Mathematical Reasoning, Proof Principles, and Logic .

- construct and read mathematical proofs

Motivation

- Computer scientists and engineers write programs and build systems.
- It is very important to have rigorous methods to check that these programs and systems behave as expected (are correct, have no bugs).
- It is also important to have methods to analyze the complexity of programs (time/space complexity)

- What are basic reasoning principles and rules of logic?

- What is a proof?

- Proofs can be very informal, using a set of loosely defined logical rules, possibly omitting steps and premises

- Proofs can be completely formal, using a very clearly defined set of rules and premises.
- Such proofs are usually processed or produced by programs called proof checkers and theorem provers

- It should be said that it is practically impossible to write formal proofs.

- This is because it would be extremely tedious and time-consuming to write such proofs and these proofs would be huge and thus, very hard to read

- In principle, it is possible to write formalized proofs and sometimes it is desirable to do so if we want to have absolute confidence in a proof.

- For example
  - A flight-control system is not buggy so that a plane does not accidentally crash
  - A program running a nuclear reactor will not malfunction,
  - Nuclear missiles will not be fired as a result of a buggy “alarm system"

- However, 99.99% of us will not have the time or energy to write formal proofs

- it is important to understand clearly what are the rules of reasoning that we use when we construct informal proofs

1.2 Inference Rules, Deductions, The Proof Systems $\mathscr{N}_m^{\Rightarrow}$ and $\mathscr{NG}_m^{\Rightarrow}$

axioms and proof rules (also called inference rules)

constructing proofs

In mathematics, we prove statements.

Statements may be atomic or compound,

A compound statement is built up from simpler statements using logical connectives,

such as implication (if–then), conjunction (and), disjunction (or), negation (not), and (existential or
universal) quantifiers

1. “A student is eager to learn.”
2. “A student wants an A.”
3. “An odd integer is never 0.”
4. “The product of two odd integers is odd.”

Atomic statements may also contain “variables” (standing for arbitrary objects).
For example
1. human(x): “x is a human.”
2. needs-to-drink(x): “x” needs to drink

An example of a compound statement is human(x) ⇒ needs-to-drink(x)

∀x(human(x) ⇒ needs-to-drink(x));

This is read: “For every x, if x is a human then x needs to drink.”

$\exists x(human(x) \Rightarrow needs\text{-}to\text{-}drink(x))$;

This is read: “There is some x such that, if x is a human then x needs to drink.

We often denote statements (also called propositions or (logical) formulae) using letters,
such as A,B,P,Q, and so on, typically upper-case letters (but sometimes Greek letters, φ, ψ, etc.)

conjunction

P ∧ Q (say, P and Q)

P and Q are statements, then their conjunction is denoted P ∧ Q (say, P and Q),

disjunction

$$P \vee Q$$

their disjunction denoted $P \vee Q$ (say, P or Q),

implication

$$P \Rightarrow Q$$

their implication $P \Rightarrow Q$ or $P \supset Q$ (say, if P then Q).

.

The atomic statements $\bot$ (falsity), which corresponds to false

The atomic statement $\top$ (truth), which corresponds to true

Constant $\bot$ is also called falsum or absurdum.

Cause confusing

Then, it is convenient to define the negation of P as P ⇒⊥ and to abbreviate it as ¬P (or sometimes ~ P).

Thus, ¬P (say, not P) is just a shorthand for P ⇒⊥.

$\neg P = (P \Rightarrow \bot)$ is true if and only if P is not true

---

because if both P and $P \Rightarrow \bot$ were

true then we could conclude that $\bot$ is true, an absurdity,

and if both P and $P \Rightarrow \bot$ were false then P would have to be both true and false, again, an absurdity.

One true one not true

¬P is provable iff for every proof of P we can derive a contradiction (namely, ⊥ is provable). In particular, P should not be provable.

For example, $\neg(Q \wedge \neg Q)$ is provable (as we show later, because any proof of $Q \wedge \neg Q$ yields a proof of $\bot$).

