Lecture 2: Self-interpretation in the Lambda-calculus

H. Geuvers

Radboud University
Nijmegen, NL

21st Estonian Winter School in Computer Science
Winter 2016

Outline

Self-interpretation in the Lambda Calculus

The limitations of Self-interpretation

More on data types

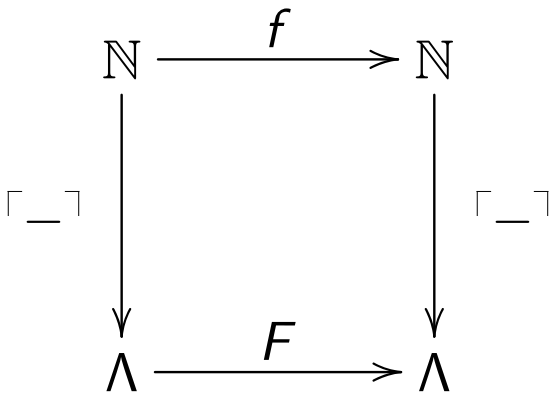
A data type D is a set with some operations (functions) on it.
An k-ary operation is a function $f : D^k \to D$.
Thereby a 0-ary operation $c : D^0 \to D$ is identified with a $c \in D$.
A datatype is determined by its operations on D:

$$\begin{array}{rcl} c_1, \ldots, c_{k_0} & : & D^0 \to D \;=\; D \\ f^1_1, \ldots, f^1_{k_1} & : & D^1 \to D \\ f^2_1, \ldots, f^2_{k_2} & : & D^2 \to D \\ & \ldots & \end{array}$$

$\mathtt{Nat}$ has $\mathtt{z} : \mathtt{Nat},\ \mathtt{s} : \mathtt{Nat} \to \mathtt{Nat}$.
$\mathtt{Tree}$ has $\mathtt{l} : \mathtt{Tree}, \mathtt{j} : \mathtt{Tree}^2 \to \mathtt{Tree}$

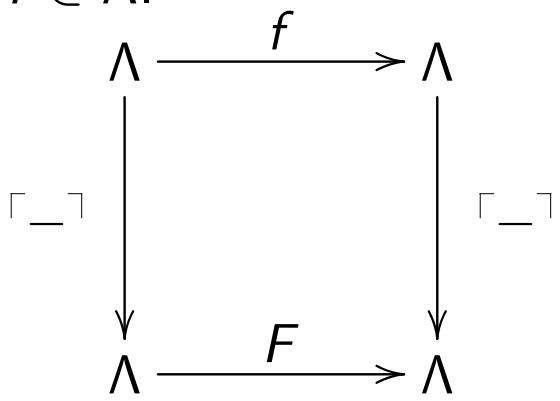
Defining functions on data types in the lambda-calculus

F λ-defines the function $f : \mathbb{N} \to \mathbb{N}$ if $\ulcorner f(n) \urcorner = F \ulcorner n \urcorner$ for all n:



Can we enode the λ-calculus in itself?

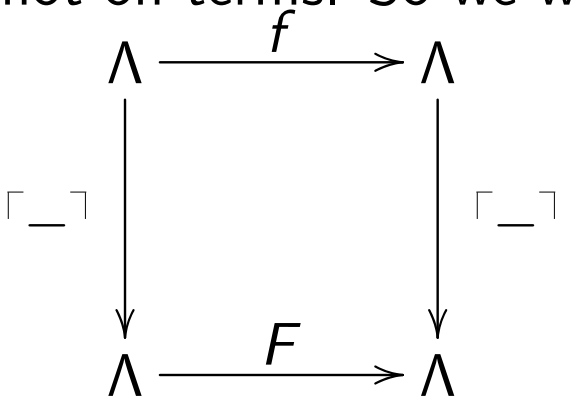
F λ-defines the function $f : \Lambda \to \Lambda$ if $\ulcorner f(M) \urcorner = F \ulcorner M \urcorner$ for all $M \in \Lambda$.



Is $\ulcorner\ \urcorner$ just the identity?
Why is this useful?

Defining functions on codes of lambda-terms

Some functions $f : \Lambda \to \Lambda$ can only be defined on codes of terms, not on terms. So we will need to talk about codes.



