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 algoritm 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^{th} decimal of π 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

6. Languages

Before being a theory of proofs, logic is a theory of languages ($\lambda o \gamma o \varsigma$)

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

An exercise

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?

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

But $\{0, 2, 4, 6, 8 ...\}$

is the smallest

A theorem

A set of rules of the form

- if $a_1, ..., a_n$ are in E then $f(a_1, ..., a_n)$ 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

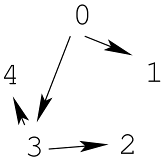
Smallest set: $\{0, 2, 4...\}$

Another example

$$\overline{b}$$

$$\frac{X}{a\ X\ a}$$

Another example



$$\frac{}{x\ C\ x}$$

$$\frac{x\ R\ y \qquad y\ C\ z}{x\ C\ z}$$

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)

$F(A) = \bigcup_i \{f_i(a_1, ..., a_{n_i}) \mid a_1, ..., a_{n_i} \in A\}$

Union of $\varnothing$, $F(\varnothing)$, $F(F(\varnothing))$...

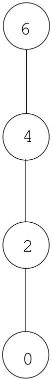
Example
$\varnothing = \{\}$
$F(\varnothing) = \{0\}$,
$F(F(\varnothing)) = \{0, 2\}$
$F(F(F(\varnothing))) = \{0, 2, 4\}$
union: $\{0, 2, 4, 6...\}$

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



$$\dfrac{\dfrac{\dfrac{\overline{0}}{2}}{4}}{6}$$

Labels

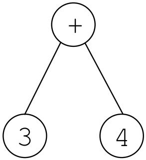
0	0	z	.	z
2	2	f	.	f
4	4	f	.	f
6	6	f	.	f

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

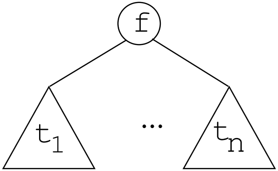


Variable free languages

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, ..., t_n are expressions, and f is a symbol of arity n, then $f(t_1, ..., t_n)$ 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

$$\mathit{odd}(S(S(S(0)))) \Rightarrow \mathit{even}(S(S(S(S(0)))))$$

as a tree

If a number is odd then its successor is even

$$\forall x \ (odd(x) \Rightarrow even(S(x)))$$

Variables
Symbols that bind variables

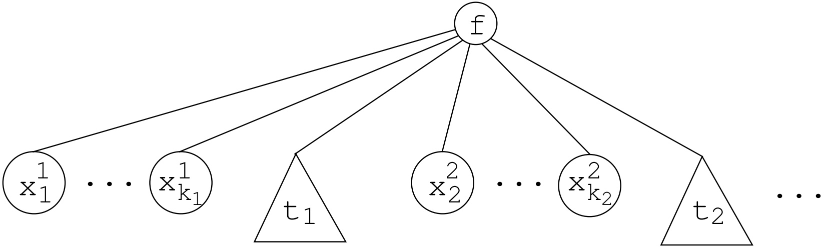
Languages with variables

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

$f(x_1^1 \dots x_{k_1}^1 t_1, \dots, x_1^n \dots x_{k_n}^n t_n)$ is the tree



Free and bound variables

- $Var(x) = \{x\}$,
- $Var(f(x_1^1...x_{k_1}^1 t_1, ..., x_1^n...x_{k_n}^n t_n))$
  $= Var(t_1) \cup \{x_1^1, ..., x_{k_1}^1\} \cup \cdots \cup Var(t_n) \cup \{x_n^n, ..., x_{k_n}^n\}$.

$Var(\forall x\ (x = x))$?

- $FV(x) = \{x\}$,
- $FV(f(x_1^1...x_{k_1}^1 t_1, ..., x_1^n...x_{k_n}^n t_n))$
  $= (FV(t_1) \setminus \{x_1^1, ..., x_{k_1}^1\}) \cup \cdots \cup (FV(t_n) \setminus \{x_n^n, ..., x_{k_n}^n\})$

$FV(\forall x\ (x = x))$?

An exercise

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

What is the arity of `let`?

Languages with several sorts of objects

0, S, $+$, $\times$, $even$, odd, $\Rightarrow$, $\forall$

We want to distinguish 0, $S(0)$, $S(x)$, ...: terms
from $even(0)$, $odd(0)$, $\forall x\ (even(x))$, ...: propositions

But may be also vectors from scalars, integers from floating point numbers...

Languages with several sorts of objects

A set of sorts $\{Term, Prop\}$, more generally $\mathcal{S}$

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

If t_1 is a 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

$$\langle\langle s_1^1, \ldots, s_{k_1}^1, s'^1\rangle, \ldots, \langle s_1^n, \ldots, s_{k_n}^n, s'^n\rangle, s''\rangle$$

Example $\forall$ has arity $\langle\langle \mathit{Term}, \mathit{Prop}\rangle, \mathit{Prop}\rangle$

II. The languages of predicate logic

A set $\mathcal{S}$ of sort of terms plus a sort *Prop*

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

The symbols are divided into

- function symbols f of arity $\langle s_1, ..., s_n, s' \rangle$
- predicate symbols P of arity $\langle s_1, ..., s_n, Prop \rangle$ (written $\langle s_1, ..., s_n \rangle$)
- symbols common to all languages $\top$, $\bot$, $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$

$$\forall x \; (even(x) \Rightarrow odd(S(x)))$$

An exercise

$$\forall x\ (\mathit{even}(x) \Rightarrow \mathit{odd}(S(x)))$$

What is x, S, even? odd? S? $\Rightarrow$? $\forall$?
$S(x)$? $\mathit{odd}(S(x))$?

Message to take home

Two fundamental notions: inductive definition, language in general

A particular case: the langages of predicate logic

Tomorrow

Proof