# Mathematical Logics

## First Order Logic: Reasoning and Tableaux*

Fausto Giunchiglia and Mattia Fumagalli

University of Trento


*Originally by Luciano Serafini and Chiara Ghidini
Modified by Fausto Giunchiglia and Mattia Fumagalli

# Reasoning tasks in FOL

In First order logics we have the same reasoning tatsk as in propositional logics (and any other logics)

## Model checking

For a closed formula $\varphi$ check if $I \models \varphi$

## Satisfiability

Find an interpretation $I$ that satisfies a closed formula $\varphi$. I.e., check if there is a I such that $I \models \varphi$.

## Validity

Check if a formula $\varphi$ is valid, i.e., if for all interpretations $I$, $I \models \varphi$

## Logical consequence

Check if a formula $\varphi$ is a logical consequence of a set of formulas $\Gamma$, i.e., $\Gamma \models \varphi$

# Reasoning tasks in FOL

- FOL has to do with objects which have some properties, we might be interested in knowing the set of objects which share a given property. More in general we might be interested in knowing the set of *n*-tuples of objects which are in a certain *n*-ary relation.
- This task is similar to what we do when we query a database. E.g. we want to know the set of people who earn more than 1300 euro per month, or the set of pair of people who works in the same project.
- A property in FOL can be expressed by a formula with free variable $\varphi(x_1, \ldots, x_n)$.
- $person(x) \wedge earn(x, y) \wedge y > 1000$: the persons (free variable $x$) who earns more than 1000 euros
- $\exists z\ (worksFor(x, z) \wedge worksFor(y, z))$: the pairs of people (the free variables $(x, y)$) who works in the same project.

# Query answering in FOL

## Query answering

Given an interpretation $I$ (a database instance) of a FOL L and a formula $\varphi(x_1, \ldots, x_n)$ with $n$-free variables, find all the $n$-tuples of elements of the domain $(d_1, \ldots, d_n) \in (\Delta^I)^n$ such that

$I \models \varphi[a[x_1/d_1 \ldots x_n/d_n]]$

# Example of query

## Exampl

What is the result of the following queries agains the interpretation above?

1. $friends(x, alice)$ $\{1, 4\}$
2. $\neg friends(x, bob)$ $\{2, 3, 5, 6\}$
3. $friends(x, y) \wedge friends(y, z)$ $\{(1, 2, 1), (1, 2, 4), (2, 1, 2), (2, 1, 4), (3, 4, 3), (4, 3, 4), (4, 2, 4), (4, 1, 4), (4, 4, 1), (4, 4, 2), (4, 4, 3), (4, 4, 4)\}$

   $\exists y\,(friends(x, y) \wedge friends(y, z))$ $\{(1, 1), (1, 4), (2, 2), (2, 4), (3, 3), (4, 4), (4, 1), (4, 2), (4, 3)\}$
4. $\forall y\,(friends(x, y) \rightarrow supervisor(x) = y)$ $\{3, 5, 6\}$ Notice that 5 and 6 are there because they don't have any friends so the premise of the implication is always false.

The interpretation $I$ is defined as follows:

**Symbols** Constants: *alice*, *bob*, *carol*, *robert* Function: *supervisor* (with arity equal to 1) Predicate: *friends* (with arity equal to 2)

**Domai**

**nterpretation**

$\Delta^I = \{1, 2, 3, 4, 5, 6\}$

$I(alice) = 1$, $I(bob) = 2$, $I(carol) = 3$, $I(robert) = 2$

$I(supervisor) = S$ $S(1) = 3$ $S(2) = 1$ $S(3) = 4$ $S(4) = 5$ $S(5) = 5$ $S(6) = 5$

$I(friends) = F = \{(1, 2), (2, 1), (3, 4), (4, 3), (4, 2), (2, 4), (4, 1), (1, 4), (4, 4)\}$

# Hilbert style axiomatization

**Axioms for propositional connectives** They are the same as in propositional logic

**A1** $\varphi \supset (\psi \supset \varphi)$

**A2** $(\varphi \supset (\psi \supset \vartheta)) \supset ((\varphi \supset \psi) \supset (\varphi \supset \vartheta))$

**A3** $(\neg\psi \supset \neg\varphi) \supset ((\neg\psi \supset \varphi) \supset \psi)$

**MP** $\dfrac{\varphi \quad \varphi \supset \psi}{\psi}$

**Axioms and rules for quantifiers**

**A4** $\forall x.(\varphi(x)) \supset \varphi(t)$ if $t$ is free for $x$ in $\varphi(x)$

**A5** $\forall x.(\varphi \supset \psi) \supset (\varphi \supset \forall x.\psi)$ if $x$ does not occur free in $\varphi$

**Gen** $\dfrac{\varphi}{\forall x.\varphi}$

# Example of Hilbert style proof in FOL

## Example

To show that the formula $P(a) \supset \neg \forall x \neg P(x)$ is valid we have to generate a sequence of formulas (i.e., a Hilbert proof) starting from the axioms (A1-A5), using the rules (MP) and (GEN). (In the example we only report the inferences that involves first order reasoning, propositional proofs are omitted)

(1) $\forall x \neg P(x) \supset \neg P(a)$ instance of (A4)

⋮ a proof in Propositional Logic

(2) $(\varphi \supset \psi) \supset (\neg\psi \supset \neg\varphi)$

(3) $(\forall x \neg P(x) \supset \neg P(a)) \supset (\neg\neg P(a) \supset \neg\forall x \neg P(x))$ Instance of (2)

(4) $\neg\neg P(a) \supset \neg\forall x \neg P(x)$ From (1) and (3) by (MP)

⋮ a proof in Propositional Logic

(5) $(\neg\neg\varphi \supset \psi) \supset (\varphi \supset \psi)$

(6) $(\neg\neg P(a) \supset \neg\forall x \neg P(x)) \supset (P(a) \supset \neg\forall x \neg P(x))$ Instance of (5)

(7) $P(a) \supset \neg \forall x \neg P(x)$ from (4) and (6) by (MP)

# Automatic reasoning based on Hilber Style

- Hilbert style proof system was invented with the main purpose of describing the minimal rational assumptions behind mathematical reasoning.
- Hilber style proofs are suppesed to be provided by humans, who can use their intuition to apply smart heuristics to generate them.
- Writing an algorithm that decides on the validity of a formula by searching a Hilbert style proof, is not a good idea.
- We look at alternative ways to write algorithms for deciding the falidity of a FOL formula.

