Foundations of Computer Science
Logic, models, and computations

Chapter: Proofs

Course CSC_41012_EP

of l'Ecole Polytechnique

Olivier Bournez

bournez@lix.polytechnique.fr

Version of August 20, 2024

Proofs

The objective of this chapter is to start to address the fundamental following question: what is a *demonstration* (i.e. a (mathematical) *proof*).

To to this, more precisely, we will focus on this chapter on the following problem: Given some *propositional formula* F, how to decide if F is a *tautology*? A tautology is also called a *theorem*.

This will lead us to describe some particular *algorithms*.

1 Introduction

A first method to solve this problem is the one that we have used in the previous chapter: If F is of the form $F(p_1, \cdots, p_n)$, we can test for each of the 2^n valuations v, i.e. for the 2^n functions from $\{1,2,\ldots,n\}$ to $\{0,1\}$, if v is indeed a *model* of F. If this is the case, then F is a tautology. In any other case, F is not a tautology. It is easy to program such a method in your favorite programming language.

The good news is that this method exists: The problem to determine if a given formula is a tautology is *decidable*, using the terminology that we will see in the next chapters.

Remark 1 *This observation can seem strange, and, in some sense, to expect little, but we will see that when we consider more general logic, even simple logic, this becomes problematic: There does not always exist some algorithm to determine if a given formula F is a tautology.*

However, this method is particularly inefficient. It has the main inconvenient to guarantee that when F is a tautology, we will do 2^n times a similar test of type "is the valuation v a model of F?". When n is big, 2^n explodes very quickly: If this method can indeed be programmed, it is in practise useless, since it takes a huge time, as soon as one considers some formulas F with a high number of variables.

Let's then come back to our problem: One can say that in the classical reasoning in mathematics, the usual method to prove that some assertion is a theorem is to *prove* it.

If one wants to do better than the previous exhaustive method, there are two angles of attacks. The first angle of attack is to try to come close to the notion of *demonstration* in the usual reasoning: Proof methods in the spirit of the coming

sections will appear. The second angle of attack is to try to produce algorithms as efficient as possible: Methods such as *proof!by resolution* method or *tableau method* then appear.

In general, one expects that a proof method is always *valid*: It produces only correct deductions. In any case, the question of the *completeness* of the proof method makes sense: Can all the theorems (tautology) be proved using this proof method?

We will see in what follows, four deductions systems that are valid and complete: The proofs *à la Hilbert*, the *natural deduction*, the *resolution method* and the *tableau method*. We will prove the validity and the completeness only for the tableau method. Every time, we will denote by $\vdash$ the underlying notion of proof: $T \vdash F$ means that the formula F can be proved starting from a set of propositional formulas T. We write $\vdash T$ if $\emptyset \vdash T$.

At first sight, one needs a different symbol $\vdash$ for each notion of demonstration.

However, the validity and completeness theorem that follow will prove that, every time, what is provable for each notion of demonstration is exactly the same, that is to say the tautologies of the propositional calculus.

In summary, the symbol $\models$ and the symbol $\vdash$ denotes exactly the same notion: For each of the variants of $\vdash$ mentioned in what follows, we have $\vdash F$ if and only if $\models F$, that is to say if and only if F is a tautology.

2 Proofs à la Frege and Hilbert

In this deduction system, we start from a set of axioms from propositional logic, that are tautologies, and we use a unique *deduction rule*, the *modus ponens*, also called *cut rule* that aims to capture a very usual type of reasoning in mathematics.

The modus ponens states that from a formula F and from a formula $F \Rightarrow G$, we deduce G.

Graphically:

$$\frac{F \quad (F \Rightarrow G)}{G}$$

Example 1 *For example, starting from* $(A \wedge B)$ *and from* $(A \wedge B) \Rightarrow C$ *we deduce* C.

We consider then a set of *axioms*, that are actually some instances of a finite number of axioms.

Definition 1 (Instance) *A formula F is said to be an instance of a formula G if F is obtained by substituting certain propositional variables in G by some formulas F_i.*

Example 2 *The formula* $((C \Rightarrow D) \Rightarrow (\neg A \Rightarrow (C \Rightarrow D)))$ *is an instance of* $(A \Rightarrow (B \Rightarrow A))$, *by taking* $(C \Rightarrow D)$ *for* A, *and* $\neg A$ *for* B.