However, the fact that a proposition P is not provable does not imply that ¬P is provable. There are plenty of propositions such that both P and ¬P are not provable,

such as Q ⇒ R, where Q and R are two

unrelated propositions (with no common symbols

matching parentheses

Whenever necessary to avoid ambiguities, we add matching parentheses: $(P \wedge Q)$, $(P \vee Q)$,$(P \Rightarrow Q)$.

For example, $P \vee Q \wedge R$ is ambiguous; it means either $(P \vee (Q \wedge R))$ or $((P \vee Q) \wedge R)$.

equivalence

If P and Q are statements,
then their equivalence, denoted
$P \equiv Q$ is an abbreviation for
$(P \Rightarrow Q) \wedge (Q \Rightarrow P)$.

We often say “P if and only if Q” or even “P iff Q” for $P \equiv Q$.

As we show shortly, to prove a logical equivalence $P \equiv Q$, we have to prove both implications

$P \Rightarrow Q$ and $Q \Rightarrow P$

An implication $P \Rightarrow Q$ should be understood as an if–then statement; if P is true then Q is also true.

A better interpretation is that any proof of $P \Rightarrow Q$ can be used to construct a proof of Q given any proof of P.

As a consequence of this interpretation, we show later that if ¬P is provable, then P ⇒ Q is also provable (instantly) whether or not Q is provable. In such a situation, we often say that P⇒Q

However, we have to wait until Section 1.3 to see this)

For example, $(P \wedge \neg P) \Rightarrow Q$ is provable for any arbitrary Q (because if we assume that $P \wedge \neg P$ is provable, then we derive a contradiction,
and then another rule of logic tells us that any proposition whatsoever is provable.

Typically, the statements that we prove depend on some set of hypotheses, also called premises (or assumptions).

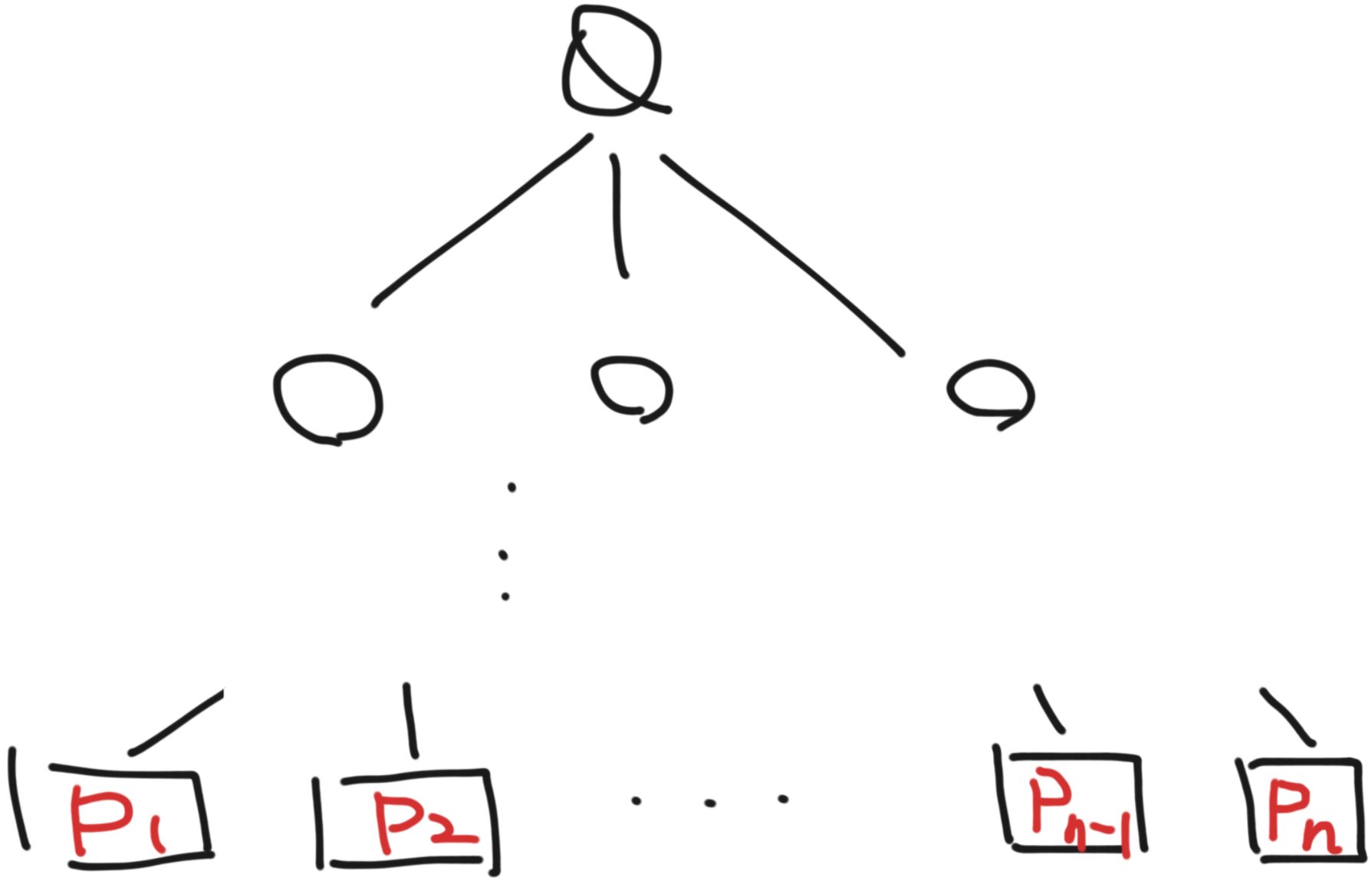
As we show shortly, this amounts to proving implications of the form $(P1 \wedge P2 \wedge \cdots \wedge Pn) \Rightarrow Q$.

