Lecture
Models of Computation
(DIT310, TDA184)

Nils Anders Danielsson

2016-12-05

Today

- Coding (for χ).
- Representing inductively defined sets as strings.
- Turing-computability.
- Representing Turing machines.
- A self-interpreter (a universal Turing machine).
- The halting problem.
- A Turing machine that is a χ interpreter.

Coding

$\llcorner \_ \lrcorner$

One way to give a semantics to $\llcorner \_ \lrcorner$:

- $\llcorner \_ \lrcorner$ is a constructor of a variant of Exp:

$$\frac{e \in Exp}{\llcorner e \lrcorner \in \overline{Exp}} \qquad \frac{e_1 \in \overline{Exp} \qquad e_2 \in \overline{Exp}}{\mathsf{apply}\ e_1\ e_2 \in \overline{Exp}} \qquad \cdots$$

- This variant is the domain of $\ulcorner \_ \urcorner$:

$$\begin{aligned} \ulcorner \_ \urcorner &\in \overline{Exp} \to Exp \\ \ulcorner \llcorner e \lrcorner \urcorner &= e \\ \ulcorner \mathsf{apply}\ e_1\ e_2 \urcorner &= \mathsf{Apply}(\ulcorner e_1 \urcorner, \ulcorner e_2 \urcorner) \\ &\vdots \end{aligned}$$

$\llcorner - \lrcorner$

- Example:

$$\ulcorner eval \llcorner code\ e \lrcorner \urcorner = \mathsf{Apply}(\ulcorner eval \urcorner, code\ e)$$

- Note that you do not have to use $\llcorner - \lrcorner$.

Coding

Probably not what you want:

$$\lambda p. \ulcorner eval\ p \urcorner = \lambda p.\ \mathsf{Apply}(\ulcorner eval \urcorner, \mathsf{Var}(\ulcorner p \urcorner))$$

If p corresponds to 0:

$$\lambda p.\ \mathsf{Apply}(\ulcorner eval \urcorner, \mathsf{Var}(\mathsf{Zero}()))$$

A constant function.

Coding

Perhaps more useful:

$$\lambda p.\, \ulcorner eval \llcorner code\ p \lrcorner \urcorner = \lambda p.\, \mathsf{Apply}(\ulcorner eval \urcorner, code\ p)$$

For any expression e:

$$(\lambda p.\, \ulcorner eval \llcorner code\ p \lrcorner \urcorner)\, \ulcorner e \urcorner \Downarrow \ulcorner eval\, \ulcorner e \urcorner \urcorner$$

Quiz

What is the result of evaluating
$(\lambda p.\ eval \ulcorner eval \llcorner code\ p \lrcorner \urcorner) \ulcorner \mathsf{Zero}() \urcorner$?

- Nothing
- Zero()
- $\ulcorner$ Zero() $\urcorner$
- $\ulcorner\ulcorner$ Zero() $\urcorner\urcorner$
- $\ulcorner\ulcorner\ulcorner$ Zero() $\urcorner\urcorner\urcorner$
- $\ulcorner\ulcorner\ulcorner\ulcorner$ Zero() $\urcorner\urcorner\urcorner\urcorner$

Representing inductively defined sets

Natural numbers

One method:

$$
\begin{aligned}
&\ulcorner \_ \urcorner \in \mathbb{N} \to List\ \{1\} \\
&\ulcorner \mathsf{zero} \urcorner \quad = [\,] \\
&\ulcorner \mathsf{suc}\ n \urcorner = 1 :: \ulcorner n \urcorner
\end{aligned}
$$

Natural numbers

Another method:

$$
\begin{aligned}
&\ulcorner \_ \urcorner \in \mathbb{N} \to \mathit{List}\ \{0, 1\} \\
&\ulcorner \mathsf{zero} \urcorner \ \ = 0 :: [\,] \\
&\ulcorner \mathsf{suc}\ n \urcorner = 1 :: \ulcorner n \urcorner
\end{aligned}
$$

This method is used below.