EXAMPLE. There is no λ-term F such that

$$F(M\,N) = M \text{ for all } M, N$$

- There is a λ-term F such that $F\ulcorner M\,N\urcorner = \ulcorner M\urcorner$ for all M, N. (With a suitable encoding $\ulcorner - \urcorner$.)
- We also have an evaluator E:

$$E\ulcorner M\urcorner = M$$

Packing and unpacking λ-terms

Given $M_1, \ldots, M_k$, define

$$\langle M_1, \ldots, M_k \rangle \;:=\; \lambda z.z\, M_1 \ldots M_k$$

Define $\mathbf{U}_i^k$, with $1 \leq i \leq k$ by

$$\mathbf{U}_i^k \;:=\; \lambda x_1 \ldots x_k.x_i$$

Then

$$\boxed{\begin{array}{rcl} \mathbf{U}_i^k M_1 \ldots M_k & = & M_i \\ \langle M_1, \ldots, M_k \rangle \mathbf{U}_i^k & = & \mathbf{U}_i^k M_1 \ldots M_k \;=\; M_i \end{array}}$$

Note that $\mathbf{K} = \mathbf{U}_1^2$

Second encoding of data types (Böhm-Piperno-Guerini)

Consider the data type D with

$$c : D,\ f : D \to D,\ g : D^2 \to D$$

The Böhm-Piperno-Guerini coding (also denoted by $\ulcorner t \urcorner$) is

$$\begin{aligned} \ulcorner c \urcorner &= \lambda e.e\,\mathbf{U}_1^3\,e \\ \ulcorner f(t) \urcorner &= \lambda e.e\,\mathbf{U}_2^3\,\ulcorner t \urcorner\,e \\ \ulcorner g(t_1,t_2) \urcorner &= \lambda e.e\,\mathbf{U}_3^3\,\ulcorner t_1 \urcorner\,\ulcorner t_2 \urcorner\,e \end{aligned}$$

PROPOSITION. The constructors (c, f, g) can be λ-defined: There are lambda terms F, G such that

$$\begin{aligned} F\ulcorner t \urcorner &= \ulcorner f(t) \urcorner \\ G\ulcorner t_1 \urcorner\ulcorner t_2 \urcorner &= \ulcorner g(t_1,t_2) \urcorner \end{aligned}$$

PROOF. Take

$$\begin{aligned} F &:= \lambda x\,e.e\,\mathbf{U}_2^3\,x\,e \\ G &:= \lambda x\,y\,e.e\,\mathbf{U}_3^3\,x\,y\,e. \end{aligned}$$

■

Recursion

THEOREM. Given $A_1, A_2, A_3 \in \Lambda$ there is an $H \in \Lambda$ such that

$$\begin{aligned} H\ulcorner c \urcorner &= A_1 H \\ H(\ulcorner f(t) \urcorner) &= A_2 \ulcorner t \urcorner H \\ H(\ulcorner g(t_1, t_2) \urcorner) &= A_3 \ulcorner t_1 \urcorner \ulcorner t_2 \urcorner H \end{aligned}$$

PROOF. Try $H = \langle\langle B_1, B_2, B_3 \rangle\rangle$.

$$\begin{aligned} H\ulcorner c \urcorner &= \langle\langle B_1, B_2, B_3 \rangle\rangle \ulcorner c \urcorner \\ &= \ulcorner c \urcorner \langle B_1, B_2, B_3 \rangle \\ &= \langle B_1, B_2, B_3 \rangle \mathbf{U}_1^3 \langle B_1, B_2, B_3 \rangle \\ &= B_1 \langle B_1, B_2, B_3 \rangle \\ &= A_1 \langle\langle B_1, B_2, B_3 \rangle\rangle \qquad \text{if } B_1 := \lambda z.A_1\langle z \rangle \\ &= A_1 H \\ H\ulcorner f(t) \urcorner &= \langle B_1, B_2, B_3 \rangle \mathbf{U}_2^3 \ulcorner t \urcorner \langle B_1, B_2, B_3 \rangle \\ &= B_2 \ulcorner t \urcorner \langle B_1, B_2, B_3 \rangle \\ &= A_2 \ulcorner t \urcorner H \qquad \text{if } B_2 := \lambda x\, z.A_2 x \langle z \rangle \\ H\ulcorner g(t_1, t_2) \urcorner &= A_3 \ulcorner t_1 \urcorner \ulcorner t_2 \urcorner H \qquad \text{if } B_3 := \lambda x\, y\, z.A_3 x y \langle z \rangle. \blacksquare \end{aligned}$$

