

Lecture
Computability
(DIT313, DAT415)

Nils Anders Danielsson

2022-11-28

Today

- ▶ Rice's theorem.
- ▶ Turing machines.

Rice's theorem

A variant of Rice's theorem

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

- ▶ P is non-trivial:

There are expressions $e_{\text{true}}, e_{\text{false}} \in CExp$ satisfying $P e_{\text{true}} = \text{true}$ and $P e_{\text{false}} = \text{false}$.

- ▶ P respects pointwise semantic equality:

$\forall e_1, e_2 \in CExp.$

if $\forall e \in CExp. \llbracket e_1 e \rrbracket = \llbracket e_2 e \rrbracket$ then

$P e_1 = P e_2$

Then P is χ -undecidable.

Rice's theorem

The halting problem reduces to P :

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

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 $\lambda n. \text{Suc}(n)$?

Respond at <https://pingo.coactum.de/921051>.

Is $e \in CExp$ an implementation of the successor function for natural numbers?

This problem is not χ -decidable.

Use Rice's theorem:

- ▶ $P e = \mathbf{if} \forall n \in \mathbb{N}. \llbracket e \ulcorner n \urcorner \rrbracket = \ulcorner \text{suc } n \urcorner \rrbracket$
 then true else false.
- ▶ P is non-trivial:
 - ▶ $P (\lambda n. \text{Suc}(n)) = \text{true.}$
 - ▶ $P (\text{Zero}()) = \text{false.}$
- ▶ P respects pointwise semantic equality:

$$\begin{aligned} (\forall e \in CExp. \llbracket e_1 e \rrbracket = \llbracket e_2 e \rrbracket) &\Rightarrow \\ (\forall n \in \mathbb{N}. \llbracket e_1 \ulcorner n \urcorner \rrbracket = \llbracket e_2 \ulcorner n \urcorner \rrbracket) &\Rightarrow \\ P e_1 = P e_2 & \end{aligned}$$

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 $\Sigma \cup \{\sqcup\} \subseteq \Gamma$.
- ▶ $\delta \in S \times \Gamma \rightarrow S \times \Gamma \times \{L, R\}$:
The transition “function”.

Abstract syntax

$$\begin{array}{l} S \text{ is a finite set} \quad s_0 \in S \\ \Sigma \text{ is a finite set} \quad \sqcup \notin \Sigma \\ \Gamma \text{ is a finite set} \quad \Sigma \cup \{\sqcup\} \subseteq \Gamma \\ \delta \in S \times \Gamma \rightarrow S \times \Gamma \times \{\mathbf{L}, \mathbf{R}\} \\ \hline (S, s_0, \Sigma, \Gamma, \delta) \in TM \end{array}$$

Operational semantics

Positioned tapes

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

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

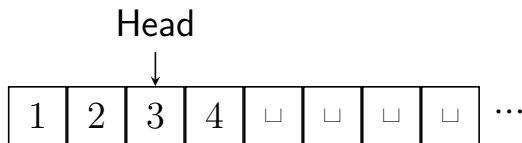
- ▶ Here (ls, rs) stands for

$$reverse\ ls \uplus rs$$

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

Positioned tapes

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



The symbol under the head

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

$$\begin{aligned} \text{head}_T &\in \text{Tape} \rightarrow \Gamma \\ \text{head}_T (ls, rs) &= \text{head } rs \end{aligned}$$

$$\begin{aligned} \text{head} &\in \text{List } \Gamma \rightarrow \Gamma \\ \text{head } [] &= \sqcup \\ \text{head } (x :: xs) &= x \end{aligned}$$

Writing

Writing to the tape:

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

The “tail” of a sequence:

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

Moving

Moving the head:

$$\text{move} \in \{\text{L}, \text{R}\} \rightarrow \text{Tape} \rightarrow \text{Tape}$$
$$\text{move R } (ls, rs) = (\text{head } rs :: ls, \text{tail } rs)$$
$$\text{move L } ([], rs) = ([], rs)$$
$$\text{move L } (ls, rs) = (\text{tail } ls, \text{head } ls :: rs)$$

Actions

Actions describe what the head will do:

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

Note:

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

First write, then move:

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

Quiz

Which of the following equalities are valid?

1. $act(0, L)(act(1, L)([], [])) = ([], [0, 1])$
2. $act(0, L)(act(1, L)([], [])) = ([0, 1], [])$
3. $act(0, L)(act(1, L)([], [])) = ([1, 0], [])$
4. $act(0, R)(act(1, R)([], [])) = ([], [0, 1])$
5. $act(0, R)(act(1, R)([], [])) = ([0, 1], [])$
6. $act(0, R)(act(1, R)([], [])) = ([1, 0], [])$

Respond at <https://pingo.coactum.de/921051>.

