Lecture
Models of Computation
(DIT310, TDA184)

Nils Anders Danielsson

2016-11-28

Today

- A comment about types.
- Rice's theorem.
- Turing machines.

Types

Types

- The language χ is untyped.
- However, it may be instructive to see certain programs as typed.

Types

- $Rep\ A$: Representations of programs of type A.
- Some examples:

$$\begin{array}{ll}
\mathsf{Zero}() & : \mathbb{N} \\
\ulcorner \mathsf{Zero}() \urcorner & : Rep\ \mathbb{N} \\
\ulcorner \mathsf{zero} \urcorner & : \mathbb{N} \\
\lambda f.\ \lambda x.\ f\ x & : (A \to B) \to A \to B \\
\lambda f.\ \lambda x.\ \mathsf{Apply}(f, x) & : Rep\ (A \to B) \to \\
 & \quad Rep\ A \to Rep\ B \\
eval & : Rep\ A \to Rep\ A \\
code & : Rep\ A \to Rep\ (Rep\ A) \\
terminates\text{-}in & : Rep\ A \times \mathbb{N} \to Bool \\
\ulcorner terminates\text{-}in \urcorner & : Rep\ (Rep\ A \times \mathbb{N} \to Bool)
\end{array}$$

Types

A reduction from last week:

$$halts = \lambda p.\ not\ (pointwise\text{-}equal' \\ \ulcorner \lambda n.\ terminates\text{-}in\ \mathsf{Pair}(\llcorner\ code\ p\ \lrcorner, n)\ \urcorner \\ \ulcorner \lambda \_.\ \mathsf{False}()\ \urcorner)$$

Expanded:

$$\lambda p.\ not\ (pointwise\text{-}equal' \\ \mathsf{Lambda}(\ulcorner n \urcorner, \\ \mathsf{Apply}(\ulcorner terminates\text{-}in \urcorner, \\ \mathsf{Const}(\ulcorner \mathsf{Pair} \urcorner, \\ \mathsf{Cons}(code\ p, \\ \mathsf{Cons}(\mathsf{Var}(\ulcorner n \urcorner), \mathsf{Nil}())))))) \\ \ulcorner \lambda \_.\ \mathsf{False}()\ \urcorner)$$

Types

If

$$\begin{aligned}&\mathit{pointwise\text{-}equal}' :\\&\quad \mathit{Rep}\ (\mathbb{N} \to \mathit{Bool}) \times \mathit{Rep}\ (\mathbb{N} \to \mathit{Bool}) \to \mathit{Bool}\end{aligned}$$

then

$$\mathit{halts} : \mathit{Rep}\ A \to \mathit{Bool}.$$

Rice's theorem

Rice's theorem

Assume that $P \in CExp \rightarrow Bool$ satisfies the following properties:

- P is non-trivial:
  There are expressions e_{true}, $e_{\text{false}} \in CExp$ satisfying $P\ e_{\text{true}} = \text{true}$ and $P\ e_{\text{false}} = \text{false}$.
- P respects pointwise semantic equality:

$$\begin{array}{l} \forall\ e_1, e_2 \in CExp. \\ \quad \text{if } \forall\ e \in CExp.\ [\![e_1\ e]\!] = [\![e_2\ e]\!] \text{ then} \\ \qquad P\ e_1 = P\ e_2 \end{array}$$

Then P is χ-undecidable.

Rice's theorem

The halting problem reduces to P:

$$
\begin{aligned}
& halts = \lambda\, e.\ \mathbf{case}\ P \ulcorner \lambda\, \_.\ \mathbf{rec}\ x = x \urcorner\ \mathbf{of} \\
& \quad \{ \mathsf{False}() \rightarrow \\
& \qquad P \ulcorner \lambda\, x.\ (\lambda\, \_.\ e_{\mathsf{true}}\ x)\ (eval \llcorner\ code\ e \lrcorner) \urcorner \\
& \quad ;\ \mathsf{True}()\ \rightarrow \\
& \qquad not\ (P \ulcorner \lambda\, x.\ (\lambda\, \_.\ e_{\mathsf{false}}\ x)\ (eval \llcorner\ code\ e \lrcorner) \urcorner) \\
& \quad \}
\end{aligned}
$$

Quiz

Which of the following problems are χ-decidable?

- Is $e \in CExp$ an implementation of the successor function for natural numbers?
- Is $e \in CExp$ syntactically equal to $\lambda\, n.\ \mathsf{Succ}(n)$?

