Mathematical Logic
PL - Reasoning as deduction

Fausto Giunchiglia and Mattia Fumagalli*

University of Trento

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

Mathematical Logic

PL - Reasoning as deduction

Fausto Giunchiglia and Mattia Fumagalli*

University of Trento

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

Lecture index

1. Recap of basic notions
2. Reasoning as deduction
3. Hilbert systems (VAL – forward chaining)
4. Tableaux systems ((un)-SAT – backward chaining)

Tableaux

- Early work by Beth and Hintikka (around 1955). Later refined and popularised by Raymond Smullyan:
  - R.M. Smullyan. First-order Logic. Springer-Verlag, 1968.
- Modern expositions include:
  - M. Fitting. First-order Logic and Automated Theorem Proving. 2nd edition. Springer-Verlag, 1996.
  - M. DAgostino, D. Gabbay, R. H'ahnle, and J. Posegga (eds.). Handbook of Tableau Methods. Kluwer, 1999.
  - R. H'ahnle. Tableaux and Related Methods. In: A. Robinson and A. Voronkov (eds.), Handbook of Automated Reasoning, Elsevier Science and MIT Press, 2001.
  - Proceedings of the yearly Tableaux conference:

    http://i12www.ira.uka.de/TABLEAUX/

How does it work?

The tableau method is a method for proving, in a mechanical manner, that a given set of formulas is not satisfiable. In particular, this allows us to perform automated *deduction*:

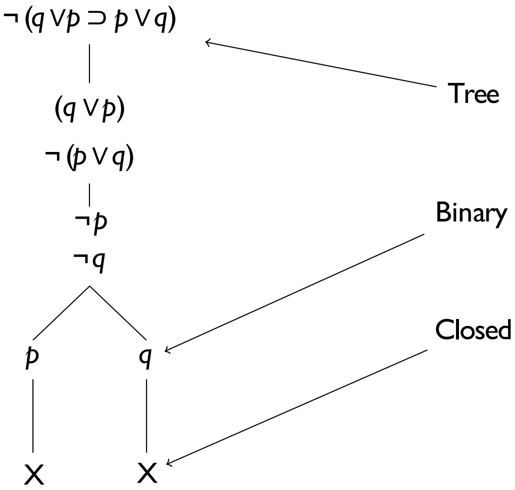
Given : set of premises Γ and conclusion φ

Task : prove $\Gamma \vDash \varphi$

How? show $\Gamma \cup \{\neg\varphi\}$ is not satisfiable (which is equivalent),
i.e. add the complement of the conclusion to the premises and derive a contradiction (refutation procedure)

See refutation theorem

An example



Constructing Tableau Proofs

- **Data structure**: a proof/ deduction is represented as a tableau - i.e., a binary tree - the nodes of which are labelled with formulas.
- **Start**: we start by putting the premises and the negated conclusion into the root of an otherwise empty tableau.
- **Expansion**: we apply expansion rules to the formulas on the tree, thereby adding new formulas and splitting branches. Compare with Hilbert calculus (forward vs backward chaining, axioms+theorems vs goal)
- **Closure**: we close branches that are obviously contradictory.

  Success: a proof is successful iff we can close all branches.

Expansion Rules of Propositional Tableau

α rules

$$\frac{\varphi \wedge \psi}{\begin{matrix}\varphi \\ \psi\end{matrix}} \qquad \frac{\neg(\varphi \vee \psi)}{\begin{matrix}\neg\varphi \\ \neg\psi\end{matrix}} \qquad \frac{\neg(\varphi \supset \psi)}{\begin{matrix}\varphi \\ \neg\psi\end{matrix}}$$

¬ ¬ -Elimination

$$\frac{\neg\neg\varphi}{\varphi}$$

β rules

$$\frac{\varphi \vee \psi}{\varphi \mid \psi} \qquad \frac{\neg(\varphi \wedge \psi)}{\neg\varphi \mid \neg\psi} \qquad \frac{\varphi \supset \psi}{\neg\varphi \mid \psi}$$

Branch Closure

$$\frac{\begin{matrix}\varphi \\ \neg\varphi\end{matrix}}{X}$$

Note: These are the standard ("Smullyan-style") tableau rules.

