0. What we have seen so far

Proofs, constructive proof, theory

Today

An example of theory: arithmetic

A new topic: simply typed lambda-calculus

A theory: arithmetic

Examples of propositions

$$\forall x \exists y \ (x = 2 \times y \lor x = 2 \times y + 1)$$

$$\exists y \ (4 = 2 \times y)$$

$$\exists x \exists y \ (7 = (x + 2) \times (y + 2))$$

$$\forall x \exists y \ (y \geq x \land \mathit{prime}(y))$$

$\geq$, *prime*?

The languagage of arithmetic

0, S, $Pred$, $+$, $\times$
$Null$, $=$

The axioms of infinity

$$Pred(0) = 0$$

$$\forall x \; (Pred(S(x)) = x)$$

$$Null(0)$$

$$\forall x \; \neg Null(S(x))$$

$$Pred(0) \longrightarrow 0$$

$$Pred(S(x)) \longrightarrow x$$

$$Null(0) \longrightarrow \top$$

$$Null(S(x)) \longrightarrow \bot$$

The axioms of addition and multiplication

$$\forall y\ (0 + y = y)$$

$$\forall x\ \forall y\ (S(x) + y = S(x + y))$$

$$\forall y\ (0 \times y = 0)$$

$$\forall x\ \forall y\ (S(x) \times y = (x \times y) + y)$$

$$0 + y \longrightarrow y$$

$$S(x) + y \longrightarrow S(x + y)$$

$$0 \times y \longrightarrow 0$$

$$S(x) \times y \longrightarrow (x \times y) + y$$

Induction

No other numbers than those constructed with 0 and S

Every class containing 0 and closed by S contains everything

Besides ι, a sort κ for classes, a predicate symbol ϵ

$$\forall c\ (0 \epsilon c \Rightarrow \forall x\ (x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow \forall y\ y \epsilon c)$$

Comprehension axiom scheme: existence of some classes

$$\forall x_1 ... \forall x_n \exists c \forall y\ (y \epsilon c \Leftrightarrow A)$$

if A does not contain ϵ (predicative arithmetic)

Comprehension as a rewrite rule

$$\forall x_1 ... \forall x_n \exists c \forall y \ (y \in c \Leftrightarrow A)$$

Introduce a notation for this class: $f_{x_1,...,x_n,y,A}(x_1, ..., x_n)$

$$\forall x_1 ... \forall x_n \forall y \ (y \in f_{x_1,...,x_n,y,A}(x_1, ..., x_n) \Leftrightarrow A)$$

$$y \in f_{x_1,...,x_n,y,A}(x_1, ..., x_n) \longrightarrow A$$

How to use these axioms to prove $\forall y\ (y + 0 = y)$?

High school proof:
$0 + 0 = 0$
$\forall x\ (x + 0 = x \Rightarrow S(x) + 0 = S(x))$
hence $\forall y\ (y + 0 = y)$

Using the axioms

$$\forall y\ (0 + y = y)$$

$$\forall x\ \forall y\ (S(x) + y = S(x + y))$$

How do we know

$$0 + 0 = 0 \Rightarrow \forall x\ (x + 0 = x \Rightarrow S(x) + 0 = S(x)) \\ \Rightarrow \forall y\ (y + 0 = y)\ ?$$

How do we know

$$0 + 0 = 0 \Rightarrow \forall x\ (x + 0 = x \Rightarrow S(x) + 0 = S(x)) \\ \Rightarrow \forall y\ (y + 0 = y)\ ?$$

$$\forall c\ (0 \epsilon c \Rightarrow \forall x\ (x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow \forall y\ y \epsilon c)$$

How do we know

$$0 + 0 = 0 \Rightarrow \forall x\ (x + 0 = x \Rightarrow S(x) + 0 = S(x)) \Rightarrow \forall y\ (y + 0 = y)\ ?$$

$$\forall c\ (0 \mathrel{\epsilon} c \Rightarrow \forall x\ (x \mathrel{\epsilon} c \Rightarrow S(x) \mathrel{\epsilon} c) \Rightarrow \forall y\ y \mathrel{\epsilon} c)$$

$$\exists c \forall y\ (y \in c \Leftrightarrow y + 0 = y)$$

Induction as a rewrite rule

Induction axiom: all objects of sort ι are natural numbers
Alternative: not all objects are natural numbers, a predicate symbol N for the natural numbers

$$\forall c\ (0 \ \epsilon\ c \Rightarrow \forall x\ (x \ \epsilon\ c \Rightarrow S(x) \ \epsilon\ c) \Rightarrow \forall y\ (N(y) \Rightarrow y \ \epsilon\ c))$$