Turing machines

Intuitive idea

- A tape that extends arbitrarily far to the right.
- The tape is divided into squares.
- The squares can contain symbols, chosen from a finite alphabet.
- A read/write head, positioned over one square.
- The head can move from one square to an adjacent one.
- Rules that explain what the head does.

Rules

- A finite set of states.
- When the head reads a symbol
  (blank squares correspond to a special symbol):
  - Check if the current state contains a matching rule, with:
    - A symbol to write.
    - A direction to move in.
    - A state to switch to.
  - If not, halt.

Motivation

- Turing motivated his design partly by reference to what a human computer does.
- Please read his text.

Abstract syntax

Abstract syntax

A Turing machine (one variant) is specified by giving the following information:

- S: A finite set of states.
- $s_0 \in S$: An initial state.
- Σ: The input alphabet,
  a finite set of symbols with $\sqcup \notin \Sigma$.
- Γ: The tape alphabet,
  a finite set of symbols with $\Sigma \cup \{\sqcup\} \subseteq \Gamma$.
- $\delta \in S \times \Gamma \rightharpoonup S \times \Gamma \times \{\mathsf{L}, \mathsf{R}\}$:
  The transition "function".

Abstract syntax

$$\frac{\begin{array}{cc} S \text{ is a finite set} & s_0 \in S \\ \Sigma \text{ is a finite set} & \sqcup \notin \Sigma \\ \Gamma \text{ is a finite set} & \Sigma \cup \{\sqcup\} \subseteq \Gamma \\ \multicolumn{2}{c}{\delta \in S \times \Gamma \rightharpoonup S \times \Gamma \times \{\mathsf{L}, \mathsf{R}\}} \end{array}}{(S, s_0, \Sigma, \Gamma, \delta) \in TM}$$

Operational semantics

Positioned tapes

- Representation of the tape and the head's position:

$$Tape = List\ \Gamma \times List\ \Gamma$$

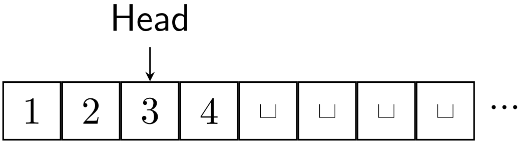
- Here (ls, rs) stands for

$$reverse\ ls \mathbin{+\!\!+} rs$$

followed by an infinite sequence of blanks ($\sqcup$).

Positioned tapes

$([2, 1], [3, 4, \sqcup, \sqcup])$ stands for:



The symbol under the head

The head is located over the first symbol in rs (or a blank, if rs is empty):

$$
\begin{aligned}
&head_T \in Tape \to \Gamma \\
&head_T\ (ls, rs) = head\ rs
\end{aligned}
$$

$$
\begin{aligned}
&head \in List\ \Gamma \to \Gamma \\
&head\ [\,] \qquad\quad = \sqcup \\
&head\ (x :: xs) = x
\end{aligned}
$$

Writing

Writing to the tape:

$$
\begin{aligned}
&\mathit{write} \in \Gamma \rightarrow \mathit{Tape} \rightarrow \mathit{Tape} \\
&\mathit{write}\ x\ (\mathit{ls}, \mathit{rs}) = (\mathit{ls}, x :: \mathit{tail}\ \mathit{rs})
\end{aligned}
$$

The “tail” of a sequence:

$$
\begin{aligned}
&\mathit{tail} \in \mathit{List}\ \Gamma \rightarrow \mathit{List}\ \Gamma \\
&\mathit{tail}\ [\,] \qquad\quad\ = [\,] \\
&\mathit{tail}\ (r :: \mathit{rs}) = \mathit{rs}
\end{aligned}
$$

Moving

Moving the head:

$$
\begin{aligned}
&move \in \{\mathsf{L}, \mathsf{R}\} \to Tape \to Tape \\
&move\ \mathsf{R}\ (ls, rs) = (head\ rs :: ls, tail\ rs) \\
&move\ \mathsf{L}\ ([\,], rs) = ([\,] \qquad\qquad\quad , rs) \\
&move\ \mathsf{L}\ (ls, rs) = (tail\ ls \qquad\quad , head\ ls :: rs)
\end{aligned}
$$

Actions

Actions describe what the head will do:

$$Action = \Gamma \times \{\mathsf{L}, \mathsf{R}\}$$

Note:

$$\delta \in S \times \Gamma \rightharpoonup S \times Action$$

First write, then move:

$$act \in Action \to Tape \to Tape$$
$$act\ (x, d)\ t = move\ d\ (write\ x\ t)$$

Quiz

Which of the following equalities are valid?