Definition 2 (Axioms of boolean logic) *An* axiom of Boolean logic *is any instance of the following formulas:*

1. $(X_1 \Rightarrow (X_2 \Rightarrow X_1))$ *(axiom 1 for the implication);*
2. $((X_1 \Rightarrow (X_2 \Rightarrow X_3)) \Rightarrow ((X_1 \Rightarrow X_2) \Rightarrow (X_1 \Rightarrow X_3)))$ *(axiom 2 for the implication);*
3. $(X_1 \Rightarrow \neg\neg X_1)$ *(axiom 1 for the negation);*
4. $(\neg\neg X_1 \Rightarrow X_1)$ *(axiom 2 for the negation);*
5. $((X_1 \Rightarrow X_2) \Rightarrow (\neg X_2 \Rightarrow \neg X_1))$ *(axiom 3 for the negation);*
6. $(X_1 \Rightarrow (X_2 \Rightarrow (X_1 \wedge X_2)))$ *(axiom 1 for the conjunction);*
7. $((X_1 \wedge X_2) \Rightarrow X_1)$ *(axiom 2 for the conjunction);*
8. $((X_1 \wedge X_2) \Rightarrow X_2)$ *(axiom 3 for the conjunction);*
9. $(X_1 \Rightarrow (X_1 \vee X_2))$ *(axiom 1 for the disjunction);*
10. $(X_2 \Rightarrow (X_1 \vee X_2))$ *(axiom 2 for the disjunction);*
11. $((((X_1 \vee X_2) \wedge (X_1 \Rightarrow C)) \wedge (X_2 \Rightarrow C)) \Rightarrow C)$ *(axiom 3 for the disjunction).*

We obtain a notion of demonstration.

Definition 3 (Demonstration by modus ponens) *Let T be a set of propositional formulas, and F be some propositional formula. A* proof (by modus ponens) of F from T *is a finite sequence $F_1, F_2, \cdots, F_n$ of propositional formulas such that F_n is equal to F, and for all i, either F_i is in T, or F_i is some axiom of Boolean logic, or F_i is obtained by modus ponens from two formulas F_j, F_k with $j < i$ and $k < i$.*

We write $T \vdash F$ if F is provable (by modus ponens) from T. We write $\vdash F$ if $\emptyset \vdash F$, and we say that F is *provable (by modus ponens)*

Example 3 *Let F, G, H three propositional formulas. Here is a proof of $(F \Rightarrow H)$ from $\{(F \Rightarrow G), (G \Rightarrow H)\}$:*

- $F_1 : (G \Rightarrow H)$ *(hypothesis);*
- $F_2 : ((G \Rightarrow H) \Rightarrow (F \Rightarrow (G \Rightarrow H)))$ *(instance of axiom 1.);*
- $F_3 : (F \Rightarrow (G \Rightarrow H))$ *(modus ponens from F_1 and F_2);*
- $F_4 : ((F \Rightarrow (G \Rightarrow H)) \Rightarrow ((F \Rightarrow G) \Rightarrow (F \Rightarrow H)))$ *(instance of axiom 2.);*
- $F_5 : ((F \Rightarrow G) \Rightarrow (F \Rightarrow H))$ *(modus ponens from F_3 and F_4);*

- $F_6 : (F \Rightarrow G)$ *(hypothesis);*
- $F_7 : (F \Rightarrow H)$ *(modus ponens from F_6 and F_5).*

Exercise 1 *(solution on page 206) Prove $(F \Rightarrow F)$.*

In the following exercises, you can use the previous exercises to solve each of the questions.

Exercise 2 *(solution on page 206) [Deduction theorem] Let T be a family of propositional formulas, and let F and G be two propositional formulas. Prove that $T \vdash F \Rightarrow G$ is equivalent to $T \cup \{F\} \vdash G$.*

Exercise 3 *(solution on page 207) Prove the following assertions:*

- *$T \cup \{F\} \vdash G$ is equivalent to $T \cup \{\neg G\} \vdash \neg F$.*
- *If we have both $T \vdash F$ and $T \vdash \neg F$, then we have $T \vdash G$ for any formula G.*

Exercise 4 *(solution on page 207) Prove that $\{(\neg G \Rightarrow G)\} \vdash G$, for any formula G.*

Exercise 5 *(solution on page 207) Prove that if we have both $T \cup \{F\} \vdash G$ and $T \cup \{\neg F\} \vdash G$ then we have $T \vdash G$.*

