Lecture Computability (DIT312, DAT415)

Nils Anders Danielsson

2019-12-02

Today

- Rice's theorem.
- Turing machines.

Rice's theorem

A variant of Rice's theorem

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

- P is non-trivial:
  There are expressions e_{true}, $e_{\mathsf{false}} \in CExp$ satisfying $P\ e_{\mathsf{true}} =$ true and $P\ e_{\mathsf{false}} =$ 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:

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

Quiz

Which of the following problems are χ-decidable?

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

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:

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

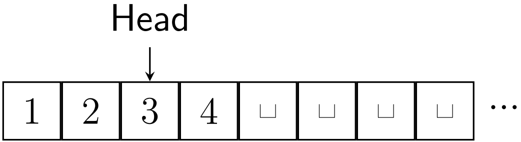
- Here (ls, rs) stands for

$$reverse\ ls +\!\!+\ rs$$

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:

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

The "tail" of a sequence:

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

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?

1. $act\,(0, \mathsf{L})\;(act\,(1, \mathsf{L})\;([\,], [\,])) = ([\,], [0, 1])$
2. $act\,(0, \mathsf{L})\;(act\,(1, \mathsf{L})\;([\,], [\,])) = ([0, 1], [\,])$
3. $act\,(0, \mathsf{L})\;(act\,(1, \mathsf{L})\;([\,], [\,])) = ([1, 0], [\,])$
4. $act\,(0, \mathsf{R})\;(act\,(1, \mathsf{R})\;([\,], [\,])) = ([\,], [0, 1])$
5. $act\,(0, \mathsf{R})\;(act\,(1, \mathsf{R})\;([\,], [\,])) = ([0, 1], [\,])$
6. $act\,(0, \mathsf{R})\;(act\,(1, \mathsf{R})\;([\,], [\,])) = ([1, 0], [\,])$

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, then the result consists of the contents of the tape, up to the last non-blank symbol.
- (In 2016/2017 I required the machine to halt with the head over the left-most square.)

The machine's result

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

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

Constructing the result

The function list converts the representation of the tape to a list, and remove removes all trailing blanks:

$$
\begin{aligned}
&\mathit{list} \in \mathit{Tape} \to \mathit{List}\,\Gamma \\
&\mathit{list}\,(\mathit{ls}, \mathit{rs}) = \mathit{reverse}\;\mathit{ls} +\!\!+\; \mathit{rs}
\end{aligned}
$$

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

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

Quiz

Which properties does $\Downarrow$ satisfy?

1. Is it deterministic (for every Turing machine)?

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

2. 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:

$$
\begin{array}{l}
[\![\_]\!] \in \ \forall\, tm \in \ TM.\, List\, \Sigma_{tm} \rightharpoonup List\, \Gamma_{tm} \\
[\![tm]\!]\; xs = ys \ \text{ if } xs \Downarrow_{tm} ys
\end{array}
$$

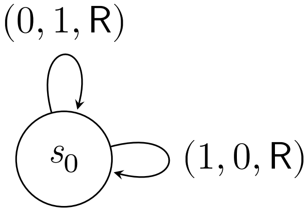
Two examples

An example

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

Transition function

$$\delta\left(s_0, 0\right) = \left(s_0, 1, \mathsf{R}\right)$$
$$\delta\left(s_0, 1\right) = \left(s_0, 0, \mathsf{R}\right)$$



