Introduction to proof theory (for computer scientists)
Why do proofs matter to computer scientists?
A bad start

Hilbert’s program: find an algorithm that applies to a proposition, returns 1 if the proposition has a proof and 0 if it does not

Church’s theorem (1936): negative answer: provability undecidable

Proofs and algorithms are two completely different things
A byproduct: computability theory
Two different methods

Is 143 prime or composite?

Find a proof of the proposition “143 is composite”
or apply an algorithm to decide primality / non primality

General method: build a proof
Algorithms can only be used for very specific decidable problems

But...
1. Computers are truth judgment machines

The 100\textsuperscript{th} decimal of \(\pi\) is a 9
2. Proof-checking and proof-search algorithms

Provability undecidable

but correctness of proof decidable: proof-checking algorithms
and provability semi-decidable: proof-search semi-algorithms
3. Proofs of algorithms and programs

Critical systems: transportation, energy, medicine...
A way to avoid bugs

Prove your programs correct

Programs (for instance +): do, do, do... what for?
4. Brouwer-Heyting-Kolmogorov interpretation

Proofs are algorithms

The language of proofs: a programming language where all programs terminate
(always terminating languages?)
5. Theories

Proofs are not purely logical objects

Theories: arithmetic, set theory, type theory, etc.

Theories: sets of axioms
Some theories: algorithms
Before being a theory of proofs, logic is a theory of languages (λογος)

Logic and (part of) computer science are two branches of a (new) theory of languages
This course

Today: Inductive definitions – Languages

Tomorrow: Natural deduction

Wednesday: Arithmetic – Simply typed λ-calculus

Thursday: The termination of proof reduction

Friday: Automated theorem proving
Inductive definitions
Do you know a set $E$ of numbers that

- contains 0
- contains $a + 2$ each time it contains $a$
Is it the only one?
Is it the only one?

\{0, 2, 4, 6, 8\ldots\}
\{0, 1, 2, 3, 4\ldots\} = \mathbb{N}
\mathbb{Z}
\mathbb{Q}
\mathbb{R}
...

But $\{0, 2, 4, 6, 8\ldots\}$ is the smallest
A theorem

A set of rules of the form

- if $a_1, \ldots, a_n$ are in $E$ then $f(a_1, \ldots, a_n)$ is in $E$

then there exists a smallest set satisfying these rules

An inductive definition
An example

- 0 is in $E$
- if $a$ is in $E$ then $a + 2$ is in $E$

Smallest set: $\{0, 2, 4\ldots\}$
Another example

\[ \overline{b} \]

\[ \frac{X}{\overline{a} X a} \]
Another example

\[ x \mapsto C \mapsto x \]
\[ x \mapsto R \mapsto y \mapsto y \mapsto C \mapsto z \]
\[ x \mapsto C \mapsto z \]
Three characterizations of this smallest set (1)

Consider all sets $S$ such that if $a_1, ..., a_n$ are in $S$ then so is $f(a_1, ..., a_n)$
Take their intersection

Example: the intersection of $\{0, 2, 4, 6, 8, ...\}$, $\{0, 1, 2, 3, 4, ...\}$, $\mathbb{Z}$...
is $\{0, 2, 4, 6, 8, ...\}$,
Three characterizations of this smallest set (2)

\[ F(A) = \bigcup_i \{ f_i(a_1, \ldots, a_{n_i}) \mid a_1, \ldots, a_{n_i} \in A \} \]

Union of \( \emptyset \), \( F(\emptyset) \), \( F(F(\emptyset)) \)...

Example
\( \emptyset = \{ \} \)
\( F(\emptyset) = \{0\}, \)
\( F(F(\emptyset)) = \{0, 2\} \)
\( F(F(F(\emptyset))) = \{0, 2, 4\} \)
union: \( \{0, 2, 4, 6\ldots\} \)
Three characterizations of this smallest set (3)

$a$ in $E$ if and only if there exists a tree such that

- if a node is labeled with $x$ then its children are labeled with $y_1, ..., y_n$ such that $x = f(y_1, ..., y_n)$
- the root is labeled with $a$

Example

```
       6
      / \
     4   2
    /   / \
   0   2   4
```
Languages
I. Languages, in general
Forget the linearity constraint of natural languages
Do not care if you write $3 + 4$, $+(3, 4)$ or $3\ 4\ +$

Expressions are trees
Variable free languages

A (variable free) language is a set of symbols, each having an arity also called its number of arguments.