Exercise 6 *Prove the following assertions:*

- $\{F\} \vdash \neg\neg F$
- $\{F, G\} \vdash F \vee G$
- $\{\neg F\} \vdash \neg(F \wedge G)$
- $\{\neg G\} \vdash \neg(F \wedge G)$
- $\{F\} \vdash F \vee G$
- $\{G\} \vdash F \vee G$
- $\{\neg F, \neg G\} \vdash \neg(F \vee G)$
- $\{\neg F\} \vdash (F \Rightarrow G)$
- $\{G\} \vdash (F \Rightarrow G)$
- $\{F, \neg G\} \vdash \neg(F \Rightarrow G)$

Exercise 7 *For v some partial function from $\{X_i\}$ in $\{0,1\}$, we set*

$$T_V = \{X_i | v(X_i) = 1\} \cup \{\neg X_i | v(X_i) = 0\}.$$

Prove that any formula H whose variables are among the domain of V, the relation $v \models H$ implies $T_V \vdash H$ and the relation $v \not\models H$ implies $T_V \vdash \neg H$.

This proof method is valid: By checking that all the axioms are tautologies, it is easy to get convinced by recurrence on the length of a proof that the following results are true.

Theorem 1 (Validity) *Every provable propositional formula is a tautology.*

What is less trivial, and more interesting is the converse: Any tautology has a proof of this type.

Theorem 2 (Completeness) *Every tautology is provable (by modus ponens).*

We will not do the proof of this result here, but this corresponds to the following exercise:

Exercise 8 *Prove this result by using the previous exercises: The key is the possibility to reason by cases (Exercise 5) and the Exercise 7 which make the relation between semantic and syntax.*

We have just described a deduction system that is very closed to the usual notion of proof in mathematics. However, this system is not easily exploitable to build an algorithm that would determine if a given formula F is a tautology.

This is easy to be convinced of that by trying to do the previous exercises, and by observing how hard it is to find a proof using this method.

3 Demonstrations by natural deduction

3.1 Rules from natural deduction

The previous notion of demonstration is in practise hard to use. Indeed, in the previous system, we are, somehow, constrained to keep the hypotheses during all the demonstration. We can not easily express some however common reasoning. We want to prove that $A \Rightarrow B$, supposing A and proving B under this hypothesis. This remarks leads to introduce a notion of couple made of a finite set of hypotheses and a conclusion. Such a couple is called a *sequent*.

We consider in this section that the propositional formulas also include $\bot$, interpreted by false, and $\top$ interpreted by true.

Definition 4 (Sequent) *A* sequent *is a couple* $\Gamma \vdash A$*, where* Γ *is a finite set of propositional formulas, and* A *is a propositional formula.*

The deduction rules of natural deduction are then the following:

$$\frac{}{\Gamma \vdash A}\ \text{axiom for each } A \in \Gamma$$

$$\frac{}{\Gamma \vdash \top}\ \top\text{-intro}$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \bot\text{-elim}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-intro}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-elim}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\text{-elim}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee\text{-intro}$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee\text{-intro}$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ \vee\text{-elim}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \Rightarrow\text{-intro}$$
$$\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \Rightarrow\text{-elim}$$
$$\frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A} \neg\text{-intro}$$
$$\frac{\Gamma \vdash A \quad \Gamma \vdash \neg A}{\Gamma \vdash \bot} \neg\text{-elim}$$
$$\frac{}{\Gamma \vdash A \vee \neg A} \text{ exclusive middle}$$

The rules $\top$-intro, $\wedge$-intro, $\vee$-intro, $\Rightarrow$-intro, $\neg$-intro, $\forall$-intro and $\exists$-intro are termed *introduction rules* and the rules $\bot$-elim, $\wedge$-elim, $\vee$-elim, $\Rightarrow$-elim, $\neg$-elim, $\forall$-elimand $\exists$-elimare termed *elimination rules*. The rules of natural deduction are hence classified in four groups: the introduction rules, the elimination rules, the *axiom* rule and the *exclusive middle* rule.

A *demonstration* of a sequent $\Gamma \vdash A$ is a derivation of this sequent, that is to say a tree whose nodes are labeled by a sequent, whose root is labeled by $\Gamma \vdash A$, and such that if a node is labeled by some sequent $\Delta \vdash B$, then its children are labeled by sequent $\Sigma_1 \vdash C_1, \ldots, \Sigma_n \vdash C_n$ such that there exists some natural deduction rule, that permits to deduce $\Delta \vdash B$ of $\Sigma_1 \vdash C_1, \ldots, \Sigma_n \vdash C_n$.