More axioms

$$N(0)$$

$$\forall x\ (N(x) \Rightarrow N(S(x)))$$

Relativization of quantifiers

$$\forall x\ (N(x) \Rightarrow \exists y\ (N(y) \wedge (x = 2 \times y \vee x = 2 \times y + 1)))$$

$$\forall y\ (N(y) \Rightarrow \forall c\ (0 \in c \Rightarrow \forall x\ (x \in c \Rightarrow S(x) \in c) \Rightarrow y \in c))$$

Converse provable (with $N(0)$ and $\forall x\ (N(x) \Rightarrow N(S(x)))$)
Alternative:

$$\forall y\ (N(y) \Leftrightarrow \forall c\ (0 \in c \Rightarrow \forall x\ (\in c \Rightarrow S(x) \in c) \Rightarrow y \in c))$$

($N(0)$ and $\forall x\ (N(x) \Rightarrow N(S(x)))$ dropped)

$$N(y) \longrightarrow \forall c\ (0 \in c \Rightarrow \forall x\ (x \in c \Rightarrow S(x) \in c) \Rightarrow y \in c)$$

Equality

Classes also used to express the properties of equality

$$\forall x \forall y \ (x = y \Leftrightarrow \forall c \ (x \ \epsilon \ c \Rightarrow y \ \epsilon \ c))$$

$$x = y \longrightarrow \forall c \ (x \ \epsilon \ c \Rightarrow y \ \epsilon \ c)$$

Exercise: prove reflexivity, symmetry, transitivity, and substitutivity

To sum up

$$Pred(0) \longrightarrow 0$$

$$Pred(S(x)) \longrightarrow x$$

$$Null(0) \longrightarrow \top$$

$$Null(S(x)) \longrightarrow \bot$$

$$0 + y \longrightarrow y$$

$$S(x) + y \longrightarrow S(x + y)$$

$$0 \times y \longrightarrow 0$$

$$S(x) \times y \longrightarrow (x \times y) + y$$

$$y \in f_{x_1,\ldots,x_n,y,A}(x_1, \ldots, x_n) \longrightarrow A$$

$$N(y) \longrightarrow \forall c\ (0 \in c \Rightarrow \forall x\ (x \in c \Rightarrow S(x) \in c) \Rightarrow y \in c)$$

$$x = y \longrightarrow \forall c\ (x \in c \Rightarrow y \in c)$$

Exercise

Prove

$$\forall x \forall y \ (S(x) = S(y) \Rightarrow x = y)$$

$$\forall x \ \neg(0 = S(x))$$

Another exercise

Prove

$$\forall x \exists y \; (x = 2 \times y \vee x = 2 \times y + 1)$$

Message to take home

A lot of proofs can be formalized in arithmetic

Arithmetic can be defined with rewrite rules only

Simply typed lambda-calculus

I. Functional programming languages

Functions

Do not focus on what computers do, but on what they compute

A function mapping input to output

A programming language: a language of functions

Functions

18^{th} century: $2 \times x + 4$
20^{th} century: $x \mapsto 2 \times x + 4$, $\lambda x\ 2 \times x + 4$

Two constructions:
abstraction (building a function): $\lambda x\ t$
application (using a function): $app(t, u) = (t\ u)$

Example: $((\lambda x\ (2 \times x + 4))\ 5)$

Exercise: arity of these symbols?

The β-reduction rule

$$((\lambda x : A\ t)\ u) \longrightarrow (u/x)t$$

Termination?

Termination?

$\delta = \lambda x\ (x\ x)$
$(\delta\ \delta) \longrightarrow (\delta\ \delta) \longrightarrow (\delta\ \delta) \longrightarrow (\delta\ \delta) \longrightarrow \ldots$

Simple types

Many sorted language with an infinite number of sorts
An inductive definition

- P_1, ..., P_n are simple types (e.g. *nat*, *string*, *bool*)
- if A and B are simple types, then $A \rightarrow B$ is a simple type

Terms of simply typed λ-calculus

For each simple type, an infinite set of variables of this type

- if x is a variable of type A, then x is a term of type A
- if t is a term of type $A \to B$ and u a term of type A, then $(t\ u)$ is a term of type B
- if x is a variable of type A and t a term of type B, then $\lambda x : A\ t$ is a term of type $A \to B$

Typing rules

Context: a type to some variables: $x : nat, g : nat \rightarrow nat \rightarrow nat$

In the context $\Gamma = (x : nat, g : nat \rightarrow nat \rightarrow nat)$ the term $t = \lambda y : nat\ (g\ y\ x)$ has type $A = (nat \rightarrow nat)$

$\Gamma \vdash t : A$

Inductive definition

$$\frac{}{\Gamma \vdash x : A} \; x : A \text{ in } \Gamma$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A \; t : A \to B}$$

$$\frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash (t \; u) : B}$$

II. The termination of the Simply typed λ-calculus

A first attempt

Induction over term structure:

- $t = x$ irreducible, hence terminating
- if t terminates, then $\lambda x : A\ t$ terminates
- if t and v terminate, $(t\ v)$ terminates