Lists

Assume that A can be represented using a function $\ulcorner \_ \urcorner \in A \to \mathit{List}\ \Sigma$ which satisfies the following properties:

- It is injective.
- There is a function

  $$\mathit{split} \in \mathit{List}\ \Sigma \to \mathit{List}\ \Sigma \times \mathit{List}\ \Sigma$$

  such that, for any $x \in A$, $xs \in \mathit{List}\ \Sigma$,

  $$\mathit{split}\ (\ulcorner x \urcorner +\!\!+\ xs) = (\ulcorner x \urcorner, xs).$$

Lists

Assume that A can be represented using a function $\ulcorner \_ \urcorner \in A \to \mathit{List}\ \Sigma$ which satisfies the following properties:

- It is injective.
- There is a function

  $$\mathit{split} \in \mathit{List}\ \Sigma \to \mathit{List}\ \Sigma \times \mathit{List}\ \Sigma$$

  such that, for any $x \in A$, $xs \in \mathit{List}\ \Sigma$,

  $$\mathit{split}\ (\ulcorner x \urcorner +\!\!+\ xs) = (\ulcorner x \urcorner, xs).$$

Note that split can only be defined for one of the presented methods for representing natural numbers.

Lists

Representation of $List\ A$:

$$\begin{aligned}
\ulcorner _ \urcorner &\in List\ A \to List\ (\Sigma \cup \{0, 1\}) \\
\ulcorner [\,] \urcorner &= 0 :: [\,] \\
\ulcorner x :: xs \urcorner &= 1 :: \ulcorner x \urcorner +\!\!+ \ulcorner xs \urcorner
\end{aligned}$$

This function also satisfies the given properties.

Quiz

Which list of natural numbers does 11110101110100 stand for?

- None
- $[3, 0, 2]$
- $[3, 0, 2, 0]$
- $[3, 2, 0]$
- $[4, 1, 3, 1]$
- $[4, 1, 3, 1, 0]$

Turing-computability

Turing-computable functions

Assume that we have methods for representing members of the sets A and B as elements of $List\ \Sigma$, where Σ is a finite set.

A partial function $f \in A \rightharpoonup B$ is *Turing-computable* if there is a Turing machine tm such that:

- $\Sigma_{tm} = \Sigma$.
- $\forall a \in A.\ [\![tm]\!] \ulcorner a \urcorner = \ulcorner f\ a \urcorner$.

Languages

- A language over an alphabet Σ is a subset of $List\ \Sigma$.

Turing-decidable

A language L over Σ is *Turing-decidable* if there is a Turing machine tm (with accepting states) such that:

- $\Sigma_{tm} = \Sigma$.
- $\forall xs \in List\ \Sigma$. if $xs \in L$ then $Accept_{tm}\ xs$.
- $\forall xs \in List\ \Sigma$. if $xs \notin L$ then $Reject_{tm}\ xs$.

Turing-recognisable

A language L over Σ is *Turing-recognisable* if there is a Turing machine tm (with accepting states) such that:

- $\Sigma_{tm} = \Sigma$.
- $\forall xs \in List\ \Sigma.\ xs \in L$ iff $Accept_{tm}\ xs$.

Representing Turing machines

States

Assume that $S = \{ s_0, \ldots, s_n \}$.
Note that S is always non-empty.

$$\ulcorner S \urcorner = \ulcorner n \urcorner$$
$$\ulcorner s_k \urcorner = \ulcorner k \urcorner$$

Alphabets

Assume that $\Sigma = \{ c_1, \ldots, c_m \}$ and $\Gamma = \{ \sqcup \} \cup \{ c_1, \ldots, c_{m+n} \}$.

$$\begin{aligned} \ulcorner \Sigma \urcorner &= \ulcorner m \urcorner \\ \ulcorner \Gamma \urcorner &= \ulcorner n \urcorner \\ \ulcorner \sqcup \urcorner &= \ulcorner 0 \urcorner \\ \ulcorner c_k \urcorner &= \ulcorner k \urcorner \end{aligned}$$