Quiz

$act(0, L) (act(1, L) ([], [])) =$
 $act(0, L) (move\ L (write\ 1 ([], []))) =$
 $act(0, L) (move\ L ([], [1])) =$
 $act(0, L) ([], [1]) =$
 $move\ L (write\ 0 ([], [1])) =$
 $move\ L ([], [0]) =$
 $([], [0])$

Quiz

$act(0, R) (act(1, R) ([], [])) =$
 $act(0, R) (move\ R (write\ 1 ([], []))) =$
 $act(0, R) (move\ R ([], [1])) =$
 $act(0, R) ([1], []) =$
 $move\ R (write\ 0 ([1], [])) =$
 $move\ R ([1], [0]) =$
 $([0, 1], [])$

Small-step operational semantics

A configuration consists of a state and a tape:

$$\textit{Configuration} = \textit{State} \times \textit{Tape}$$

The small-step operational semantics relates configurations:

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

Reflexive transitive closure

Zero or more small steps:

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

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's contents is initially a string of characters from the input alphabet Σ .
- ▶ 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 $List \Gamma$:

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

Constructing the result

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

$$\text{list} \in \text{Tape} \rightarrow \text{List } \Gamma$$

$$\text{list} (ls, rs) = \text{reverse } ls \uplus rs$$

$$\text{remove} \in \text{List } \Gamma \rightarrow \text{List } \Gamma$$

$$\text{remove} [] = []$$

$$\text{remove} (x :: xs) = \text{cons}' x (\text{remove } xs)$$

$$\text{cons}' \in \Gamma \rightarrow \text{List } \Gamma \rightarrow \text{List } \Gamma$$

$$\text{cons}' \sqcup [] = []$$

$$\text{cons}' x xs = x :: xs$$

Quiz

Which properties does \Downarrow satisfy?

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

$$\forall tm \in TM.$$

$$\forall xs \in List \Sigma_{tm}. \forall ys, zs \in List \Gamma_{tm}.$$

$$xs \Downarrow_{tm} ys \wedge xs \Downarrow_{tm} zs \Rightarrow ys = zs$$

2. Is it total (for every Turing machine)?

$$\forall tm \in TM. \forall xs \in List \Sigma_{tm}.$$

$$\exists ys \in List \Gamma_{tm}. xs \Downarrow_{tm} ys$$

Respond at <https://pingo.coactum.de/921051>.

The machine's partial function

The semantics as a partial function:

$$\begin{aligned} \llbracket - \rrbracket &\in \forall tm \in TM. List \Sigma_{tm} \rightarrow List \Gamma_{tm} \\ \llbracket tm \rrbracket xs = ys &\text{ if } xs \Downarrow_{tm} ys \end{aligned}$$

Two examples

An example

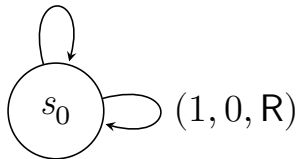
- ▶ Input alphabet: $\{0, 1\}$.
- ▶ Tape alphabet: $\{0, 1, \sqcup\}$.
- ▶ States: $\{s_0\}$.
- ▶ Initial state: s_0 .

Transition function

$$\delta(s_0, 0) = (s_0, 1, R)$$

$$\delta(s_0, 1) = (s_0, 0, R)$$

(0, 1, R)

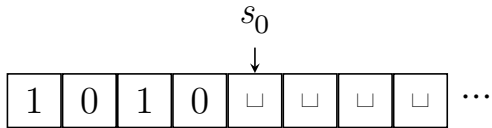
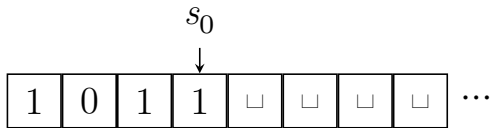
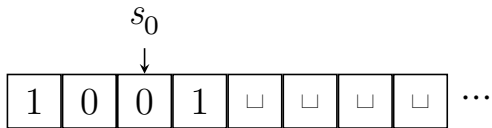
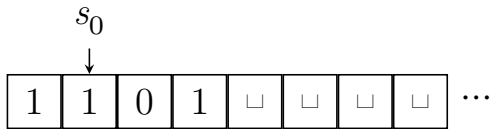
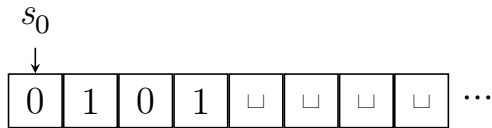


Quiz

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□
7. 1010□

Respond at <https://pingo.coactum.de/921051>.