# Tableaux Calculus

- The Tableaux Calculus is an algorithm solving the problem of satisfiability.
- If a formula is satisfiable, then there exists an open branch in the tableaux of this formula.
- the procedure attempts to construct the tableaux for a formula. Sometimes it's not possible since the model of the formula is infinite.
- The basic idea is to incrementally build the model by looking at the formula, by decomposing it in a top/down fashion. The procedure exhaustively looks at all the possibilities, so that it can possibly prove that no model could be found for unsatisfiable formulas.

# Semantic tableaux

**Definition**

A tableau is a rooted tree, where each node carries a first order sentence (closed formula), and the children of a node *n* are generated by applying a set of expansion rules to *n* or to one of the ancestors of *n*.

**Definition**

The expansion rules for a first order semantic tableaux are those for the propositional semantic tableaux, extended with the following rules that deal with the quantifiers:

$\gamma$ rules $\quad \dfrac{\forall x.\varphi(x)}{\varphi(t)} \quad \dfrac{\neg\exists x.\varphi(x)}{\neg\varphi(t)}$

Where $t$ is a term free for $x$ in $\varphi$

$\delta$ rules $\quad \dfrac{\neg\forall x.\varphi(x)}{\neg\varphi(c)} \quad \dfrac{\exists x.\varphi(x)}{\varphi(c)}$

where $c$ is a new constant not previously appearing in the tableaux

# Tableaux production rules for propositional logic

## . . . for propositional connectives

$\alpha$ rules

$$\frac{\varphi \wedge \psi}{\begin{array}{c}\varphi\\ \psi\end{array}} \qquad \frac{\neg(\varphi \wedge \psi)}{\begin{array}{c}\neg\varphi\\ \neg\psi\end{array}} \qquad \frac{\neg\neg\varphi}{\varphi} \qquad \frac{\neg(\varphi \supset \psi)}{\begin{array}{c}\varphi\\ \neg\psi\end{array}}$$

$\beta$ rules

| $\varphi \wedge \psi$ | | $\varphi \supset \psi$ | | $\neg(\varphi \wedge \psi)$ | | $\varphi \equiv \psi$ | | $\neg(\varphi \equiv \psi)$ | |
|---|---|---|---|---|---|---|---|---|---|
| $\varphi$ | $\psi$ | $\neg\varphi$ | $\psi$ | $\neg\varphi$ | $\neg\psi$ | $\varphi$<br>$\psi$ | $\neg\varphi$<br>$\neg\psi$ | $\varphi$<br>$\neg\psi$ | $\neg\varphi$<br>$\psi$ |

# Substitution $\varphi[x/t]$

If $\varphi(x)$ is a free variable and $t$ is a term, we use the notation $\varphi(t)$ instead of the more precise notation $\varphi[x/t]$ to represent the substitution of $x$ for $t$ in $\varphi$.

**Substitution**

$\varphi[x/t]$ denotes the formula we get by replacing each free occurrence of the variable $x$ in the formula $\varphi$ by the term $t$. This is admitted if $t$ does not contain any variable $y$ such that $x$ occurs in the scope of a quantifier for $y$ (i.e., in the scope of $\forall y$ or $\exists y$).

# Substitution $\varphi[x/t]$

**Example (of substitution)**

$$P(x, y, f(x))[x/a] = P(a, y, f(a))$$

$$\forall x P(x, y)[x/b] = \forall x P(x, y)$$

$$\exists x P(x, x) \wedge Q(x)[x/c] = \exists x P(x, x) \wedge Q(c)$$

$$P(x, g(y))[y/f(x)] = P(x, g(f(x)))$$

$$\forall x.P(x, y)[y/f(x)] =$$ Not allowed since $f(x)$ is not free for $y$ in $\forall x.P(x, y)$

# Universal quantification rule

$$\frac{\forall x \varphi(x)}{\varphi(t)}$$

- $\forall x \varphi(x)$ means that for every object of the domain, the property $\varphi(x)$ should be true.
- a term $t$ that occurs in the tableaux denotes an object of the domain
- therefore, $\varphi(t)$ must be true for all the terms $t$ that occurs in the tableaux. I.e., the $\forall$ rule can be applied as many time as one want to any term that appear in the tableaux.

**Exercize**

Show that the following tableaux rule is sound.

$$\frac{\forall x \exists y P(y, x)}{\exists y P(y, f(x))}$$

# Existential quantification rule

$$\frac{\forall x \varphi(x)}{\varphi(c)} \quad \text{for a new constant } c$$

- $\exists x \varphi(x)$ means that for some object of the domain, the property $\varphi(x)$ should be true.
- we don't know which object of the domain has the property $\varphi$, we know only that there is one.
- this means that this rule cannot be applied to the terms that already occur in the tableaux, since otherwise we would introduce an unjustified joiche on the element that has the property $\varphi$.
- the trick is to introduce a term to denote an unconditioned objects (sometimes called "fresh" constant/variable) for denoting an "unknown" object, i.e., an object on which we haven't done any commitment.
- therefore we allow only to infer $\varphi(c)$ form $\exists x \varphi(x)$, where $c$ is fresh. Only one application is possible.