Directions

$$\ulcorner \mathsf{L} \urcorner = [0]$$
$$\ulcorner \mathsf{R} \urcorner = [1]$$

The transition function

- A rule $\delta\,(s,x) = (s', x', d)$ is represented by
$$\ulcorner s \urcorner \mathbin{+\!\!+} \ulcorner x \urcorner \mathbin{+\!\!+} \ulcorner s' \urcorner \mathbin{+\!\!+} \ulcorner x' \urcorner \mathbin{+\!\!+} \ulcorner d \urcorner.$$
- The transition function is represented by the representation of a list containing all of its rules (ordered in some way).

Turing machines and strings

- A Turing machine $(S, s_0, \Sigma, \Gamma, \delta) \in TM$ is represented by

$$\ulcorner S \urcorner +\!\!+ \ulcorner s_0 \urcorner +\!\!+ \ulcorner \Sigma \urcorner +\!\!+ \ulcorner \Gamma \urcorner +\!\!+ \ulcorner \delta \urcorner.$$

- A pair consisting of a Turing machine tm and a corresponding input string xs is represented by

$$\ulcorner tm \urcorner +\!\!+ \ulcorner xs \urcorner.$$

- Note that this encoding only uses two non-blank symbols, 0 and 1.

Quiz

What Turing machine does $000110010011101010110001110101010001$ represent?

- None
- $S = \{s_0\}$, $\Sigma = \{0\}$, $\Gamma = \{0, \sqcup\}$, $\delta(s_0, 0) = (s_0, 0, \mathsf{L})$
- $S = \{s_0\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \sqcup\}$, $\delta(s_0, 0) = (s_0, 1, \mathsf{R})$

Self-interpreter

Self-interpreter

A self-interpreter or *universal Turing machine* $eval$ is a witness to the fact that $[\![\_]\!]$ is Turing-computable:

$$\Sigma_{eval} = \{0, 1\}$$

$$\forall\ tm \in TM.\ \forall\ xs \in List\ \Sigma_{tm}.$$
$$[\![eval]\!]\ \ulcorner (tm, xs) \urcorner = \ulcorner [\![tm]\!]\ xs \urcorner$$

Implementation sketch

Possibly buggy:

- Let us use three tapes in the implementation.
  Can convert to a one-tape machine later.
- Mark the left end of the input tape.
  Convert to a two-symbol machine later.
- Move the input string to the second tape.
  Mark the left end and the head's position.
- Write the initial state to the third tape.
  Mark the left end.

Implementation sketch

- Simulate the input TM,
  using the rules on the first tape.
- If the simulation halts successfully
  (with the head at the start of its tape),
  write the result to the first tape and
  halt successfully.
- If the simulation halts unsuccessfully,
  halt unsuccessfully.

The halting problem

The halting problem

$$
\begin{aligned}
&\mathit{halts} \in \{\, (\mathit{tm}, \mathit{xs}) \mid \mathit{tm} \in \mathit{TM}, \mathit{xs} \in \mathit{List}\ \Sigma_{\mathit{tm}} \,\} \to \mathit{Bool} \\
&\mathit{halts}\ (\mathit{tm}, \mathit{xs}) = \\
&\quad \textbf{if}\ \exists\ \mathit{ys} \in \mathit{List}\ \Gamma_{\mathit{tm}}.\ [\![\mathit{tm}]\!]\ \mathit{xs} = \mathit{ys}\ \textbf{then} \\
&\qquad \mathsf{true} \\
&\quad \textbf{else} \\
&\qquad \mathsf{false}
\end{aligned}
$$

This function is not Turing-computable.

The halting problem

The halting problem can also be viewed as a language:

$$\begin{aligned}\{ \ulcorner (tm, xs) \urcorner \mid {} & tm \in TM, \\ & xs \in List\ \Sigma_{tm}, \\ & ys \in List\ \Gamma_{tm}, \\ & [\![tm]\!]\ xs = ys \}\end{aligned}$$

This language is Turing-undecidable.