However, there are certain advantages in defining the notion of proof (or deduction) of a proposition from a set of premises.

Sets of premises are usually denoted using upper-case Greek letters such as Γ or Δ.

Roughly speaking, a deduction of a proposition Q from a set of premises Γ is a finite labeled tree whose root is labeled with Q (the conclusion), whose leaves are
labeled with premises from Γ (possibly with multiple occurrences), and such that every interior node corresponds to a given set of proof rules (or inference rules). Certain simple deduction trees are declared as obvious proofs, also called axioms

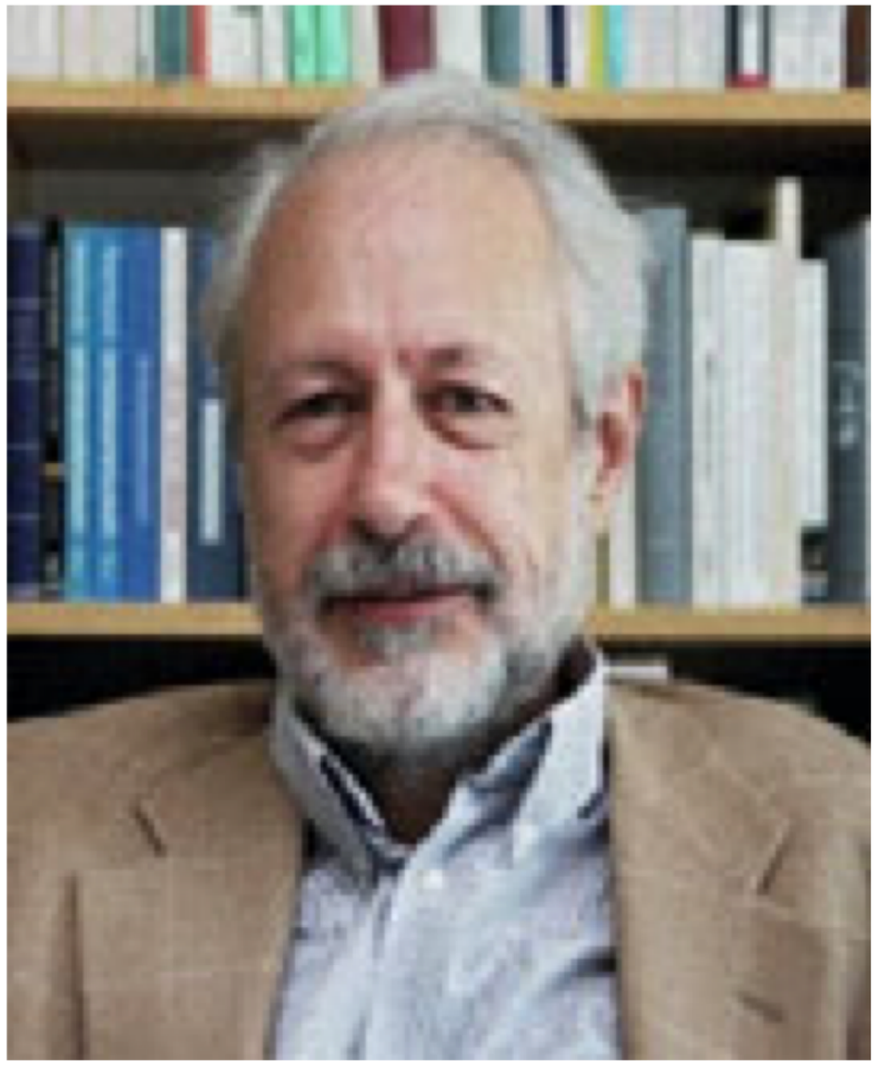
$$\Gamma = \{ P_1 , P_2 , P_3 , P_4 \cdots P_n \}$$