- $act\ (0, \mathsf{L})\ (act\ (1, \mathsf{L})\ ([\,], [\,])) = ([\,], [0, 1])$
- $act\ (0, \mathsf{L})\ (act\ (1, \mathsf{L})\ ([\,], [\,])) = ([0, 1], [\,])$
- $act\ (0, \mathsf{L})\ (act\ (1, \mathsf{L})\ ([\,], [\,])) = ([1, 0], [\,])$
- $act\ (0, \mathsf{R})\ (act\ (1, \mathsf{R})\ ([\,], [\,])) = ([\,], [0, 1])$
- $act\ (0, \mathsf{R})\ (act\ (1, \mathsf{R})\ ([\,], [\,])) = ([0, 1], [\,])$
- $act\ (0, \mathsf{R})\ (act\ (1, \mathsf{R})\ ([\,], [\,])) = ([1, 0], [\,])$

Small-step operational semantics

A configuration consists of a state and a tape:

$$Configuration = State \times Tape$$

The small-step operational semantics relates configurations:

$$\frac{\delta\ (s, head_T\ t) = (s', a)}{(s, t) \longrightarrow (s', act\ a\ t)}$$

Reflexive transitive closure

Zero or more small steps:

$$\frac{}{c \longrightarrow^\star c} \qquad \frac{c_1 \longrightarrow c_2 \qquad c_2 \longrightarrow^\star c_3}{c_1 \longrightarrow^\star c_3}$$

The machine halts if it ends up in a configuration c for which there is no c' such that $c \longrightarrow c'$.

The machine's result

- The machine is started in state s_0.
- The head is initially over the left-most square.
- The tape initially contains a string of characters from the input alphabet Σ (followed by blanks).
- If the machine halts with the head in the left-most square, then the result consists of the contents of the tape, up to the last non-blank symbol.

The machine's result

A relation between $List\ \Sigma$ and $List\ \Gamma$:

$$\frac{(s_0, [\,], xs) \longrightarrow^\star (s, [\,], rs) \qquad \nexists c.\ (s, [\,], rs) \longrightarrow c \qquad remove\ rs = ys}{xs \Downarrow ys}$$

Removing blanks

The function *remove* removes all trailing blanks:

$$\begin{aligned} &remove \in List\ \Gamma \to List\ \Gamma \\ &remove\ [\,] \qquad\qquad = [\,] \\ &remove\ (x :: xs) = cons'\ x\ (remove\ xs) \end{aligned}$$

$$\begin{aligned} &cons' \in \Gamma \to List\ \Gamma \to List\ \Gamma \\ &cons'\ \sqcup\ [\,]\ = [\,] \\ &cons'\ x\ xs = x :: xs \end{aligned}$$

Quiz

Which properties does $\Downarrow$ satisfy?

- Is it deterministic (for every Turing machine)?

$$\forall\, xs \in \mathit{List}\ \Sigma.\ \ \forall\, ys, zs \in \mathit{List}\ \Gamma.\\ xs \Downarrow ys \wedge xs \Downarrow zs \Rightarrow ys = zs$$

- Is it total (for every Turing machine)?

$$\forall\, xs \in \mathit{List}\ \Sigma.\ \ \exists\, ys \in \mathit{List}\ \Gamma.\ xs \Downarrow ys$$

The machine's partial function

The semantics as a partial function:

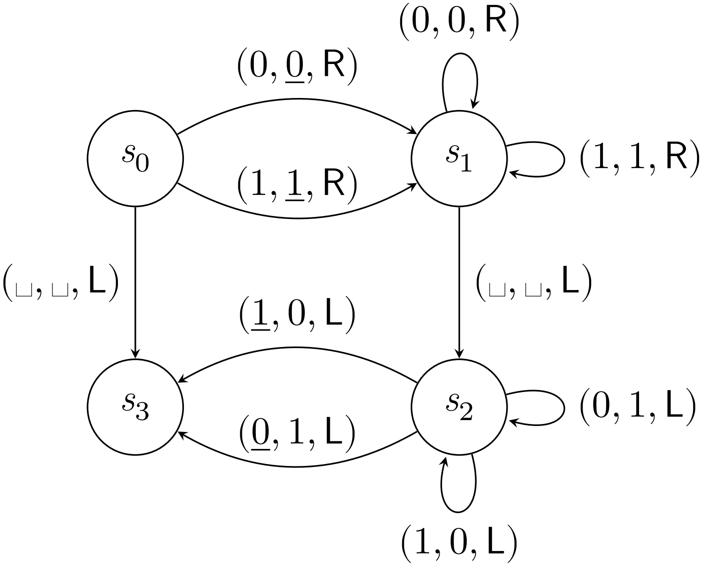
$$\llbracket \_ \rrbracket \in \forall\, tm \in TM.\, List\ \Sigma_{tm} \rightharpoonup List\ \Gamma_{tm}$$
$$\llbracket tm \rrbracket\ xs = ys \ \text{ if } xs \Downarrow_{tm} ys$$