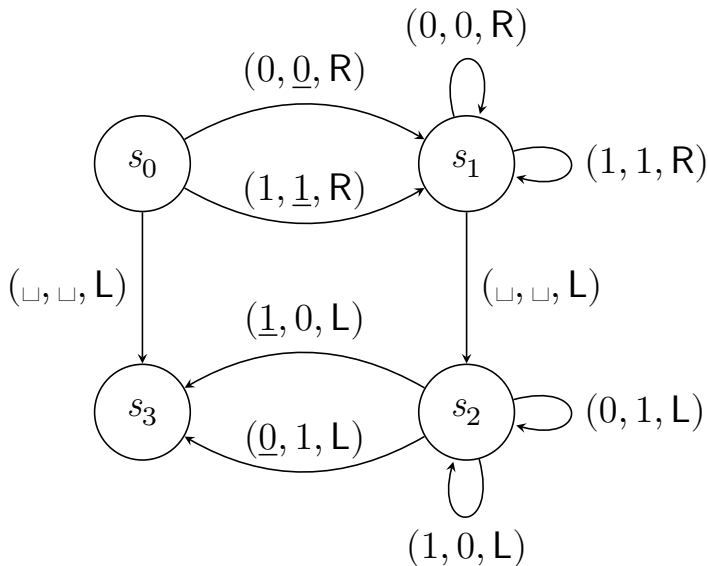


Another example

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*:

$$\begin{array}{l} S \text{ is a finite set} \quad s_0 \in S \quad A \subseteq S \\ \Sigma \text{ is a finite set} \quad \sqcup \notin \Sigma \\ \Gamma \text{ is a finite set} \quad \Sigma \cup \{\sqcup\} \subseteq \Gamma \\ \delta \in S \times \Gamma \rightarrow S \times \Gamma \times \{L, R\} \\ \hline (S, s_0, A, \Sigma, \Gamma, \delta) \in TM \end{array}$$

Is the string accepted?

A relation on $List\ \Sigma$:

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

Is the string rejected?

A relation on $List\ \Sigma$:

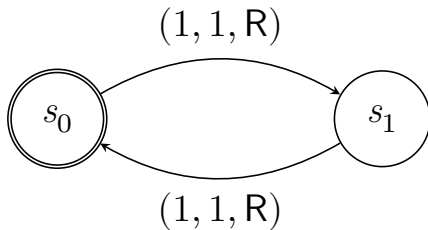
$$\frac{(s_0, [], xs) \longrightarrow^* (s, t) \quad \nexists c. (s, t) \longrightarrow c \quad s \notin A}{Reject\ xs}$$

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

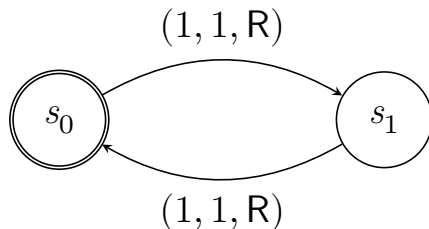
An example

- ▶ 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? (Respond at <https://pingo.coactum.de/921051>.)

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:

$$\lceil _ \rceil \in \mathbb{N} \rightarrow \text{List } \{1\}$$

$$\lceil \text{zero} \rceil = []$$

$$\lceil \text{suc } n \rceil = 1 :: \lceil n \rceil$$

Natural numbers

Another method:

$$\ulcorner _ \urcorner \in \mathbb{N} \rightarrow \text{List } \{0, 1\}$$

$$\ulcorner \text{zero} \urcorner = 0 :: []$$

$$\ulcorner \text{suc } n \urcorner = 1 :: \ulcorner n \urcorner$$

This method is used below.

Lists

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

- ▶ It is injective.
- ▶ There is a function

$$split \in List \Xi \rightarrow List \Xi \times List \Xi$$

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

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

Lists

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

- ▶ It is injective.
- ▶ There is a function

$$split \in List \Xi \rightarrow List \Xi \times List \Xi$$

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

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

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

Lists

Representation of *List A*:

$$\ulcorner _ \urcorner \in \text{List } A \rightarrow \text{List } (\Xi \cup \{0, 1\})$$

$$\ulcorner [] \urcorner = 0 :: []$$

$$\ulcorner x :: xs \urcorner = 1 :: \ulcorner x \urcorner \uparrow\uparrow \ulcorner xs \urcorner$$

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]

Respond at <https://pingo.coactum.de/921051>.

Pairs

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

Representation of $A \times B$:

$$\begin{aligned}\ulcorner _ \urcorner &\in A \times B \rightarrow \text{List } \Xi \\ \ulcorner (x, y) \urcorner &= \ulcorner x \urcorner^A \uparrow\uparrow \ulcorner y \urcorner^B\end{aligned}$$

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 $f \in A \rightarrow B$ is *Turing-computable* (with respect to these methods) if there is a Turing machine tm such that:

- ▶ $\Sigma_{tm} = \Sigma$.
- ▶ $\forall a \in A. \llbracket tm \rrbracket \ulcorner a \urcorner = \ulcorner f a \urcorner$.

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.