Data type for coding lambda terms

To encode λ-terms we consider the data type D with

$$\begin{aligned} \mathtt{var} &: \quad D \to D \\ \mathtt{app} &: \quad D \to D \to D \\ \mathtt{abs} &: \quad D \to D \end{aligned}$$

- This data types is a bit strange: there is no base case.
- It is a priori unclear how to encode the λ-terms in this data type. How to encode the variable binding? (Later slide.)

Like before, we can define the constructors `Var`, `App`, `Abs`:

$$\begin{aligned} \mathtt{Var} &:= \lambda x\, e.e\, \mathbf{U}^3_1\, x\, e \\ \mathtt{App} &:= \lambda x\, y\, e.e\, \mathbf{U}^3_2\, x\, y\, e \\ \mathtt{Abs} &:= \lambda x\, e.e\, \mathbf{U}^3_3\, x\, e \end{aligned}$$

Recursion for the lambda terms data type

Like before, we have a recursion theorem:

THEOREM I. Given $A_1, A_2, A_3 \in \Lambda$ there is an $H \in \Lambda$ such that

$$\begin{aligned} H(\mathtt{Var}\, x) &= A_1\, x\, H \\ H(\mathtt{App}\, x\, y) &= A_2\, x\, y\, H \\ H(\mathtt{Abs}\, x) &= A_3\, x\, H \end{aligned}$$

PROOF. Take $H = \langle\langle B_1, B_2, B_3\rangle\rangle$ with

$$\begin{aligned} B_1 &:= \lambda x\, z.A_1 x\langle z\rangle \\ B_2 &:= \lambda x\, y\, z.A_2 xy\langle z\rangle \\ B_3 &:= \lambda x\, z.A_3 x\langle z\rangle. \end{aligned}$$

Then the equations hold indeed. ■.
(Exercise: Check this.)

Coding of lambda terms

Coding lambda terms $M \rightsquigarrow \ulcorner M \urcorner$

DEFINITION (Mogensen) The coding of λ-terms inside the λ-calculus is defined as follows.

$$\begin{array}{rcl} \ulcorner x \urcorner & := & \mathtt{Var}\ x \\ \ulcorner MN \urcorner & := & \mathtt{App} \ulcorner M \urcorner \ulcorner N \urcorner \\ \ulcorner \lambda x.M \urcorner & := & \mathtt{Abs}\ (\lambda x. \ulcorner M \urcorner) \end{array}$$

A variable x is encoded using x itself and abstraction "$\lambda x.$" is encoded using the same abstraction "$\lambda x.$".

Note: coding is not λ-definable: there is no term C such that $C\ M = \ulcorner M \urcorner$ for all M.
The reverse operation, evaluation, is λ-definable.

Self-interpretation

THEOREM. There exists a λ-term $\mathbf{E}$ (evaluator) such that for all $M \in \Lambda$

$$\mathbf{E}\ulcorner M \urcorner = M$$

PROOF. By recursion we can find an $\mathbf{E}$ such that

$$\begin{aligned} \mathbf{E}(\mathtt{Var}\ x) &= x \\ \mathbf{E}(\mathtt{App}\ x\ y) &= \mathbf{E}\,x\,(\mathbf{E}\,y) \\ \mathbf{E}(\mathtt{Abs}\ x) &= \lambda z.\mathbf{E}\,(x\,z) \end{aligned}$$

Then

$$\begin{array}{rcccccl} \mathbf{E}(\ulcorner x \urcorner) & = & \mathbf{E}(\mathtt{Var}\ x) & = & x & & \\ \mathbf{E}(\ulcorner MN \urcorner) & = & \mathbf{E}(\mathtt{App}\ulcorner M \urcorner\ulcorner N \urcorner) & = & \mathbf{E}\ulcorner M \urcorner(\mathbf{E}\ulcorner N \urcorner) & = & MN \\ \mathbf{E}(\ulcorner \lambda x.M \urcorner) & = & \mathbf{E}(\mathtt{Abs}(\lambda x.\ulcorner M \urcorner)) & = & \lambda x.\mathbf{E}\ulcorner M \urcorner & = & \lambda x.M. \end{array}$$

Filling in the details of $\mathbf{E}$ one has (writing $\mathbf{C} := \lambda x\,y\,z.x\,z\,y$)

$$\mathbf{E} = \langle\langle \mathbf{K}, \mathbf{S}, \mathbf{C} \rangle\rangle. \qquad \blacksquare$$

Why is this encoding so cool?