# Open and Closed Branches

- a tableaux rooted with $\varphi$ is a method to search an interpretation that satisfy $\varphi$
- Every branch of a tableaux with root equal to $\varphi$, corresponds to an attempt to find an interpretation $I$ that satisfies $\varphi$.
- The interpretation corresponding to a branch $b$ of a tableaux should satisfy all the formulas that appear in the branch.
- If the branch contains two opposite literals, i.e. $P(t_1, \ldots, t_n)$ and $\neg P(t_1, \ldots, t_n)$, then the branch cannot correspond to an interpretation, since there is no interpretation that satisfy at the same time $P(t_1, \ldots, t_n)$ and $\neg P(t_1, \ldots, t_n)$. So we can consider this attempt to find an interpretation failed. In this case we say that the branch is closed.
- if in a branch $b$ all the rules has been applied and there is no opposite literals, then this branch corresponds to an interpretation. We call such a branch open

# Open and Closed Branches

**Definition**

- A branch of a tableau is said to be closed if it contains a a pair of formulas $\varphi$ and $\neg\varphi$.
- A branch of a tableau is said to be open if it is not closed.
- A tableau is said to be closed if each of its paths is closed.

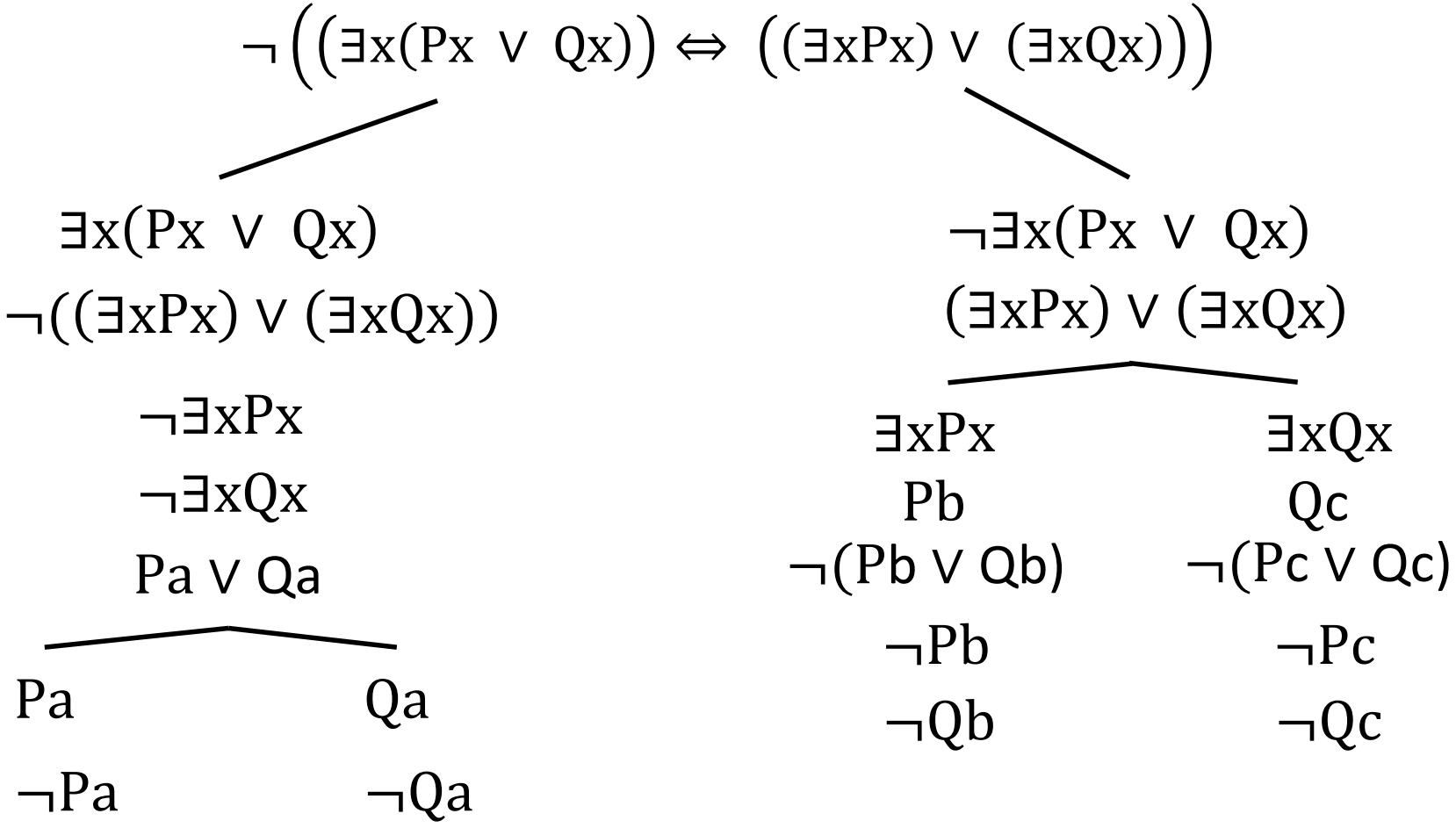
# The tableaux method

1. To test a formula $\varphi$ for validity, form tableau starting with $\neg\varphi$. If the tableau closes off, then $\varphi$ is logically valid.
2. To test whether $\varphi$ is a logical consequence of $\Gamma$ form a tableau starting with each formula in $\Gamma$ and $\neg\varphi$. If the tableau closes off, then $\varphi$ is indeed a logical consequence of $\Gamma$.
3. To test a set of formulas $\Gamma$ is satisfiable, form a tableau starting with $\Gamma$ or equivalently an unsigned If the tableau closes off, then $\Gamma$ is not satisfiable. If the tableau does not close off, then $\Gamma$ is satisfiable, and from any open branch we can read off an interpretation satisfying $\Gamma$.