A sequent $\Gamma \vdash A$ is hence provable if there exists some demonstration of this sequent.

3.2 Validity and completeness

We can prove the following results:

Theorem 3 (Validity) *For any set of propositional formulas Γ and for any propositional formula A, if $\Gamma \vdash A$ is provable, then A is* a consequence *of Γ.*

Theorem 4 (Completeness) *Let Γ be any set of propositional formulas. Let A be some propositional formula that is* a consequence *of Γ.*
Then $\Gamma \vdash A$ is provable.

4 Proofs by resolution

We present briefly the notion of proof by resolution. This proof methods is maybe less natural, but is simpler to be implemented on a computer.

The resolution applies to a formula in conjunctive normal form. Since any propositional formula can be put in an equivalent conjunctive normal form, this is not restrictive.

Remark 2 *At least in appearance. Indeed, it requires to be more clever than in the previous chapter to transform a propositional formula in conjunctive normal form, if we want to avoid the problem of the explosion of the size of the*

formulas, and if we want to implement efficiently the method.

We call *clause* a disjunction of literals. Remember that a *literal* is a propositional variable or its negation. We represent a clause c by the set of the literals on which the disjunction applies.

Example 4 *We hence write* $\{p, \neg q, r\}$ *instead of* $p \vee \neg q \vee r$.

Given some literal u, we write $\overline{u}$ for the literal equivalent to $\neg u$: In other words, if u is the propositional variable p, $\overline{u}$ values $\neg p$, and if u is $\neg p$, $\overline{u}$ is p. Finally, we introduce some empty clause, denoted by $\square$, whose value is 0 for any valuation.

Definition 5 (Resolvant) *Let* C_1, C_2 *be two clauses. The clause* C *is a resolvent of* C_1 *and* C_2 *if there exists some literal* u *such that:*

- $u \in C_1$;
- $\overline{u} \in C_2$;
- C *is given by* $(C_1 \setminus \{u\}) \cup (C_2 \setminus \{\overline{u}\})$.

Example 5 *The clauses* $\{p, q, r\}$ *and* $\{\neg r, s\}$ *have* $\{p, q, s\}$ *as a resolvent.*

Example 6 *The clauses* $\{p, q\}$ *and* $\{\neg p, \neg q\}$ *have two resolvents, namely* $\{q, \neg q\}$ *and* $\{p, \neg p\}$. *The clauses* $\{p\}$ *and* $\{\neg p\}$ *have the resolvent* $\square$.

This provides a notion of demonstration:

Definition 6 (Proof by resolution) *Let* T *be a set of clauses. A* proof by resolution *of* T *is a finite sequence* $F_1, F_2, \cdots, F_n$ *of clauses such that* F_n *is equal to* $\square$, *and for every* i, *either* F_i *is a clause in* T, *or* F_i *is a resolvent of two clauses* F_j, F_k *with* $j < i$ *and* $k < i$.

Remark 3 *The modus ponens, at the heart of the previous Hilbert-Frege proof systems, consists in stating that from a formula* F *and a formula* $(F \Rightarrow G)$, *we deduce* G. *If we consider that the formula* $(F \Rightarrow G)$ *is equivalent to the formula* $(\neg F \vee G)$, *the modus ponens can also be seen as stating that from a formula* F *and a formula* $(\neg F \vee G)$, *we deduce* G, *which reads similar to the concept of resolvent. The resolvent of* $\{f\}$ *and of* $\{\neg f, g\}$ *is* $\{g\}$.

In some way, the resolvent is some generalized modus ponens, even if this analogy is only an analogy, and if a proof in a given proof system can not be translated directly into another.

Exercise 9 *Prove by resolution*

$$T = \{\{\neg p, \neg q, r\}, \{\neg p, \neg q, s\}, \{p\}, \{\neg s\}, \{q\}, \{t\}\}.$$

This proof method is valid (easy direction).

Theorem 5 (Validity) *Every clause that appears in a resolution proof of t is a consequence of T.*

Actually, to prove a formula, one reasons in general in this proof method on its negations, and one searches to prove that the negation is contradictory with the hypotheses. The validity is in general expressed in this way:

Corollary 1 (Validity) *If a set of clauses T admits some resolution proof, then T is contradictory.*

It turns out to be complete (harder direction).