There are many encoding of Λ in itself. Why is this one so nice?

- Older ones all go through a coding of λ-terms as numbers: $\ulcorner - \urcorner : \Lambda \to \mathbb{N} \to \Lambda$.
  This one is direct, uses λ for λ-abstraction.
- $\mathbf{E}\ulcorner M \urcorner \twoheadrightarrow M$, also for terms with free variables.
- $\mathbf{E} = \langle\langle \mathbf{K}, \mathbf{S}, \mathbf{C} \rangle\rangle$, the initials of S.C. Kleene, one of the founders of the subject!
- $M_1 \equiv_\alpha M_2 \Rightarrow \ulcorner M_1 \urcorner \equiv_\alpha \ulcorner M_2 \urcorner$.
- $M_1 \neg \equiv_\alpha M_2 \Rightarrow \ulcorner M_1 \urcorner \neq_\beta \ulcorner M_2 \urcorner$.
- One can define a term R with $R\ulcorner M \urcorner \twoheadrightarrow \ulcorner N \urcorner$, if M has N as normal form.
- There is a Second Fixed Point Theorem (later)

Recursion for lambda terms using the encoding

We can state the recursion theorem for the encoded lambda terms slightly differently, as follows.

THEOREM II. Given $A_1, A_2, A_3 \in \Lambda$ there is an $H \in \Lambda$ such that

$$\begin{aligned} H\ulcorner x\urcorner &= A_1\, x\, H \\ H\ulcorner M\, N\urcorner &= A_2\, \ulcorner M\urcorner\, \ulcorner N\urcorner\, H \\ H\ulcorner \lambda x.M\urcorner &= A_3\, (\lambda x.\ulcorner M\urcorner)\, H \end{aligned}$$

PROOF. According to Theorem I, there is an H satisfying

$$\begin{aligned} H(\mathtt{Var}\, x) &= A_1\, x\, H \\ H(\mathtt{App}\, x\, y) &= A_2\, x\, y\, H \\ H(\mathtt{Abs}\, x) &= A_3\, x\, H \end{aligned}$$

These equations immediately imply the ones of the statement of Theorem II (check this!), so the same H suffices. ■.

Application 1

If you see someone coming out of 'arrivals' in an airport, you cannot determine where he/she comes from.
Similarly, there is no F such that for all $X, Y \in \Lambda$

$$F(X\ Y) = X$$

(Given the outcome of applying a function on an argument, there is no way we can determine the function that produced this outcome.)

PROPOSITION. There exists an $F_i \in \Lambda$, $i \in \{1, 2\}$ such that

$$F_i \ulcorner X_1\ X_2 \urcorner = \ulcorner X_i \urcorner.$$

PROOF. We do this for $i = 1$. By the Recursion Theorem II, there exists F_1 s.t.

$$F_1(\ulcorner X_1\ X_2 \urcorner) = A_2\ \ulcorner X_1 \urcorner\ \ulcorner X_2 \urcorner\ F_1 = \ulcorner X_1 \urcorner, \text{ taking } A_2 = \mathbf{U}_1^3.$$

This suffices. ■

Second fixed point theorem*

LEMMA. There exists a term $\mathtt{Num} \in \Lambda$ such that for all $M \in \Lambda$

$$\mathtt{Num}\ulcorner M \urcorner =_\beta \ulcorner\ulcorner M \urcorner\urcorner$$

PROOF. Use recursion (Theorem I) for the lambda calculus data type with

$$\begin{aligned} A_1\, x\, N &= \mathtt{App}\ulcorner \mathtt{Var}\urcorner (\mathtt{Var}\ x) \\ A_2\, m\, n\, N &= \mathtt{App}\,(\mathtt{App}\ulcorner \mathtt{App}\urcorner (N\, m))(N\, n) \\ A_3\, m\, N &= \mathtt{App}\ulcorner \mathtt{Abs}\urcorner (\mathtt{Abs}\,(\lambda x.N(m\, x))) \end{aligned}$$