# Example

**Example**

To check if the formula $(\exists x(P(x) \vee Q(x))) \equiv ((\exists xP(x)) \vee (\exists xQ(x)))$ is satisfiable, we start with a tableaux with this formula:


¬((∃x(Px ∨ Qx)) ⇔ ((∃xPx) ∨ (∃xQx)))
∃x(Px ∨ Qx)
¬((∃xPx) ∨ (∃xQx))
¬∃xPx
¬∃xQx
Pa ∨ Qa
Pa
¬Pa
Qa
¬Qa
¬∃x(Px ∨ Qx)
(∃xPx) ∨ (∃xQx)
∃xPx
Pb
¬(Pb ∨ Qb)
¬Pb
¬Qb
∃xQx
Qc
¬(Pc ∨ Qc)
¬Pc
¬Qc

# Practicing with Semantic Tableaux

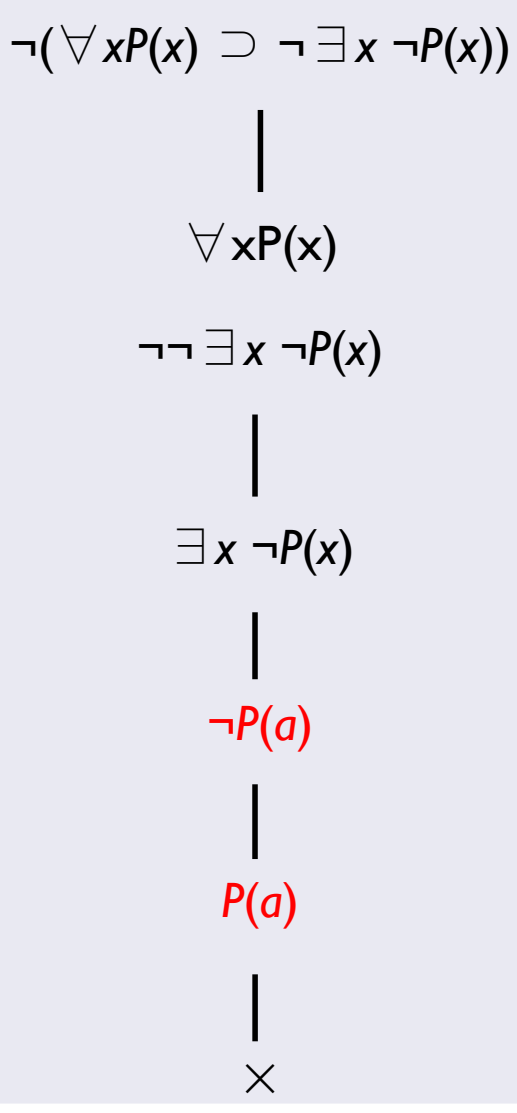
**Exercize**

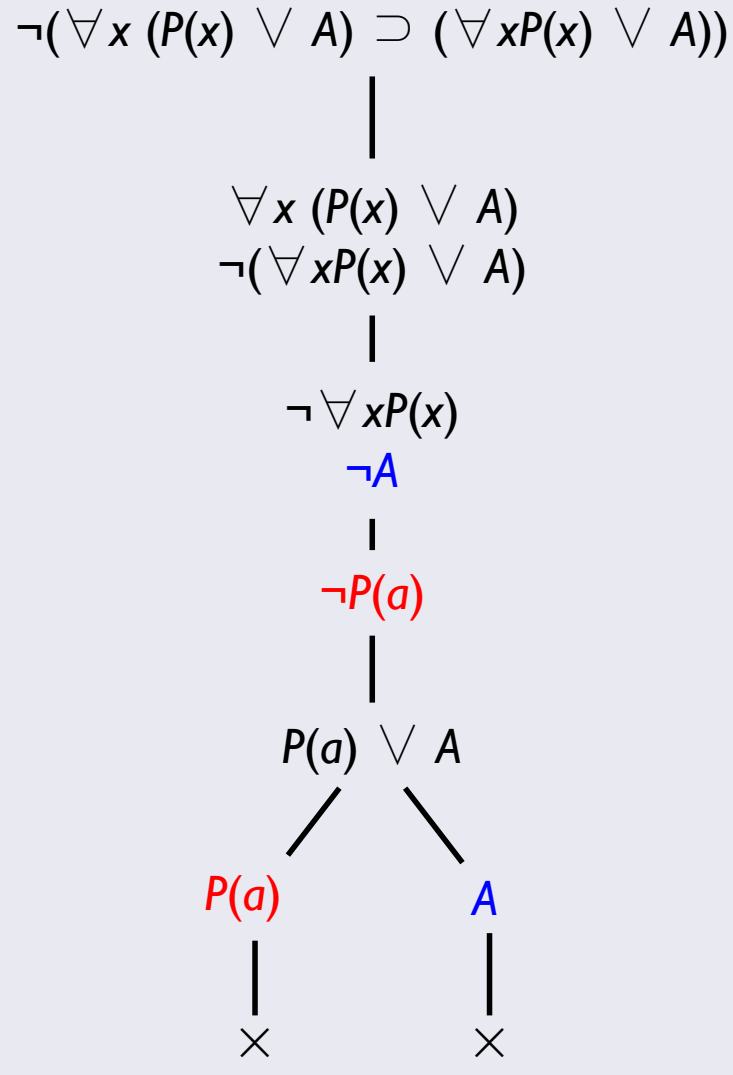
Show with the method of semantic tableaux that the following formulas are valid:

- $\forall x P(x) \supset \neg \exists x \neg P(x)$
- $\forall x\,(P(x) \vee A) \supset (\forall x P(x) \vee A)$ when $x$ is not free in $A$
- $\exists x\,(P(x) \supset \forall x P(x))$
- $\exists x\ \forall y P(x, y) \supset \forall y\ \exists x P(x, y)$