The set of expressions of the language is the set of trees inductively defined as:

- if $t_1, \ldots, t_n$ are expressions, and $f$ is a symbol of arity $n$, then $f(t_1, \ldots, t_n)$ is an expression.
An example

A nullary symbol (constant) 0
A unary symbol $S$
Two binary symbols $+, \times$
Two unary symbols even, odd
A binary symbol $\Rightarrow$

$odd(S(S(S(0)))) \Rightarrow even(S(S(S(S(0))))))$
An exercise

Write

\[ \text{odd}(S(S(S(0)))) \Rightarrow \text{even}(S(S(S(S(0)))))) \]

as a tree
If a number is odd then its successor is even

\[ \forall x \ (\text{odd}(x) \Rightarrow \text{even}(S(x))) \]

Variables
Symbols that bind variables
Languages with variables

The arity of a symbol is a tuple $\langle k_1, \ldots, k_n \rangle$ the symbol has $n$ arguments, it binds $k_1$ variables in the first, $\ldots$, $k_n$ variables in the $n^{\text{th}}$

Example: $\forall$ has arity $\langle 1 \rangle$

A set of symbols and a finite set of variables

Expressions are inductively defined by the rules:

- variables are expressions,
- if $f$ is a symbol of arity $\langle 1, 3 \rangle$, $t$ and $u$ are expression, $w, x, y, z$ are variables, then $f(w \ t, x \ y \ z \ u)$ is an expression (to be generalized)
\[ f(x_1^1 \ldots x_{k_1}^1 \ t_1, \ldots, x_1^n \ldots x_{k_n}^n \ t_n) \text{ is the tree} \]
Free and bound variables

- $\text{Var}(x) = \{x\},$

- $\text{Var}(f(x_1^{k_1} \ldots x_1^{k_1} t_1, \ldots, x_1^{n_k} \ldots x_k^{n_k} t_n))$
  $\quad = \text{Var}(t_1) \cup \{x_1^{k_1}, \ldots, x_1^{k_1}\} \cup \ldots \cup \text{Var}(t_n) \cup \{x_n^{k_n}, \ldots, x_k^{n_k}\}.$

$\text{Var}(\forall x \ (x = x))$ ?

- $\text{FV}(x) = \{x\},$

- $\text{FV}(f(x_1^{k_1} \ldots x_1^{k_1} t_1, \ldots, x_1^{n_k} \ldots x_k^{n_k} t_n))$
  $\quad = (\text{FV}(t_1) \setminus \{x_1^{k_1}, \ldots, x_1^{k_1}\}) \cup \ldots \cup (\text{FV}(t_n) \setminus \{x_n^{k_n}, \ldots, x_k^{n_k}\})$

$\text{FV}(\forall x \ (x = x))$ ?
An exercise

# let x = 4 in x + 2;;
- : int = 6

What is the arity of let?
Languages with several sorts of objects

\[ 0, \, S, \, +, \, \times, \, \text{even}, \, \text{odd}, \, \Rightarrow, \, \forall \]

We want to distinguish \( 0, \, S(0), \, S(x), \, \ldots \): terms from \( \text{even}(0), \, \text{odd}(0), \, \forall x \, (\text{even}(x)), \, \ldots \): propositions

But may be also vectors from scalars, integers from floating point numbers...
Languages with several sorts of objects

A set of sorts \( \{ \text{Term, Prop} \} \), more generally \( S \)

The arity of a symbol is a \( n + 1 \)-tuple of sorts \( \langle s_1, ..., s_n, s' \rangle \)

If \( t_1 \) is an expression of sort \( s_1 \), \( t_2 \) expression of sort \( s_2 \), ..., \( t_n \) expression of sort \( s_n \) and \( f \) a symbol of arity \( \langle s_1, ..., s_n, s' \rangle \), then \( f(t_1, ..., t_n) \) is an expression of sort \( s' \)
Several sorts of object + binders

Example \( \forall \) has arity \( \langle \langle \text{Term}, \text{Prop} \rangle, \text{Prop} \rangle \)
II. The languages of predicate logic
A set $S$ of sort of terms plus a sort $\text{Prop}$

Only $\forall$ and $\exists$ as binding symbols

The symbols are divided into
- function symbols $f$ of arity $\langle s_1, \ldots, s_n, s' \rangle$
- predicate symbols $P$ of arity $\langle s_1, \ldots, s_n, \text{Prop} \rangle$ (written $\langle s_1, \ldots, s_n \rangle$)
- symbols common to all languages $\top, \bot, \neg, \land, \lor, \Rightarrow, \forall, \exists$

$$\forall x \ (\text{even}(x) \Rightarrow \text{odd}(S(x)))$$
An exercise

\[ \forall x \ (\text{even}(x) \Rightarrow \text{odd}(S(x))) \]

What is \( x \), \( S \), \textit{even}? \( odd? \) \( S? \Rightarrow? \ \forall? \)
\( S(x)? \text{ odd}(S(x))? \)
Message to take home

Two fundamental notions: inductive definition, language in general

A particular case: the languages of predicate logic
Tomorrow

Proof