SECOND FIXED POINT THEOREM. For all F there is an X with

$$F\ulcorner X\urcorner =_\beta X$$

PROOF. Let $W := \lambda z.F(\mathtt{App}\ z(\mathtt{Num}\ z))$ and $X := W\ulcorner W\urcorner$. Then

$$\begin{aligned} X &= W\ulcorner W\urcorner \\ &= F(\mathtt{App}\ulcorner W\urcorner(\mathtt{Num}\ulcorner W\urcorner) = F(\mathtt{App}\ulcorner W\urcorner\ulcorner\ulcorner W\urcorner\urcorner) \\ &= F\ulcorner W\ulcorner W\urcorner\urcorner = F\ulcorner X\urcorner. \ \blacksquare \end{aligned}$$

Application 2^*

Self-modifying programs

For a given T (the program transformer) there exists a program P such that

$$\begin{aligned} P\,\mathbf{c}_k &= \mathbf{c}_{k+1} \quad \text{if } k \text{ is even,} \\ &= T\,\ulcorner P \urcorner\,\mathbf{c}_k \quad \text{otherwise.} \end{aligned}$$

- On even inputs, P performs a standard operation (adding 1)
- On odd inputs, the program P first modifies its own code using T.

To find P, apply the second fixed-point theorem to

$$F := \lambda p\, x.\texttt{if}\ (\text{Even}\, x)\ \texttt{then}\ (x+1)\ \texttt{else}(T\, p\, x)$$

Lambda-terms themselves are "black boxes"

- There is no λ-term F such that

$$F(M\,N) = M \qquad \text{for all } M, N$$

- There is no λ-term F such that

$$F(M\,N) = N \qquad \text{for all } M, N$$

On the other hand, we can compute with the codes of λ-terms

- There is a λ-term F such that $F\ulcorner M\,N\urcorner = \ulcorner M\urcorner$ for all M, N.
- There is a λ-term F such that $F\ulcorner M\,N\urcorner = \ulcorner N\urcorner$ for all M, N.
- We also have an evaluator E:

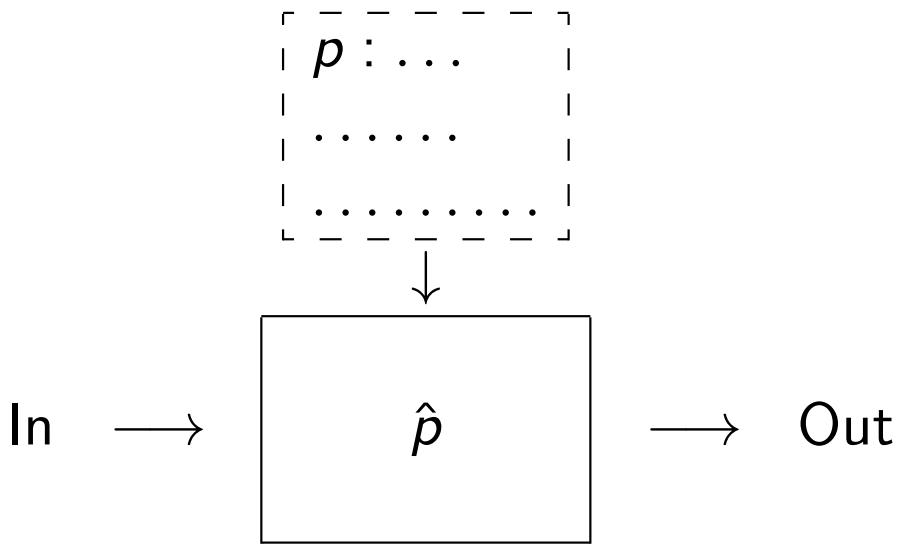
$$E\ulcorner M\urcorner = M$$

Analogy with programming

program text p $\xrightarrow{\textit{Compiler}}$ executable $\hat{p}$

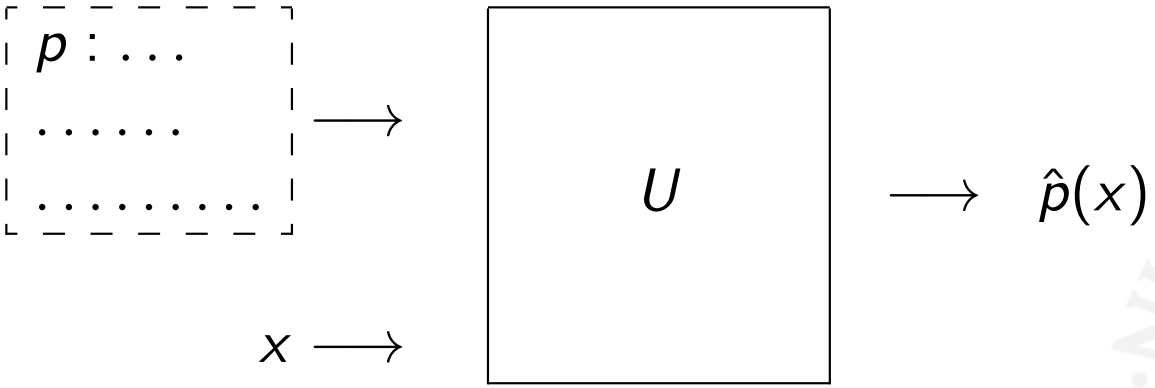
Text
- inspect,
- edit,
- ...

Black box
- input/output ("call")
- pass around



Universal **programmable** machine

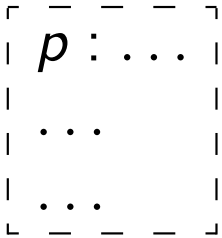
Specification of a universal (programmable) machine U:



Reflection over the programs / programming language:
What functions (programs) can we write on program text?

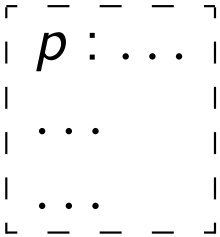
Programs about programs

EXAMPLES of programs one can write



$\longrightarrow$ M_1 $\longrightarrow$

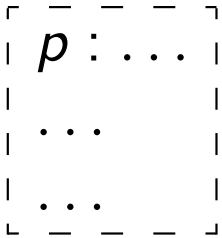
`true` if p contains a "while"
`false` if p contains no "while"



$\longrightarrow$ M_2 $\longrightarrow$

`true` if $|p| \leq 1000$
`false` if $|p| > 1000$

What about this??

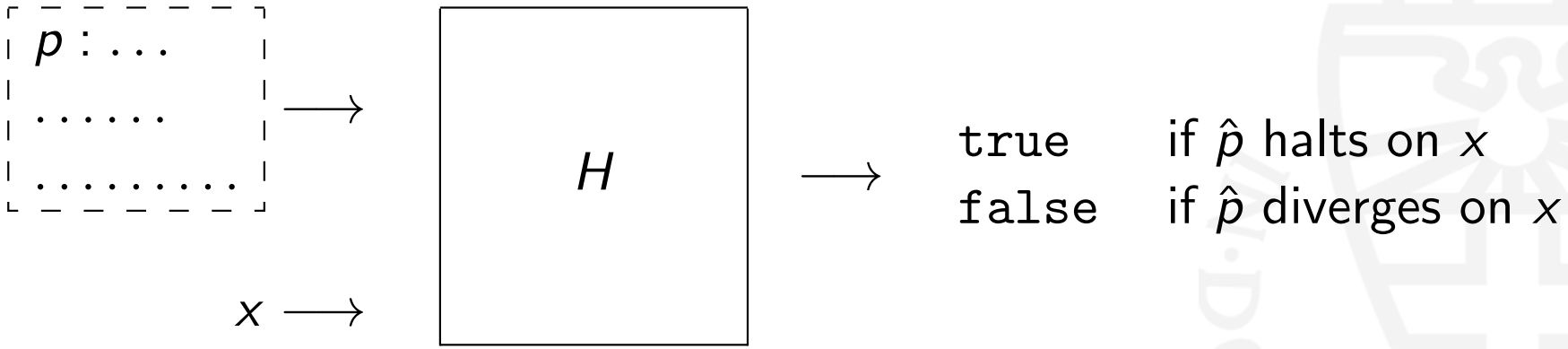


$\longrightarrow$ M_3 $\longrightarrow$

`true` if $\hat{p}$ halts on all inputs
`false` if $\hat{p}$ diverges on some input

The **Halting problem** is undecidable

THEOREM It is impossible to write a program H with the following specification



The undecidability of the Halting Problem was first proven by Turing, for his Turing machines.
We will prove it for the λ-calculus.