Theorem 6 (Completeness) *Let T be a set of contradictory clauses. It admits some proof by resolution.*

5 Proofs by tableau method

We have considered up to now some valid and complete proofs systems, without providing a proof of our theorems. We will study more completely the tableau method. We have chosen to develop this method, since it is very algorithmic, and based on the notion of tree. This will contribute to our recurring argumentation that the notion of tree is everywhere in computer science.

5.1 Principle

We can first consider that our formulas are written using only the connectors $\neg, \wedge, \vee, \Rightarrow$, since the formula $(F \Leftrightarrow G)$ can be considered as an abbreviation of formula $((F \Rightarrow G) \wedge (G \Rightarrow F))$.

Suppose that we want to prove that F is a tautology. If the formula F is of the form $(F_1 \wedge F_2)$, then one can try to prove F_1 and to prove F_2. If the formula F is of the form $(F_1 \vee F_2)$, we write F_1, F_2, and we will explore two possibilities, one for the case F_1, and one for the case F_2.

We will basically bring all the possibilities to these two configurations: If the formula F is of the form $(F_1 \Rightarrow F_2)$, using the fact that it can be considered as $(F_2 \vee \neg F_1)$, we will use the rule of $\vee$, and if F is of the form $\neg(F_1 \Rightarrow F_2)$, using that it can be considered as $(F_1 \wedge \neg F_2)$, we will use the rule of $\wedge$. All other cases can be dealt similarly using de Morgan's laws.

Doing so systematically, we will build a tree, whose root is labeled by the negation of the formula F. In other words, to prove a formula F, the method starts from the negation of formula F.

Let's do it on the example of the following formula

$$(((p \wedge q) \Rightarrow r) \Rightarrow ((p \Rightarrow r) \vee (q \Rightarrow r))).$$

that we want to prove.

We start from $\neg F$, that is to say:

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

- by transforming the implication $\neg(F_1 \Rightarrow F_2)$ into equivalent formula $(F_1 \wedge \neg F_2)$, we obtain $((p \wedge q) \Rightarrow r) \wedge \neg((p \Rightarrow r) \vee (q \Rightarrow r))$ we get to the rule of $\wedge$.

- We then apply the rule of $\wedge$: We consider the formulas $((p \wedge q) \Rightarrow r)$ and $\neg((p \Rightarrow r) \vee (q \Rightarrow r))$.

- Let's consider the latter formula. From de Morgan's law, it can be considered as $(\neg(p \Rightarrow r) \wedge \neg(q \Rightarrow r))$: We can then associate to this formula $\neg(p \Rightarrow r)$ and $\neg(q \Rightarrow r)$ by the $\wedge$ rule.

- We consider then $\neg(p \Rightarrow r)$, and we get the formulas p and $\neg r$.

- We then obtain q and $\neg r$ from $\neg(q \Rightarrow r)$.

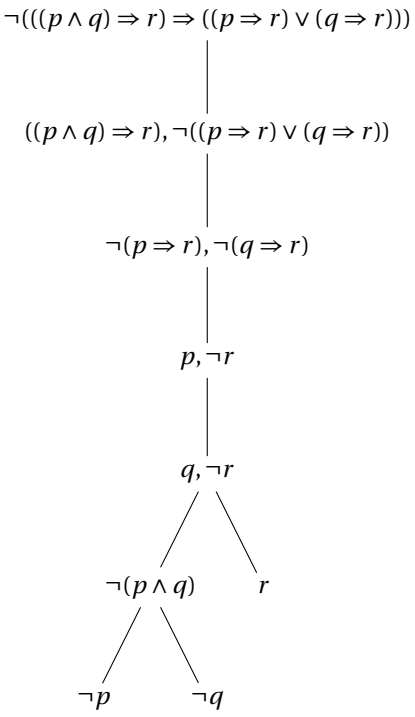
- We consider now $((p \wedge q) \Rightarrow r)$, that can be seen as $(r \vee \neg(p \wedge q))$. Thanks to the $\vee$ rule, we have the choice between the formula $\neg(p \wedge q)$ or r.

- The case of r is excluded by the previous step, where we had $\neg r$.

- In the first case, we still have the choice between $\neg p$ or $\neg q$. The two cases are excluded, since we had before p and q.

Since all branches lead to a contradiction, there is no possibility in which F could be false. Consequently, we deduce that F is a tautology.