An example

An example

- Input alphabet: $\{0, 1\}$.
- Tape alphabet: $\{0, 1, \underline{0}, \underline{1}, \sqcup\}$.
- States: $\{s_0, s_1, s_2, s_3\}$.
- Initial state: s_0.

Transition function



Quiz

What is the result of running this TM with 0101 as the input string?

- No result
- 0000
- 1111
- 0101
- 1010
- $\underline{0}101$
- $\underline{1}010$

Accepting states

Accepting states

Turing machines with *accepting states*:

$$\frac{\begin{array}{c} S \text{ is a finite set} \qquad s_0 \in S \qquad A \subseteq S \\ \Sigma \text{ is a finite set} \qquad \sqcup \notin \Sigma \\ \Gamma \text{ is a finite set} \qquad \Sigma \cup \{\sqcup\} \subseteq \Gamma \\ \delta \in S \times \Gamma \rightharpoonup S \times \Gamma \times \{\mathsf{L}, \mathsf{R}\} \end{array}}{(S, s_0, A, \Sigma, \Gamma, \delta) \in TM}$$

Is the string accepted?

A relation on $List\ \Sigma$:

$$\frac{(s_0, [\,], xs) \longrightarrow^{\star} (s, t) \qquad \nexists c.\ (s, t) \longrightarrow c \qquad s \in A}{Accept\ xs}$$

Is the string rejected?

A relation on $\mathit{List}\ \Sigma$:

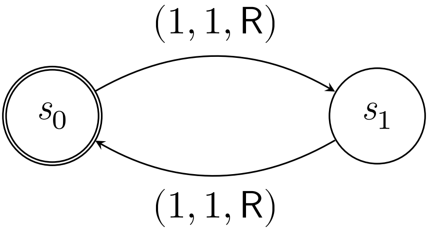
$$\frac{(s_0, [\,], xs) \longrightarrow^{\star} (s, t) \qquad \nexists c.\ (s, t) \longrightarrow c \qquad s \notin A}{\mathit{Reject}\ xs}$$

Note that if the TM fails to halt, then the string is neither accepted nor rejected.

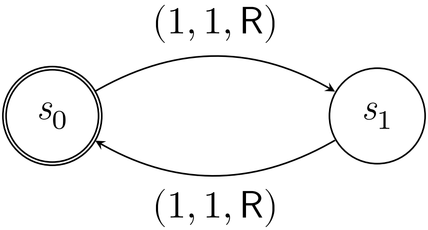
An example

- Input alphabet: $\{1\}$.
- Tape alphabet: $\{1, \sqcup\}$.
- States: $\{s_0, s_1\}$.
- Initial state: s_0.
- Accepting states: $\{s_0\}$.

Transition function



Transition function



- Quiz: Which strings are accepted by this Turing machine?

Variants

Variants

Equivalent (in some sense) variants:

- Possibility to stay put.
- A tape without a left end.
- Multiple tapes.
- Only two symbols, other than the blank one.

Representing inductively defined sets

Natural numbers

One method:

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

Natural numbers

Another method (for $z \neq s$):

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

Lists

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

- It is injective.
- There is a function

  $$split \in List\ \Sigma \rightarrow List\ \Sigma \times List\ \Sigma$$

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

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

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 $List\ A$ (for $n \neq c$):

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

This function also satisfies the given properties.

Quiz

Let n and z both stand for 0, and let s and c both stand for 1. 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 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 such that:

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

Summary

- A comment about types.
- Rice's theorem.
- Turing machines:
  - Abstract syntax.
  - Operational semantics.
  - Variants.
  - Representing inductively defined sets.
  - Turing-computability.