Remember recursion for lambda terms using encoding

THEOREM II. Given $A_1, A_2, A_3 \in \Lambda$ there is an $H \in \Lambda$ such that

$$\begin{aligned} H\,\ulcorner x \urcorner &= A_1\, x\, H \\ H\,\ulcorner M\, N \urcorner &= A_2\, \ulcorner M \urcorner\, \ulcorner N \urcorner\, H \\ H\,\ulcorner \lambda x.M \urcorner &= A_3\, (\lambda x.\ulcorner M \urcorner)\, H \end{aligned}$$

EXAMPLE
There is a λ-term C satisfying

$$\begin{aligned} C\,\ulcorner x \urcorner &= c_0 \\ C\,\ulcorner M\, N \urcorner &= c_1 \\ C\,\ulcorner \lambda x.M \urcorner &= c_2 \end{aligned}$$

PROOF Just take $A_1 := \lambda xy.c_0$, $A_2 := \lambda xyz.c_1$ and $A_3 := \lambda xy.c_2$. ■

Exercise: Write a function R that checks whether a code of a term is a redex. (So: $R\ulcorner M \urcorner = \mathbf{T}$ if M is a redex and $R\ulcorner M \urcorner = \mathbf{F}$ if M is not a redex.)

We need to compute with codes

Recall that $\mathbf{T} = \lambda x\, y.x$ and $\mathbf{F} = \lambda x\, y.y$.
The following are impossible to define with λ-terms themselves.
There is no term G satisfying

$$\begin{aligned} G\, M &= \mathbf{T} \text{ if } M \text{ has a normal form} \\ G\, M &= \mathbf{F} \text{ if } M \text{ has no normal form} \end{aligned}$$

There is no term H (compare the Halting problem) satisfying

$$\begin{aligned} H\, M\, N &= \mathbf{T} \text{ if } M\, N \text{ has a normal form} \\ H\, M\, N &= \mathbf{F} \text{ if } M\, N \text{ has no normal form} \end{aligned}$$

That these are impossible is not surprising: we can't "look inside a black box".
What if we recast these question with coded λ-terms?

The Halting problem for λ-calculus

The Halting problem is undecidable (λ-calculus version):

THEOREM There is no term H satisfying

$$\begin{aligned} H\ulcorner M\urcorner N &= \mathbf{T} \text{ if } M\,N \text{ has a normal form} \\ H\ulcorner M\urcorner N &= \mathbf{F} \text{ if } M\,N \text{ has no normal form} \end{aligned}$$

PROOF Suppose H exists, Consider

$$Q := \lambda x.H\,x\,x\,\Omega\,\mathbf{T}$$

Then

$$\begin{aligned} Q\ulcorner Q\urcorner &= H\ulcorner Q\urcorner\ulcorner Q\urcorner\,\Omega\,\mathbf{T} \\ &= \begin{cases} \mathbf{T}\Omega\,\mathbf{T} = \Omega & \text{if} \quad Q\ulcorner Q\urcorner \text{ has a normal form} \\ \mathbf{F}\Omega\,\mathbf{T} = \mathbf{T} & \text{if} \quad Q\ulcorner Q\urcorner \text{ has no normal form} \end{cases} \end{aligned}$$

Contradiction. So H doesn't exist. ■.

The "Blank tape" problem for λ-calculus

THEOREM There is no term B satisfying

$$\begin{aligned} B\ulcorner M\urcorner &= \mathbf{T} \text{ if } M \text{ has a normal form} \\ B\ulcorner M\urcorner &= \mathbf{F} \text{ if } M \text{ has no normal form} \end{aligned}$$

PROOF Suppose B exists, Consider

$$Q := \lambda x.B\,x\,\Omega\,\mathbf{T}$$

By the second fixed point theorem, there is a R such that $Q\ulcorner R\urcorner = R$, that is

$$B\ulcorner R\urcorner\,\Omega\,\mathbf{T} = R.$$

Now we have

$$\begin{aligned} R &= B\ulcorner R\urcorner\,\Omega\,\mathbf{T} \\ &= \mathbf{T}\,\Omega\,\mathbf{T} = \Omega \text{ if } R \text{ has a normal form} \\ &= \mathbf{F}\,\Omega\,\mathbf{T} = \mathbf{T} \text{ if } R \text{ has no normal form} \end{aligned}$$

Contradiction. So B doesn't exist. ■

(NB. There may be a proof by reducing H to B; I didn't see it.)