The halting problem (with self-application)

$$\{ \ulcorner tm \urcorner \mid tm \in TM, ys \in List\ \Gamma_{tm}, [\![tm]\!]\ \ulcorner tm \urcorner = ys \}$$

This language is Turing-undecidable. Proof sketch:

- Assume that the TM $halts$ decides it.
- Define a TM $terminv$ in the following way:
  - Simulate $halts$ on the input.
  - If $halts$ accepts, loop forever.
  - If $halts$ rejects, halt with a result.
- Note that $terminv$ applied to $\ulcorner terminv \urcorner$ halts iff it does not halt.

The halting problem is undecidable

$$\{\ulcorner (tm, xs) \urcorner \mid tm \in TM, xs \in List\ \Sigma_{tm}, \\ ys \in List\ \Gamma_{tm}, [\![tm]\!]\ xs = ys \}$$

Proof sketch:

- Assume that the TM *halts* decides it.
- We can then implement a TM for the halting problem with self-application:
  - If the input is not $\ulcorner tm \urcorner$ for some $tm \in TM$, reject.
  - If it is $\ulcorner tm \urcorner$, write ??? on the tape.
  - Run *halts*.

Quiz

What does ??? stand for?

- tm
- $\ulcorner tm \urcorner$
- $\ulcorner\ulcorner tm \urcorner\urcorner$
- $tm +\!\!+ \ulcorner tm \urcorner$
- $\ulcorner tm \urcorner +\!\!+ \ulcorner\ulcorner tm \urcorner\urcorner$
- $tm +\!\!+ \ulcorner tm \urcorner +\!\!+ \ulcorner\ulcorner tm \urcorner\urcorner$

$\mathcal{X}$ interpreter

A χ interpreter

The χ semantics is Turing-computable:

- X programs can be represented as strings in some finite alphabet Σ:

$$\ulcorner \_ \urcorner^{\mathsf{TM}} \in \mathit{CExp} \to \mathit{List}\ \Sigma$$

- There is a TM chi satisfying the following properties:

$$\Sigma_{\mathit{chi}} = \Sigma$$

$$\forall\ e \in \mathit{CExp}.\ [\![\mathit{chi}]\!]_{\mathsf{TM}}\ \ulcorner e \urcorner^{\mathsf{TM}} = \ulcorner [\![e]\!]_{\chi} \urcorner^{\mathsf{TM}}$$

Recursion

- How can recursion be implemented?
- One idea: An explicit stack on a separate tape.

Implementation sketch

- Come up with a small-step semantics for χ.
- Use small steps also for substitution.
- Make sure that every small step can be simulated on a TM.
- The design can be based on some abstract machine for the λ-calculus, perhaps the CEK machine.

Every χ-computable partial function in $\mathbb{N} \rightharpoonup \mathbb{N}$ is Turing-computable

Proof sketch:

- If $f \in \mathbb{N} \rightharpoonup \mathbb{N}$ is χ-computable, then

  $$\forall\, m \in \mathbb{N}.\ [\![e \ulcorner m \urcorner^{\chi}]\!]_{\chi} = \ulcorner f\ m \urcorner^{\chi}$$

  for some $e \in CExp$.
- The following TM implements f:
  - Convert input: $\ulcorner m \urcorner^{\mathsf{TM}} \mapsto \ulcorner e \ulcorner m \urcorner^{\chi} \urcorner^{\mathsf{TM}}$.
  - Simulate the χ interpreter.
  - Convert output: $\ulcorner \ulcorner n \urcorner^{\chi} \urcorner^{\mathsf{TM}} \mapsto \ulcorner n \urcorner^{\mathsf{TM}}$.

Summary

- Coding (for χ).
- Representing inductively defined sets as strings.
- Turing-computability.
- Representing Turing machines.
- A self-interpreter (a universal Turing machine).
- The halting problem.
- A Turing machine that is a χ interpreter.