A natural deduction system

- mathematical logic
- G. Gentzen in the early 1930s (thoroughly investi-gated by D. Prawitz in the mid 1960s

David Hilbert, 1862–1943



Gerhard Gentzen, 1909–1945
(middle right),
and Dag Prawitz, 1936– (right

The major advantage

It captures quite nicely the “natural” rules of reasoning that one uses when proving mathematical statements.

This does not mean that it is easy to find proofs in such a system or that this system is indeed very intuitive

How do we proceed to prove an implication, A ⇒ B

- The rule, called ⇒-intro, is: assume that A has already been proven and then prove B, making as many uses of A as needed.

odd(n) ⇒ odd(n+2)

if we assume odd(n), then we can conclude odd(n+2)

Note that the effect of rule ⇒-intro is to introduce the premise odd(n), which was temporarily assumed, into the left-hand side of the proposition odd(n) ⇒ odd(n+2).

This is why this rule is called the implication introduction

Following the rule ⇒-intro, we assume odd(n) (which means that we take as proven the fact that n is odd) and we try to conclude that n+2 must be odd. However, to say that n is odd is to say that n = 2k+1 for some whole number k. Now,
n+2 = 2k+1+2 = 2(k+1)+1,
which means that n+2 is odd. (Here, n = 2h+1, with h = k+1, and k+1 is a
whole number because k is.)
Therefore, we proved that if

After having applied the rule ⇒-intro,

we should really make sure that the premise odd(n) which was made temporarily active is deactivated, or as we say, discharged.

When we write informal proofs, we rarely (if ever) explicitly discharge premises when we apply the rule ⇒-

intro but if we want to be rigorous we really should

$$P \Rightarrow (Q \Rightarrow P)$$

Premise {P,Q},
Prove P from {P,Q}

.

According to our rule, we assume P as a premise and we try prove Q ⇒ P assuming P.

In order to prove Q ⇒ P, we assume Q as a new premise so the set of premises becomes {P,Q}, and then we try to prove P from P and Q. This time, it should be obvious that P is provable because we assumed both P and Q

one of the basic axioms of our logic

P is always provable from any set of assumptions including P itself

So, we have obtained a proof of $P \Rightarrow (Q \Rightarrow P)$.

$$P \Rightarrow ((P \Rightarrow Q) \Rightarrow Q)$$

If P and P ⇒ Q are both provable, then Q is provable

What is not entirely satisfactory about the above "proof" of $P \Rightarrow (Q \Rightarrow P)$

is that when the proof ends, the premises P and Q are still hanging around as "open" assumptions. However, a proof should not depend on any "open" assumptions and
to rectify this problem we introduce a mechanism of "discharging" or "closing"
premises

What this means is that certain rules of our logic are required to discard (the usual terminology is “discharge”) certain occurrences of premises so that the resulting proof does not depend on these premises

Technically, there are various ways of implementing the discharging mechanism but they all involve some form of tagging (with a “new” variable). For example, the rule formalizing the process that we have just described to prove an implication, $A \Rightarrow B$, known as $\Rightarrow$-introduction, uses a

tagging mechanism described precisely in Definition 1.1

For example, after a moment of thought, I think most people would
want the proposition $P \Rightarrow ((P \Rightarrow Q) \Rightarrow Q)$ to be provable.