Some other terms one can(not) write

EXAMPLE There is a term L satisfying

$$L \ulcorner M \urcorner = c_n \text{ if } n \text{ is the number of } \lambda\text{'s inside } M$$

EXAMPLE There is a term V satisfying

$$V \ulcorner M \urcorner = \mathbf{T} \text{ if } M \text{ is of the shape } x\, P_1 \ldots P_n \text{ for some } n, \vec{P},$$
$$V \ulcorner M \urcorner = \mathbf{F} \text{ if } M \text{ is not of the shape } x\, P_1 \ldots P_n.$$

EXAMPLE There is no term H' satisfying

$$H' \ulcorner M \urcorner \ulcorner N \urcorner = \mathbf{T} \text{ if } M\,N \text{ has a normal form}$$
$$H' \ulcorner M \urcorner \ulcorner N \urcorner = \mathbf{F} \text{ if } M\,N \text{ has no normal form}$$

PROOF Reduce B to H'. (Show that, if H' exists, then we can also define B.)

Scott's theorem

The impossibility (undecidability) results we have seen are all instances of a general theorem due to Scott.
We phrase it here purely in terms of λ calculus.
We assume a coding function $\ulcorner - \urcorner : \Lambda \to \Lambda$ for which we have λ-terms $\mathtt{App}$ and $\mathtt{Num}$ satisfying

$$\begin{aligned} \mathtt{App}\ulcorner M\urcorner\ulcorner N\urcorner &= \ulcorner M\,N\urcorner \\ \mathtt{Num}\ulcorner M\urcorner &= \ulcorner\ulcorner M\urcorner\urcorner \end{aligned}$$

Definition Two disjoint subsets of Λ, $A, B \subseteq \Lambda$ are separable if there is a λ-term F such that $F\ulcorner M\urcorner = \mathbf{T}/\mathbf{F}$ for every M and

$$\begin{aligned} M \in A &\Rightarrow F\ulcorner M\urcorner = \mathbf{T} \\ M \in B &\Rightarrow F\ulcorner M\urcorner = \mathbf{F} \end{aligned}$$

Theorem (Scott) If A, B are non-empty and closed under $=_\beta$, then they are not separable.

Proof of Scott's theorem

Let $A, B \subseteq \Lambda$ closed under $=_\beta$ and say $a \in A$ and $b \in B$.
Suppose: F separates A and B, so $F\,\ulcorner M \urcorner = \mathbf{T}/\mathbf{F}$ for every M and

$$\begin{array}{rcl} M \in A & \Rightarrow & F\,\ulcorner M \urcorner = \mathbf{T} \\ M \in B & \Rightarrow & F\,\ulcorner M \urcorner = \mathbf{F} \end{array}$$

We define

$$H := \lambda y.\mathtt{if}\ F(\mathtt{App}\,\ulcorner y \urcorner\,(\mathtt{Num}\,y))\ \mathtt{then}\ b\ \mathtt{else}\ a$$

and we consider $J := H\,\ulcorner H \urcorner$:

$$\begin{array}{rcl} J = H\,\ulcorner H \urcorner & = & \mathtt{if}\ F(\mathtt{App}\,\ulcorner H \urcorner\,(\mathtt{Num}\,\ulcorner H \urcorner))\ \mathtt{then}\ b\ \mathtt{else}\ a \\ & = & \mathtt{if}\ F(\ulcorner H\,\ulcorner H \urcorner \urcorner)\ \mathtt{then}\ b\ \mathtt{else}\ a \\ & = & \begin{cases} b & \text{iff} \quad H\,\ulcorner H \urcorner \in A \\ a & \text{iff} \quad H\,\ulcorner H \urcorner \in B \end{cases} \end{array}$$

This is a contradiction. So F doesn't exist. ■