We omit the rules for ≡. We rewrite $\varphi \equiv \psi$ as $(\varphi \supset \psi) \wedge (\psi \supset \varphi)$

Smullyans Uniform Notation

Two types of formulas: conjunctive (α) and disjunctive (β):

α	α_1	α_2
$\varphi \wedge \psi$	φ	ψ
$\neg(\varphi \vee \psi)$	$\neg \varphi$	$\neg \psi$
$\neg(\varphi \supset \psi)$	φ	$\neg \psi$

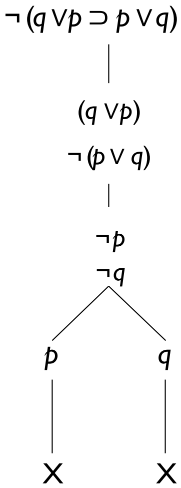
β	β_1	β_2
$\varphi \vee \psi$	φ	ψ
$\neg(\varphi \wedge \psi)$	$\neg \varphi$	$\neg \psi$
$\varphi \supset \psi$	$\neg \varphi$	ψ

We can now state α and β rules as follows:

$$\frac{\alpha}{\begin{array}{c}\alpha_1 \\ \alpha_2\end{array}} \qquad \frac{\beta}{\beta_1 \mid \beta_2}$$

Note: α rules are also called deterministic rules. β rules are also called splitting rules.

An example



Some definitions for tableaux

Definition (type-*alpha* and type-β formulae)

- Formulae of the form $\varphi \wedge \psi$, $\neg(\varphi \vee \psi)$, and $\neg(\varphi \supset \psi)$ are called type-α formulae.
- Formulae of the form $\varphi \vee \psi$, $\neg(\varphi \wedge \psi)$, and $\varphi \supset \psi$ are called type-β formulae

Note: type-*alpha* formulae are the ones where we use α rules. type-β formulae are the ones where we use β rules.

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 (Derivation $\Gamma \vdash \varphi$)

Let φ and Γ be a propositional formula and a finite set of propositional formulae, respectively. We write $\Gamma \vdash \varphi$ to say that there exists a closed tableau for $\Gamma \cup \{\neg \varphi\}$

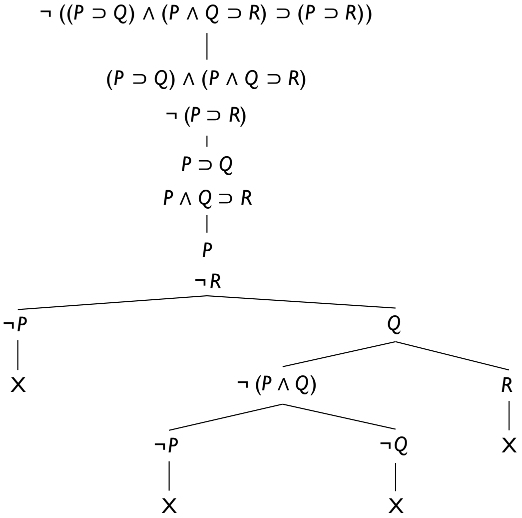
Tableaux and satisfiability

- A tableau for Γ attempts to build a propositional interpretation for Γ. If the tableaux is closed, it means that no model exist.
- We can use tableaux to check if a formula is satisfiable.

Exercise

Check whether the formula $\neg((P \supset Q) \wedge (P \wedge Q \supset R) \supset (P \supset R))$ is satisfiable

Solution



The tableau is closed and the formula is not satisfiable.

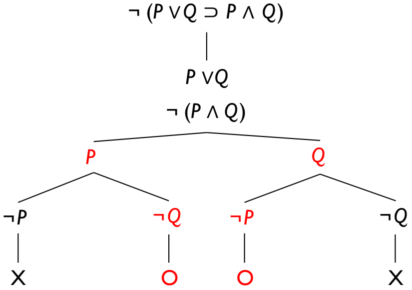
Using the tableau to build interpretations.

For each open branch in the tableau, and for each propositional atom p in the formula we define

$$I(p) = \begin{cases} \text{True} & \text{if } p \text{ belongs to the branch,} \\ \text{False} & \text{if } \neg p \text{ belongs to the branch.} \end{cases}$$

If neither p nor $\neg p$ belong to the branch we can define $I(p)$ in an arbitrary way.

Models for $\neg (P \vee Q \supset P \wedge Q)$



Two models:

- $I(P)$ = True, $I(Q)$ = False
- $I(P)$ = False, $I(Q)$ = True

Double-check with the truth tables!

P	Q	$P \vee Q$	$P \wedge Q$	$P \vee Q \supset P \wedge Q$	$\neg (P \vee Q \supset P \wedge Q)$
T	T	T	T	T	F
F	F	F	F	T	F
T	F	T	F	F	T
F	T	T	F	F	T

Double-check with the truth tables!