The computation we have just done is naturally corresponding to the following tree:

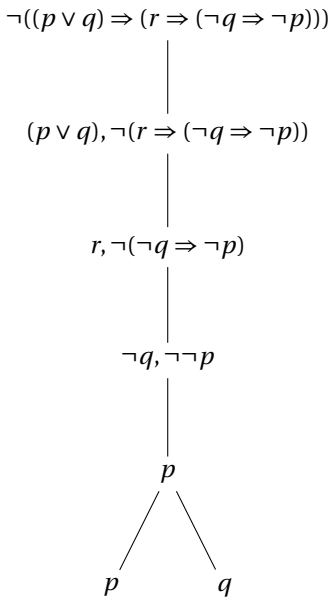


Each branch is a possible scenario. If a branch has a node labeled by some formula A such that $\neg A$ is appearing on the same branch (or their respective opposites), then one stops to develop this branch, and the branch is said to be *closed*: This means that we know that a contradiction is reached. If all the branches are closed, then we say that the *tree is closed*, and we are sure that all the possible scenarios are excluded.

Consider now the example of the formula G given by

$$((p \vee q) \Rightarrow (r \Rightarrow (\neg q \Rightarrow \neg p))).$$

With the same method, we build a tree whose root is $\neg G$.



On this example, we obtain a tree with two branches. The right branch is closed. The left branch is not: The propositional variables on this branch are $r, \neg q$ and p. By taking the valuation v with $v(r) = 1,\ v(q) = 0,\ v(p) = 1$, this provides the value 1 to $\neg G$, and hence 0 to G. In other words, we know that G is not a tautology. We say that the tree is *open. open* .

5.2 Description of the method

Let's now formalize the method. A *tableau* is a binary tree whose nodes are labeled by sets of propositional formulas, and that is built recursively from its root, vertices after vertices, by using two types of rules: The α rules and the β rules.

Remember that, in order to simplify the discussion, we have considered that the propositional formulas are written using only the connectors $\neg, \wedge, \vee, \Rightarrow$.

Formulas are divided in two groups, the α-group, and the β-group. To each formula, we associate inductively two new formulas according to the following rules:

- The formulas of the following form are α-formula:
  1. $\alpha = (A \wedge B)$. To such a formula is associated $\alpha_1 = A$ and $\alpha_2 = B$.
  2. $\alpha = \neg(A \vee B)$. To such a formula is associated $\alpha_1 = \neg A$ and $\alpha_2 = \neg B$.
  3. $\alpha = \neg(A \Rightarrow B)$. To such a formula is associated $\alpha_1 = A$ and $\alpha_2 = \neg B$.
  4. $\neg\neg A$: To such a formula is associated $\alpha_1 = \alpha_2 = A$.
- The formulas of the following form are β-formulas:
  1. $\beta = \neg(A \wedge B)$. To such a formula is associated $\beta_1 = \neg A$, $\beta_2 = \neg B$.

2. $\beta = (A \vee B)$. To such a formula is associated $\beta_1 = A$, $\beta_2 = B$.
3. $\beta = (A \Rightarrow B)$. To such a formula is associated $\beta_1 = \neg A$, $\beta_2 = B$.

If B a branch of a tableau, we denote by $\bigcup B$ the set of the formulas that appear on a vertex of B.

The two recursive rules to construct a tableau are the following:

1. An α rule consists in extending a finite branch of tableau T by the vertex labeled $\{\alpha_1, \alpha_2\}$, where α is some α-formula that appears on a vertex of B.
2. A β rule consists in extending a finite branch of a tableau T by two sons labeled respectively by $\{\beta_1\}$ and $\{\beta_2\}$, where β is some β-formula that appears in some vertex of B.

Remark 4 *Observe that this is not necessarily the last vertex of a branch B that is developed at each step, but a formula somewhere on the branch.*

A branch B is said to be *closed* if there exists some formula A such that A and $\neg A$ appears on the branch B. In the opposite case, the branch is said to be *open*.

A branch B is *developed* if

1. for any α-formula of $\bigcup B$, $\alpha_1 \in \bigcup B$ and $\alpha_2 \in \bigcup B$.
2. for any β-formula of $\bigcup B$, $\beta_1 \in \bigcup B$ or $\beta_2 \in \bigcup B$.

A tableau is said to be *developed* if all its branches are either closed or developed. A tableau is said to be *closed* if all its branches are closed. A tableau is said to be *open*open tree in tableau method if it has some open branch.

Finally, a *tableau* for a formula A (respectively for a set of formulas Σ) is a tableau whose root is labeled by $\{A\}$ (respectively by $\{A | A \in \Sigma\}$).

