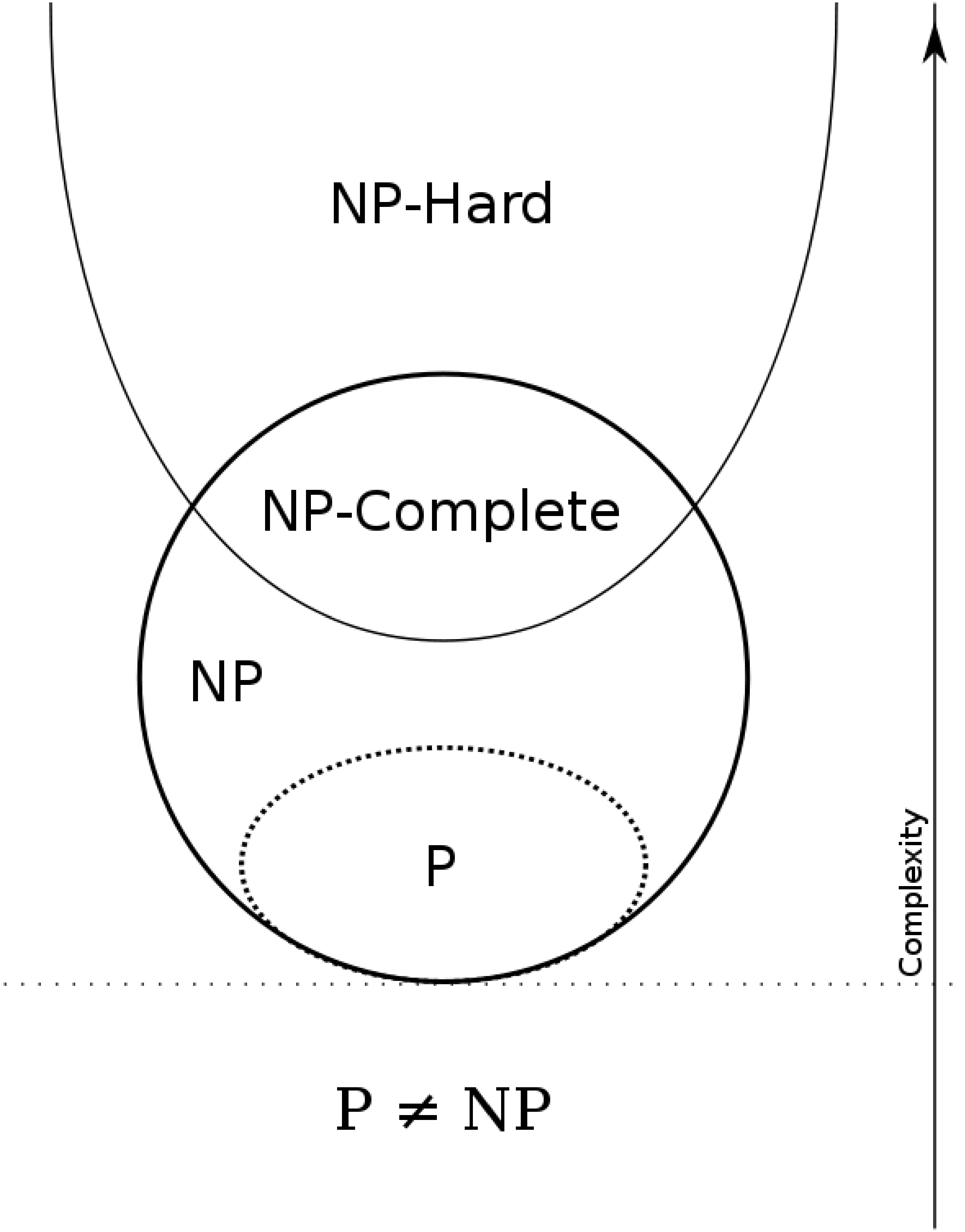
CNF

< [a],
[b],
[c, ¬a, ¬b],

[¬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 α, add $\alpha 1$ then $\alpha 2$ at the end of B.
  - If X is of type β, create a node and two new branches B1, B2 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 α, add $\alpha 1$ then $\alpha 2$ at the end of B.
  - If X is of type β, create a node and two new branches B1, B2 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*** :

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

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* : $\mathbf{\{ P, (P \supset Q) \} \vdash Q}$

  $\{ (P \supset Q) \} \vdash (P \supset Q)$ trivial

  $\{ (P \supset Q) \} \cup \{ P \} \vdash Q$ 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**)

- $\mathbf{((P \supset (Q \supset R)) \supset (Q \supset (P \supset R)))}$ is a theorem :

$\{ (P \supset (Q \supset R), P, Q \} \vdash R$	applying twice modus ponens
$\{ (P \supset (Q \supset R), Q\} \vdash (P \supset R)$	deduction theorem
$\{ (P \supset (Q \supset R)\} \vdash (Q \supset (P \supset R))$	deduction theorem
$\vdash ((P \supset (Q \supset R) \supset (Q \supset (P \supset R)))$	deduction theorem

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 $S \nvdash F$

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 $\nvdash$ Consistent(S)

a system cannot proof its own consistency