# Practicing with Semantic Tableaux

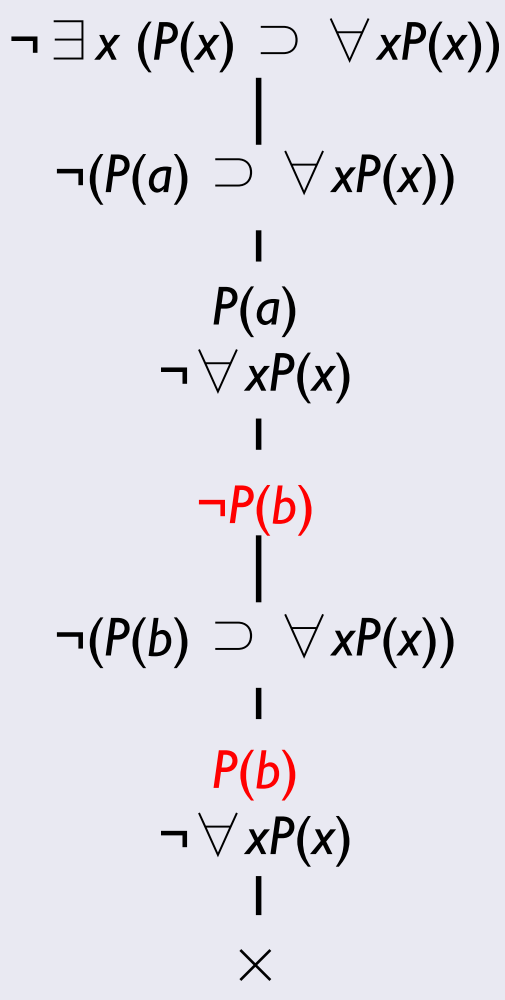
## Solutio

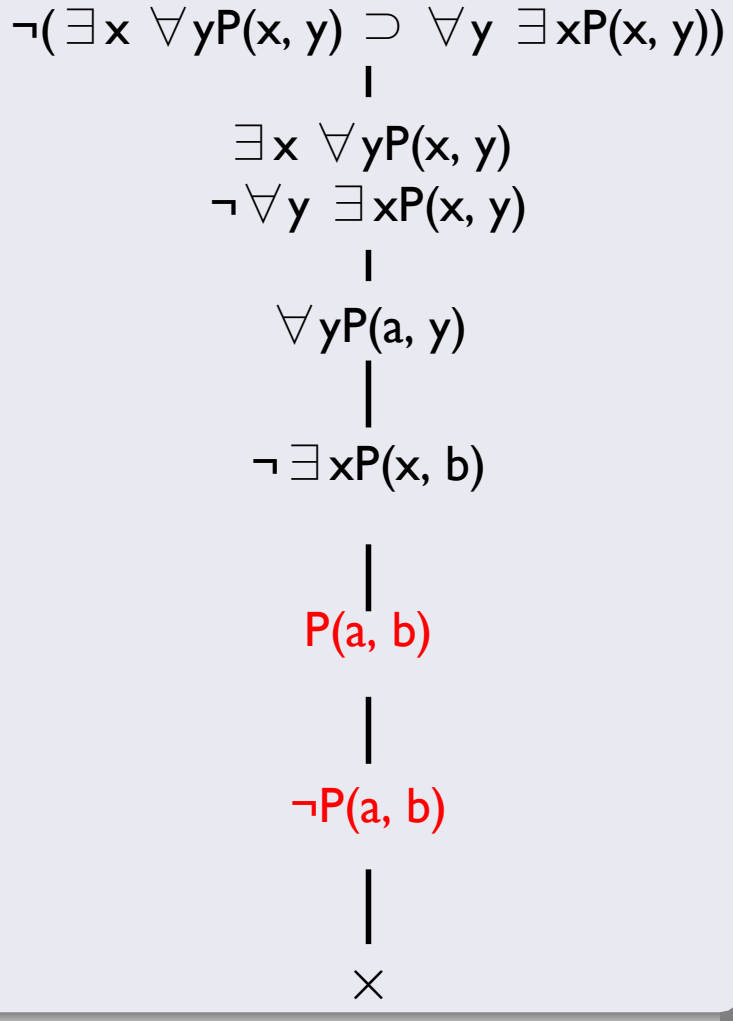


# Practicing with Semantic Tableaux

## Solution



# Example

## Example

Check if $\forall x P(x) \wedge \exists x \neg P(f(x))$ is valid/satisfiable/unsatisfiable.

## Solution

$$\forall x P(x) \wedge \exists x \neg P(f(x))$$

$$\Big|$$

$$\forall x P(x)$$
$$\exists x \neg P(f(x))$$

$$\Big|$$

$$\neg P(f(c))$$

*Now to expand $\forall x P(x)$, we can use any ground term t. Possible choices: c, f (c), f (f (c)), . . . . we choose f (c) because we want to create a clash with $\neg P(f(c))$.*

# Example (Cont'd)

## Example

Check if $\forall x P(x) \wedge \exists x \neg P(f(x))$ is valid/satisfiable/unsatisfiable.