P	Q	$P \vee Q$	$P \wedge Q$	$P \vee Q \supset P \wedge Q$	$\neg (P \vee Q \supset P \wedge Q)$
T	T	T	T	T	F
F	F	F	F	T	F
T	F	T	F	F	T
F	T	T	F	F	T

Termination

Assuming we analyze each formula at most once, we have:

Theorem (Termination)

For any propositional tableau, after a finite number of steps no more expansion rules will be applicable.

Hint for proof: This must be so, because each rule results in ever shorter formulas.

Note: Importantly, termination will *not* hold in the first-order case.

Soundness and Completeness

To actually believe that the tableau method is a valid decision procedure we have to prove:

Theorem (Soundness)

If $\Gamma \vdash \varphi$ *then* $\Gamma \models \varphi$

Theorem (Completeness)

If $\Gamma \models \varphi$ *then* $\Gamma \vdash \varphi$

Remember: We write $\Gamma \vdash \varphi$ to say that there exists a closed tableau for $\Gamma \cup \{\neg \varphi\}$.

Hint: tableau builds a branch for any possible truth assignment, and vice versa, compare with truth tables

A last definition - Fairness

Definition (Fairness)

We call a propositional tableau fair if every non-literal of a branch gets eventually analysed on this branch.

Decidability

The proof of Soundness and Completeness confirms the decidability of propositional logic:

Theorem (Decidability)

The tableau method is a decision procedure for classical propositional logic.

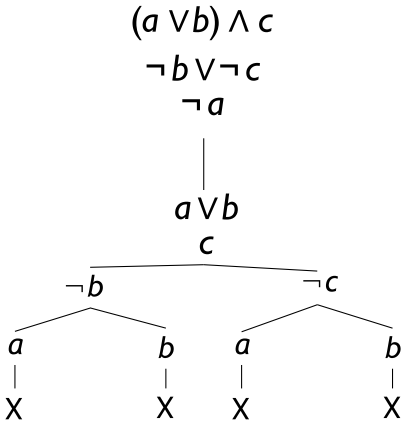
Proof. To check validity of φ, develop a tableau for $\neg\varphi$. Because of termination, we will eventually get a tableau that is either (1) closed or (2) that has a branch that cannot be closed.

- In case (1), the formula φ must be valid (soundness).
- In case (2), the branch that cannot be closed shows that $\neg\varphi$ is satisfiable (see completeness proof), i.e. φ cannot be valid.

This terminates the proof.

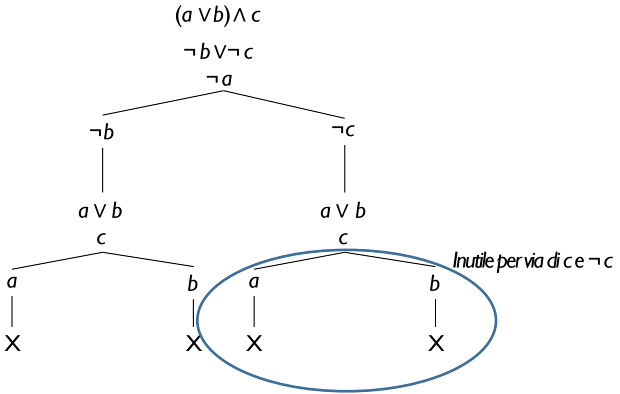
Exercise

Build a tableau for $\{(a \lor b) \land c, \neg b \lor \neg c, \neg a\}$



Another solution

What happens if we first expand the disjunction and then the conjunction?



Expanding β rules creates new branches. Then α rules may need to be expanded in all of them.

Strategies of expansion

- Using the "wrong" policy (e.g., expanding disjunctions first) leads to an increase of *size* of the tableau, which leads to an increase of *time*;
- yet, unsatisfiability is still proved if set is unsatisfiable;
- this is not the case for other logics, where applying the wrong policy may inhibit proving unsatisfiability of some unsatisfiable sets.

Finding Short Proofs

- It is an open problem to find an efficient algorithm to decide in all cases which rule to use next in order to derive the shortest possible proof.
- However, as a rough guideline always apply any applicable *non-branching rules* first. In some cases, these may turn out to be redundant, but they will never cause an exponential blow-up of the proof.

Efficiency

- Are analytic tableaus an efficient method of checking whether a formula is a tautology?
- Remember: using the truth-tables to check a formula involving *n* propositional atoms requires filling in 2^n rows (exponential = very bad).
- Are tableaux any better?
- In the worst case no, but if we are lucky we may skip some of the 2^n rows !!!

Mathematical Logic
Reasoning as deduction

Fausto Giunchiglia and Mattia Fumagalli

University of Trento

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