Reductions can be in t, in v, but also at the root
E.g. (untyped λ-calculus)
$\delta = \lambda x\ (x\ x)$ terminating, but not $(\delta\ \delta)$

The problem and a solution

If t is or reduces to $\lambda x\ u$, then $(t\ v)$ reduces to $(v/x)u$

When a proof by induction fails: try to prove a stronger property: if $t : A$ then $t \in R_A$

- If A atomic, t in R_A if t strongly terminating
- If $A = B \to C$, t in R_A if strongly terminating and whenever t reduces to $\lambda x : B\ u$, then for every v in R_B, $(v/x)u$ in R_C

Definition by induction over type structure

Four easy lemmas

1. If A is a type and x a variable, then $x \in R_A$
x strongly terminates and never reduces to abstraction
2. If t in R_A and t reduces to t', then t' in R_A
If t reduces to t' and t strongly terminates, then t' strongly terminates
If $A = B \rightarrow C$ and t' reduces to $\lambda x : B\ u$, then so does t, hence for every v in R_B, $(v/x)u$ in R_C

3. If all one-step reducts of $t = (u_1\ u_2)$ in R_A, then t in R_A

First t strongly terminates. A reduction sequence $t = t_1, t_2, ...$
If one element finite. Otherwise, $t \longrightarrow^1 t_2$, t_2 in R_A, strongly terminates, reduction sequence finite

If $A = B \rightarrow C$, and t reduces to $\lambda x : B\ v$, let
$t = t_1, t_2, ..., t_n = \lambda x : B\ v$ reduction sequence from t to $\lambda x : B\ v$.
t_1 application t_n abstraction, thus $n \geq 2$, $t \longrightarrow^1 t_2 \longrightarrow^* t_n$,
$t_2 \in R_A$, thus $(w/x)v$ in R_C

4. If $t_1 \in R_{A \to B}$ and $t_2 \in R_A$ then $(t_1\ t_2) \in R_B$

t_1 in $R_{A \to B}$, t_2 in R_A, hence strongly terminate
n_1 maximum length of a reduction sequence issued from t_1
n_2 maximum length of a reduction sequence issued from t_2
By induction on $n_1 + n_2$, $(t_1\ t_2)$ in R_B
Using 3., we only need to prove that every of its one step reducts is in R_B. If reduction in t_1 or in t_2 2. and induction hypothesis
Otherwise, $t_1 = \lambda x : A\ u$ and reduct is $(t_2/x)u$. In R_B by definition of $R_{A \to B}$

The theorem

t term of type A and σ substitution mapping each variable of type B to element of R_B, then σt in R_A

By induction on the structure of t

- If t variable, then σt in R_A by definition of σ
- If $t = (u_1\ u_2)$ then $u_1 : B \rightarrow A$ and $u_2 : B$

By induction hypothesis, σu_1 in $R_{B \rightarrow A}$ and σu_2 in R_B

By 4. $\sigma t = (\sigma u_1\ \sigma u_2)$ in R_A

- If $t = \lambda x : B\ u$ where u of type C

$\sigma t = \lambda x : B\ \sigma u$, a reduction sequence issued from this term can only reduce σu, by induction hypothesis, σu in R_C, thus reduction sequence is finite

Every reduct of σt of the form $\lambda x : B\ v$ where v reduct of σu

w any term of R_B, want $(w/x)v$ in R_C

$(w/x)v$ obtained by reducing $((w/x) \circ \sigma)u$

By induction hypothesis, $((w/x) \circ \sigma)u$ in R_C. Hence, by 2. $(w/x)v$ in R_C

Finally

If t of type A, then t strongly terminates
σ mapping each variable of type B to itself (in R_B by 1.)
$t = \sigma t$ in R_A. Hence it strongly terminates.

IV. A real programming language?

The computational expressivity of the simply typed λ-calculus

Add $0 : nat$ and $S : nat \rightarrow nat$
The natural numbers: 0, $(S\ 0)$, $(S\ (S\ 0))$...
The constant functions can be expressed in simply typed λ-calculus

$$\lambda x : nat\ (S\ (...(S\ 0)...))$$

those that add a constant to their arguments also

$$\lambda x : nat\ (S\ (...(S\ x)...))$$

But that is all

A lemma

An irreducible λ-term has the form

$$\lambda x_1 : A_1 \; ... \; \lambda x_n : A_n \; (x \; u_1 \; ... \; u_p)$$

where x is either a variable or a constant
$\lambda x_1 : A_1 \; ... \; \lambda x_n : A_n \; t'$ where t' not a λ (n possibly 0)
$t' = (t'' \; u_1 \; ... \; u_p)$ where t'' not an application (p possibly 0)
t'' not an abstraction, if $p > 0$ because the term is irreducible and if $p = 0$ because t' is not an abstraction

Message to take home

Simply typed lambda-calculus terminates

But is not very expresive

Tomorrow

The witness property for constructive proofs

Termination of proof reduction