## Solution

$\forall x P(x) \wedge \exists x \neg P(f(x))$

|

$\forall x P(x)$

$\exists x \neg P(f(x))$

|

$\neg P(f(c))$

|

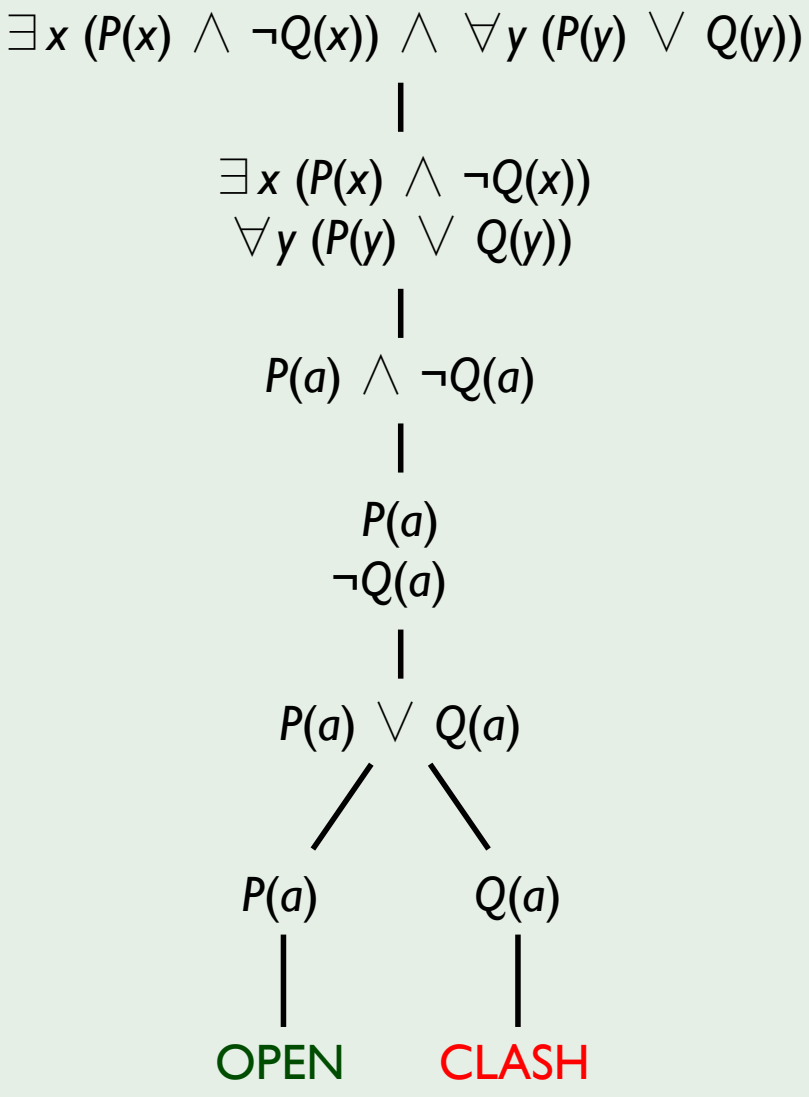
$P(f(c))$

|

$\times$

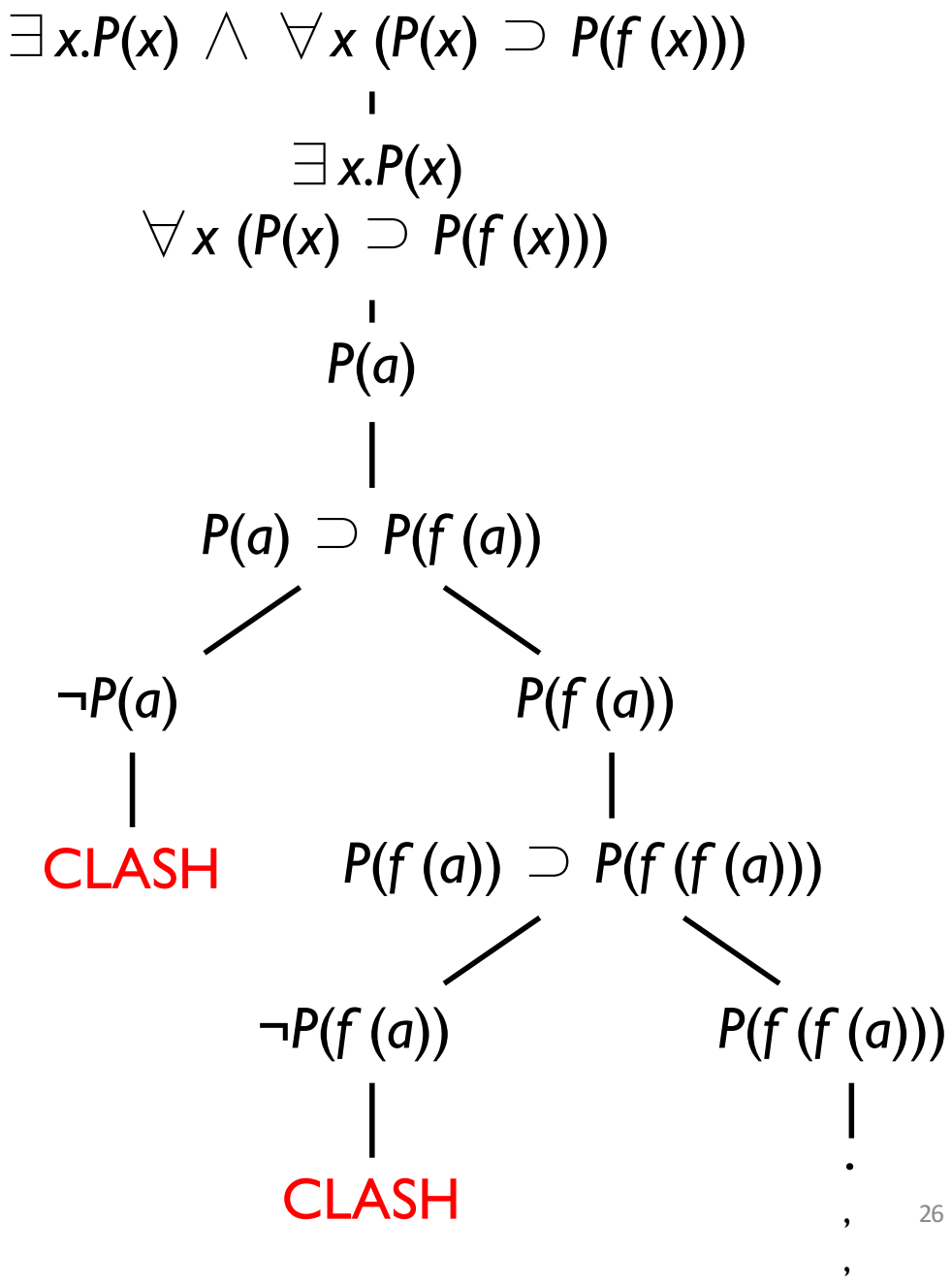
# Example of tableaux

## Example



# Termination

For certain formulas



# Tableaux

**Exercize**

G4ive tableau proofs for the following logical consequences:

- $\forall x.P(x) \lor \forall x.Q(x) \vDash \neg \exists x\, (\neg P(x) \land \neg Q(x))$
- $\vDash \exists x.(P(x) \lor Q(x)) \equiv \exists x.P(x) \lor \exists x.Q(x)$

# Some definition for tableaux

**Definition (Closed branch)**

A closed branch is a branch which contains a formula and its negation.