If we follow the procedure that we have advocated, we assume both P and $P \Rightarrow Q$ and we try to prove Q.
For this, we need a new rule, namely:
If P and $P \Rightarrow Q$ are both provable, then Q is provable.

The above rule is known as the $\Rightarrow$-elimination rule and it is formalized in tree-form in Definition 1.1

Δ the multiset, a set possibly with multiple occurrences of its members). Some of the propos in Δ may be tagged by variables. The list of untagged propositions in Δ is the list of premises of the deduction tree

$\mathscr{D}$ a deduction tree

P root

A deduction tree

$$\dfrac{\dfrac{P \Rightarrow (R \Rightarrow S) \qquad P}{R \Rightarrow S} \qquad \dfrac{\dfrac{P \Rightarrow Q \qquad P}{Q} \qquad Q \Rightarrow R}{R}}{S}$$

the premises form the multiset

$$\Delta = \{P \Rightarrow (R \Rightarrow S), P, Q \Rightarrow R, P \Rightarrow Q, P\},$$

the conclusion is S

A label

$$\frac{\dfrac{}{P^x, Q}}{Q}\,x$$

$$P \Rightarrow Q$$

$$\frac{Q \Rightarrow R \qquad Q}{R}$$

the proposition R is the result of applying the ⇒-elimination rule to the two premises Q ⇒ R and Q.

Thus, the use of the bar is just a convention used by logicians going back at least to the 1900s. Removing the bar everywhere would not change anything in our trees, except perhaps reduce their readability.

Because propositions do not arise from the vacuum but instead are built up from
a set of atomic propositions using logical connectives (here, ⇒), we assume the existence of an “official set of atomic propositions,” or set of propositional symbols, PS = {P1,P2,P3,...}. So, for example, P1⇒ P2 and P1⇒ (P2⇒ P1) are propo-sitions.

Definition 1.1. The axioms, inference rules, and deduction trees for implicational logic are defined as follows.

Axioms.

(i) Every one-node tree labeled with a single proposition P is a deduction tree for P with set of premises {P}.

(ii) The tree

$$\frac{\Gamma \qquad P}{P}$$

is a deduction tree for P with multiset set of premises, $\Gamma \cup \{P\}$

$$\frac{\overbrace{P_1, \ldots, P_1}^{k_1}, \ldots, \overbrace{P_i, \ldots, P_i}^{k_i}, \ldots, \overbrace{P_n, \ldots, P_n}^{k_n}}{P_i}$$

where $k_1, \ldots, k_n \geq 1$ and $n \geq 1$.

The ⇒-introduction rule

If $\mathscr{D}$ is a deduction tree for Q from the premises in Γ ∪{P}, then

$$\dfrac{\begin{array}{c}\Gamma, P^{x} \\ \mathscr{D} \\ Q\end{array}}{P \Rightarrow Q}\ x$$

is a deduction tree for P ⇒ Q from Γ

The ⇒-elimination rule.

If $\mathscr{D}_1$ is a deduction tree for P ⇒ Q from the premises Γ and $\mathscr{D}_2$ is a deduction for P from the premises Δ, then

$$\dfrac{\begin{array}{cc} \Gamma & \Delta \\ \mathscr{D}_1 & \mathscr{D}_2 \\ P \Rightarrow Q & P \end{array}}{Q}$$

is a deduction tree for Q from the premises in Γ ∪ Δ.

A deduction tree is either a one-node tree labeled with a single proposition or a tree constructed using the above axioms and rules.

A proof tree is a deduction tree such that all its premises are discharged.

The above proof system is denoted $\mathcal{N}_m^{\Rightarrow}$ (here, the subscript m stands for minimal, referring to the fact that this a bare-bones logical system).

Observe that a proof tree has at least two nodes. A proof tree Π for a proposition
P may be denoted
Π
P
with an empty set of premises