5.3 Termination of the method

First, observe that it is always possible to apply some α or β rules until a developed tableau is reached.

Proposition 1 *If Σ is a finite set of formulas, then there is some (finite) developed tableau for Σ.*

Proof: This is proved by recurrence on the number n of elements of Σ.

For the case $n = 1$, observe that the length of the formulas α_1, α_2, β_1, and β_2 is always strictly less than the length of α and β. The process of extension of branches that are not closed hence eventually terminates after finitely many steps. The array that is obtained a the end is developed, since otherwise it would have some extension.

For the case $n > 1$, we can write $\Sigma = \{F_1, \cdots, F_n\}$. Consider by recurrence hypothesis a developed tableau for $\Sigma = \{F_1, \cdots, F_{n-1}\}$. If this tableau is closed or if F_n is some

propositional variable, then this tableau is a developed tableau for Σ. Otherwise, we can extend all the open branches by applying all the rules corresponding to formula F_n, and by developing all the obtained branches. The process is terminating for the same reason as for the case $n = 1$. □

Remark 5 *Of course, from a given root, there are many ways to build some developed tableau.*

5.4 Validity

The previous method provides a proof method.

Definition 7 *A formula F said to be* provable by tableau *if there exists some closed tableau with the root $\{\neg F\}$. We then write $\vdash F$ when this holds.*

Exercise 10 *Prove that A is a consequence of $((A \vee \neg B) \wedge B)$ by the tableau method, i.e. $\vdash (((A \vee \neg B) \wedge B) \Rightarrow A)$.*

Exercise 11 *Prove that $\neg C$ is a consequence of $((H \wedge (P \vee C)) \Rightarrow A) \wedge H \wedge \neg A \wedge \neg P)$ by the tableau method.*

The method is valid.

Theorem 7 (Validity) *Any provable formula is a tautology.*

Proof: We will say that a branch B of a tableau is *realizable* if there exists some valuation v such that $v(A) = 1$ for any formula $A \in \bigcup B$ and $v(A) = 0$ if $\neg A \in \bigcup B$. A tableau is said to be *realizable* if it has some realizable branch.

We just need to prove the following result.

Lemma 1 *Let T' be some immediate extension of the tableau T: That is to say the tableau obtained by applying either an α or a β-rule to T. If T is realizable, then so does T'.*

This lemmas is sufficient to prove the theorem. Indeed, if F is provable, then there is some closed tableau whose root is $\neg F$. That means that in every branch, there is a formula A such that A and $\neg A$ appear on this branch, and hence none of the branches of T is realizable. By this lemma, this means that we started from a tree reduced to $\neg F$ that was not realizable. In other words, that F is a tautology.

It remains to prove the lemma. Let B be some realizable branch of T, and let B' the branch of T that is extended in T'. If $B \neq B'$, then B remains a realizable branch of T. If $B = B'$, then B is extended in T',

1. either in a branch B_α by some α-rule;
2. or by two branches B_{β_1} and B_{β_2} by some β-rule.

In the first case, let α be the formula used by the rule, and let v be a valuation that realizes B: From $v(\alpha) = 1$, we deduce that $v(\alpha_1) = 1$ and $v(\alpha_2) = 1$. Hence v is a valuation realizing B_α and the tableau T' is realizable.

In the second case, let β be the formula used by the β-rule. From $v(\beta) = 1$, we deduce that at least one of the values $v(\beta_1)$ and $v(\beta_2)$ values 1. Hence v realizes one of the branches B_{β_1} and B_{β_2}, and the tableau T' is realizable. □

5.5 Completeness

The method is complete. In other words, the converse of the previous theorem is true.

Theorem 8 (Completeness) *Every tautology is provable.*

Corollary 2 *Let F be some propositional formula.*
F is a tautology if and only if F is provable.

The rest of this subsection is devoted to prove this theorem.

Observe first that if B is a branch that is both developed and open in some tableau T, then then set $\bigcup B$ of the formulas that appear in B have the following properties:

1. there is no propositional variable such that $p \in \bigcup B$ and such that $\neg p \in \bigcup B$;
2. for every α-formula $\alpha \in \bigcup B$, $\alpha_1 \in \bigcup B$ and $\alpha_2 \in \bigcup B$;
3. for every β-formula $\beta \in \bigcup B$, $\beta_1 \in \bigcup B$ or $\beta_2 \in \bigcup B$.