**Definition (Open branch)**

An open branch is a branch which is not closed

**Definition (Closed tableaux)**

A tableaux is closed if all its branches are closed.

**Definition**

Let $\varphi$ be a first-order formula and $\Gamma$ a finite set of such formulas. We write $\Gamma \vdash \varphi$ to say that there exists a closed tableau for $\Gamma \cup \{\neg\varphi\}$

# Soundness and completeness

**Theorem**

$$\Gamma \vdash \varphi \quad \Rightarrow \quad \Gamma \models \varphi$$

**Theorem**

$$\Gamma \models \varphi \quad \Rightarrow \quad \Gamma \vdash \varphi$$

**Remark**

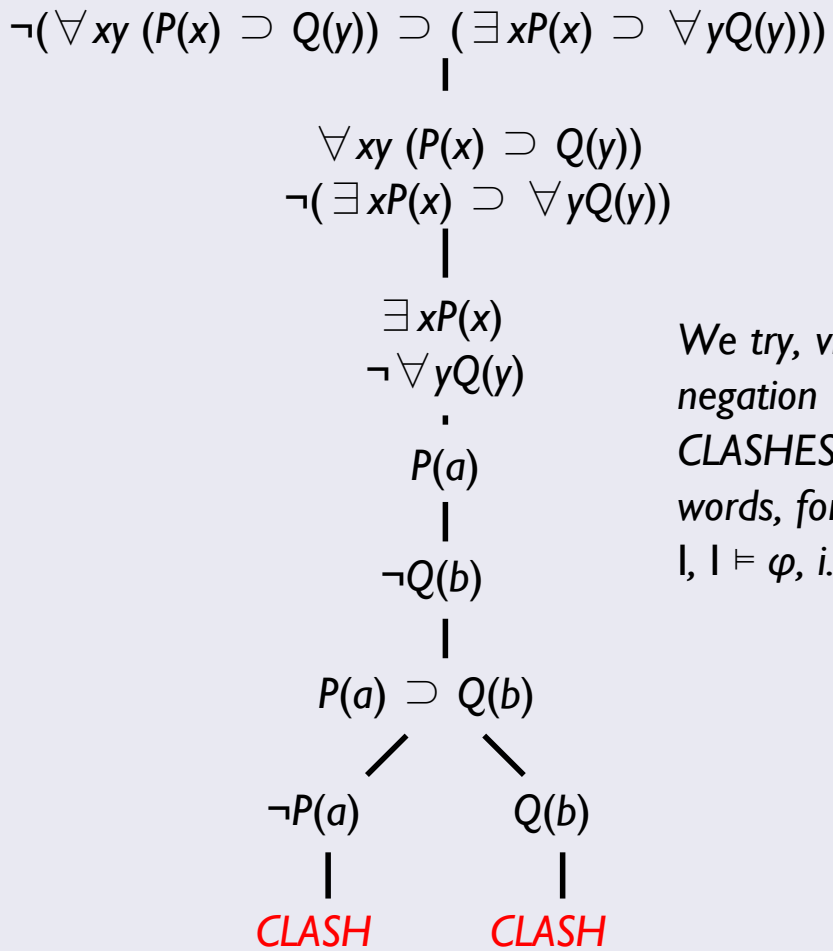
The mere existence of a closed tableau does not mean that we have an effective method to build it! Concretely: we don't know how often and in which way we have to apply ] the $\gamma$-rules ($\forall x\varphi(x) \Rightarrow \varphi[x/t]$), and what term to use in the substitution.

## Example

Check via tableaux if the validity/satisfiability of the formula

$\varphi = \forall x, y\,(P(x) \supset Q(y)) \supset (\exists x P(x) \supset \forall y Q(y))$

## Solution




*We try, with the tableaux, to build a model for the negation of φ. Since the tableaux ends with all CLASHES, there is no such a model. In other words, for all I, I ⊭ ¬φ. Which implies that for all I, I ⊨ φ, i.e., that φ is valid.*

# Infinite domains

- Differently from Prop. Logic, in FOL, models can be infinite.
- There are formulas which are satisfied only by infinite models. For instance the following formula[1]

$$\varphi = \left( \begin{array}{rl} \forall x\, \neg R(x, x) & \wedge \\ \forall xyz.(R(x, y) \wedge R(y, z) \supset R(x, z)) & \wedge \\ \forall x.\, \exists y.R(x, y) & \end{array} \right)$$

- If we build a tableaux for such a formula, searching for a model, we will end up in an infinite tableaux.

---

[1] To verify this, suppose that $I = \langle \Delta^I, \cdot^I \rangle$ is an interpretation that satisfies $\varphi$, and suppose that $|\Delta| = n$ for some finite number $n$. Consider the sequence $\langle d_1, d_2, d_3, \ldots d_{n+1} \rangle$ of $n + 1$ elements of $\Delta$, such that $\langle d_i, d_{i+1} \rangle \in R^I$. This sequence exists, because for every $d$ there is always a $d^i$ with $\langle d, d^i \rangle \in R^I$, since $I \models \forall x.\, \exists y.R(x, y)$. $I \models \forall xyz.(R(x, y) \wedge R(y, z) \supset R(x, z))$, implies that $R^I$ is transitive, and therefore for all $0 \leq i < j \leq n + 1$, $\langle d_i, d_j \rangle \in R^I$. The fact that $\Delta$ contains at most $n$ elements implies that for some $1 \leq i < j \leq n + 1$, $d_i = d_j$, which means that $\langle d_i, d_i \rangle \in R^I$ for some $1 \leq i \leq n$. But this contradicts the fact that $I \models \forall x\, \neg R(x, x)$.