Quiz

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

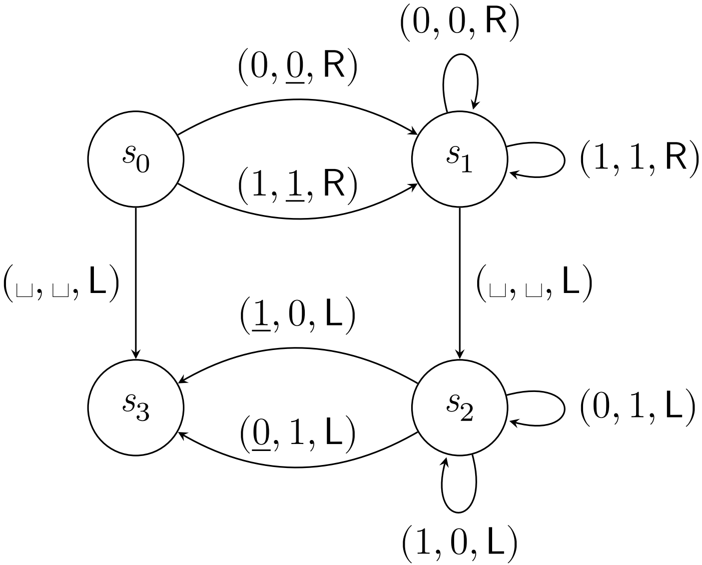
1. No result
2. 0000
3. 1111
4. 0101
5. 1010
6. $0101\sqcup$
7. $1010\sqcup$

Another example

One way to make sure that the head ends up over the left-most square:

- 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



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 $\mathit{List}\,\Sigma$:

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

Is the string rejected?

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

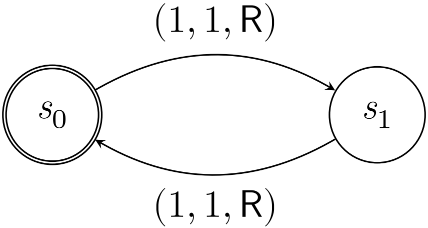
$$\frac{(s_0, [\,], xs) \longrightarrow^{\star} (s, t) \qquad \nexists c.\,(s, t) \longrightarrow c \qquad s \notin A}{\mathit{Reject}\ xs}$$

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

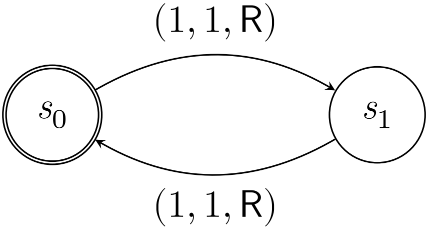
An example

- 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:

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

This method is used below.

Lists

Assume that members of A can be represented using a function $\ulcorner \_ \urcorner \in A \rightarrow \mathit{List}\ \Xi$ that is *splittable*:

- It is injective.
- There is a function

  $$\mathit{split} \in \mathit{List}\ \Xi \rightarrow \mathit{List}\ \Xi \times \mathit{List}\ \Xi$$

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

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

Lists

Assume that members of A can be represented using a function $\ulcorner \_ \urcorner \in A \to \mathit{List}\,\Xi$ that is *splittable*:

- It is injective.
- There is a function

  $$\mathit{split} \in \mathit{List}\,\Xi \to \mathit{List}\,\Xi \times \mathit{List}\,\Xi$$

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

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

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

Lists

Representation of $List\ A$:

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

This function is splittable.

Quiz

Which list of natural numbers does 11110101110100 stand for?

1. None
2. $[3, 0, 2]$
3. $[3, 0, 2, 0]$
4. $[3, 2, 0]$
5. $[4, 1, 3, 1]$
6. $[4, 1, 3, 1, 0]$

Pairs

Assume that members of A and B can be represented using functions $\ulcorner\_\urcorner^A \in A \to \mathit{List}\,\Xi$ and $\ulcorner\_\urcorner^B \in B \to \mathit{List}\,\Xi$ that are splittable.

Representation of $A \times B$:

$$\begin{aligned}
&\ulcorner_\urcorner \in A \times B \to \mathit{List}\,\Xi \\
&\ulcorner (x, y) \urcorner = \ulcorner x \urcorner^A +\!\!+ \ulcorner y \urcorner^B
\end{aligned}$$

This function is also splittable.

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* (with respect to these methods) 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

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