Lemma 2 *Every developed and open branch is realizable.*

Proof: Let B some developed and open branch of tableau T. We defined a valuation v by:

1. if $p \in \bigcup B$, then $v(p) = 1$;
2. if $\neg p \in \bigcup B$, then $v(p) = 0$;
3. if $p \notin \bigcup B$ and $\neg p \notin \bigcup B$, then set (arbitrarily) $v(p) = 1$.

We prove by structural induction on a that: if $A \in \bigcup B$, then $v(A) = 1$, and if $\neg A \in \bigcup B$, then $v(A) = 0$.

Indeed, this is true for propositional variables.

If A is a α-formula, then $\alpha_1 \in \bigcup B$ and $\alpha_2 \in \bigcup B$. By induction hypothesis, $v(\alpha_1) = 1$, $v(\alpha_2) = 1$, and hence $v(\alpha) = 1$.

If A is a β-formula, then $\beta_1 \in \bigcup B$ or $\beta_2 \in \bigcup B$. By induction hypothesis, $v(\beta_1) = 1$ or $v(\beta_2) = 1$, and hence $v(\beta) = 1$. □

Proposition 2 *If there exists some closed tableau whose root is $\neg A$, then any developed tableau whose root is $\neg A$ is closed.*

Proof: By contradiction. Let T be some developed and open tableau whose root is $\neg A$, and let B be some open branch of T. By previous lemma, B is realizable, sand since $\neg A$ is in B, $\neg A$ is satisfiable. A is hence not a tautology, and hence not provable by tableau. There is no closed tableau with the root $\neg A$. □

We have all the ingredients to prove Theorem 8.

Suppose that A is not provable by tableau. Let T be a developed tableau whose root is $\neg A$. T is not closed. As in the previous proof, if B is some open branch of T, then B is realizable, and hence $\neg A$ is satisfiable. In other words, A is not a tautology.

5.6 One consequence of compactness theorem

Definition 8 *We will say that a set Σ of formulas is* refutable by tableau *if there exists some closed tableau with the root Σ.*

Corollary 3 *Every set Σ of formulas that is not satisfiable is refutable by tableau.*

Proof: By the compactness theorem, a set of formulas Σ that is not satisfiable has a finite subset Σ_0 that is not satisfiable. This finite set of formulas as a refutation by tableau, i.e. there is a closed table with the root Σ_0. This tableau also provides a closed tableau with the root Σ. □

6 Bibliographic notes

Suggested readings To go further on the notions of this chapter, we suggest to read [Cori & Lascar, 1993], and [Mendelson, 1987] for the demonstration based on modus ponens, of [Stern, 1994] for a simple presentation of the proof methods based on resolution, and to [Lassaigne & de Rougemont, 2004] and [Nerode & Shore, 1997] for the tableau based methods.

Bibliography This chapter has been written by using [Cori & Lascar, 1993], and [Dehornoy, 2006] for the part on proof methods based on modus ponens, and [Stern, 1994] for the presentation of proofs by resolution method. The part on natural deduction is taken from [Dowek, 2008]. The section on the tableau method is taken from book [Lassaigne & de Rougemont, 2004].

Index

Bibliography

[Cori & Lascar, 1993] Cori, R. & Lascar, D. (1993). *Logique mathématique. Volume I*. Masson.

[Davis, 1989] Davis, R. E. (1989). *Truth, deduction, and computation - logic and semantics for computer science*. Principles of computer science series. Computer Science Press.

[Dehornoy, 2006] Dehornoy, P. (2006). *Logique et théorie des ensembles*. Notes de cours.

[Dowek, 2008] Dowek, G. (2008). *Les démonstrations et les algorithmes*. Polycopié du cours de l'Ecole Polytechnique.

[Lassaigne & de Rougemont, 2004] Lassaigne, R. & de Rougemont, M. (2004). *Logic and complexity*. Discrete Mathematics and Theoretical Computer Science. Springer. https://doi.org/10.1007/978-0-85729-392-3

[Mendelson, 1987] Mendelson, E. (1987). *Introduction to mathematical logic (3. ed.)*. Chapman and Hall.

[Nerode & Shore, 1997] Nerode, A. & Shore, R. A. (1997). *Logic for Applications, Second Edition*. Graduate Texts in Computer Science. Springer. https://doi.org/10.1007/978-1-4612-0649-1

[Stern, 1994] Stern, J. (1994). Fondements mathématiques de l'informatique. *Ediscience International, Paris*.