# Infinite tableaux

## Exercize

Build a tableaux for

$\forall x\, \neg R(x, x) \wedge \forall xyz.(R(x, y) \wedge R(y, z) \supset R(x, z)) \wedge \forall x.\exists y.R(x, y)$

## Solution

$\forall x\, \neg R\,(x, x\,) \wedge \forall xyz.(R\,(x, y\,) \wedge R\,(y, z) \supset R\,(x, z)) \wedge \forall x.\exists y.R\,(x, y)$

|

$\forall x\, \neg R\,(x, x)$

$\forall xyz.(R\,(x, y\,) \wedge R\,(y, z) \supset R\,(x, z\,))$

$\forall x.\exists y.R\,(x, y)$

|

$\exists y.R\,(a_0, y)$

|

$R\,(a_0, a_1)$

|

$\exists y.R\,(a_1, y)$

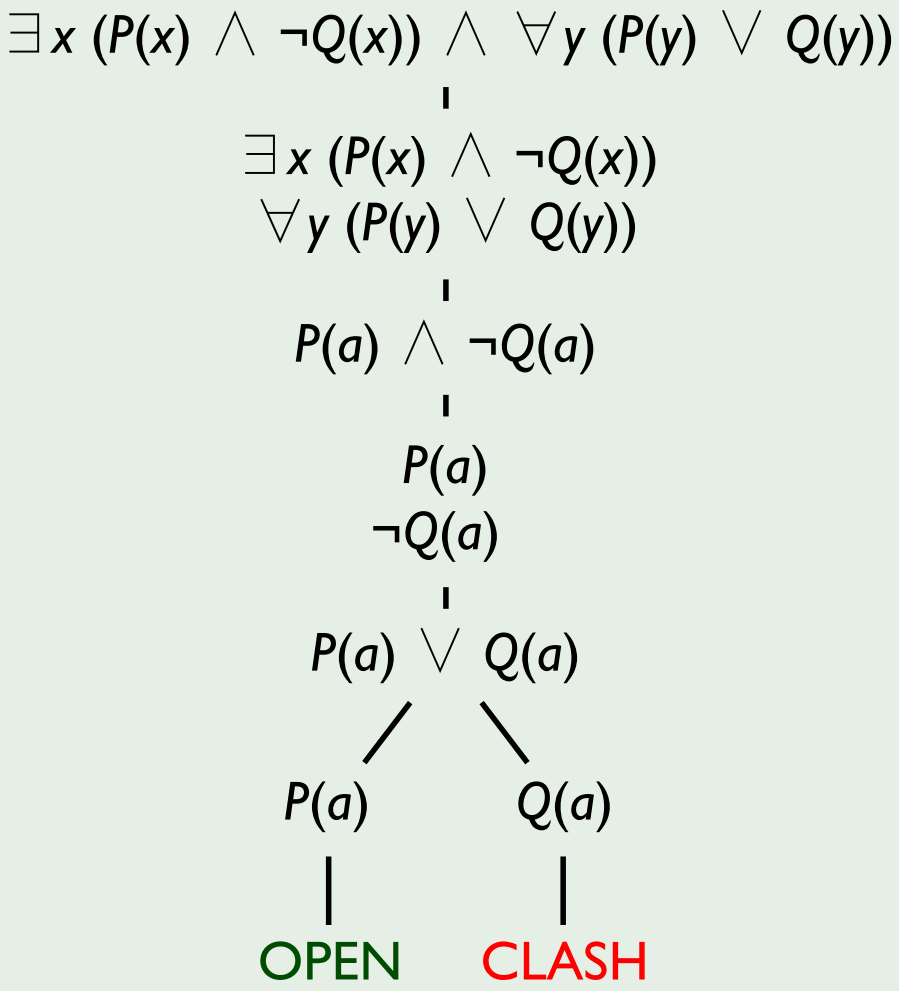
|

⋮

*By applying the γ-rule to the axiom* $\forall x\, \exists y\,(R\,(x, y\,))$, *we generate* $\exists y R(a_0, y\,)$ *for an initial constant* $a_0$, *and by applying the δ-rule to this last formula we generate a new individual* $a_1$
*This allow to apply the γ-rule again to* $\forall x\, \exists y R(x, y)$, *obtaining* $\exists y R(a_1, y)$, *and again by applying δ-rule to this new formula we generate another constant* $a_2$.
*The process can go on infinitively without reaching any clash*

# Example of tableaux

## Example




## Comments

From the formulas appearing in the OPEN branch of the tableaux it is possible to construct a model for the root formula.

- $\Delta = \{a\}$, the constants appearing in the formulas
- $I(P) = \{a\}$, since the formula $P(a)$ appears in the open branch
- $I(Q) = \{\}$ since the formula $\neg Q(a)$ appears in the open branch

# Termination fo FO tableauxs

In contrast to what happens in propositional logic, the tableau construction is not guaranteed to terminate.

If the formula $\varphi$ that labels the root is unsatisfiable, in which case the construction is guaranteed to terminate and the tableau is closed.

If the formula $\varphi$ that labels the root is satisfiable then either the construction is guaranteed to terminate and the tableau is open, or the construction does not terminate.

# Saturated Branches

## Saturated open

An open branch is called saturated if every non-literal has been analyzed at least once and, additionally, every $\gamma$-formula ($\gamma$-formulas are of the form $\forall x\varphi$ and $\neg\exists x\varphi$) has been instantiated with every term we can construct using the function symbols on the branch.

## Failing proof

A tableau with an open saturated branch can never be closed, i.e. we can stop an declare the proof a failure.

## Is this the solution?

This only helps us in special cases though.(A single 1-place function symbol together with a constant is already enough to construct infinitely many terms . . . )

# Countermodels

- If the construction of a tableaux ends in a saturated open branch, you can use it to help you define a model *M* for all the formulas on that branch.
- domain: set of all terms we can construct using the function symbols appearing on the branch (so-called Herbrand universe)

- terms are interpreted as themselves
- interpretation of predicate symbols: see literals on branch