Lecture Computability (DIT312, DAT415)

Nils Anders Danielsson

2019-12-09

Today

- Representing Turing machines.
- A self-interpreter (a universal Turing machine).
- The halting problem.
- A Turing machine that is a χ interpreter.
- The Post correspondence problem.
- Some history.

Representing Turing machines

States

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

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

Alphabets

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

$$\ulcorner \Sigma \urcorner = \ulcorner m \urcorner$$
$$\ulcorner \Gamma \urcorner = \ulcorner n \urcorner$$
$$\ulcorner \sqcup \urcorner = \ulcorner 0 \urcorner$$
$$\ulcorner c_k \urcorner = \ulcorner k \urcorner$$

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

$$\ulcorner s \urcorner +\!\!+ \ulcorner x \urcorner +\!\!+ \ulcorner s' \urcorner +\!\!+ \ulcorner x' \urcorner +\!\!+ \ulcorner d \urcorner.$$

- 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 $(S, s_{initial}, \Sigma, \Gamma, \delta) \in TM$ is represented by

$$\ulcorner S \urcorner + \ulcorner s_{initial} \urcorner + \ulcorner \Sigma \urcorner + \ulcorner \Gamma \urcorner + \ulcorner \delta \urcorner.$$

- 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 symbols, 0 and 1.

Quiz

What Turing machine does $0010100100111010101100011101010001$ represent?

1. None
2. $S = \{ s_0 \}$, $\Sigma = \{ c_1 \}$, $\Gamma = \{ c_1, c_2, \sqcup \}$, $\delta(s_0, c_1) = (s_0, c_1, \mathsf{L})$
3. $S = \{ s_0 \}$, $\Sigma = \{ c_1, c_2 \}$, $\Gamma = \{ c_1, c_2, \sqcup \}$, $\delta(s_0, c_1) = (s_0, c_2, \mathsf{R})$

Self-interpreter

Self-interpreter

A self-interpreter or *universal Turing machine* $eval$ can simulate arbitrary Turing machines with arbitrary input:

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

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

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.
- 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,
  write the result to the first tape and halt.

The halting problem

The halting problem

$$\begin{array}{l} halts \in \{\, (tm, xs) \mid tm \in TM, xs \in List\, \Sigma_{tm} \} \to Bool \\ halts\ (tm, xs) = \\ \quad \textbf{if}\ [\![tm]\!]\ xs\ \textsf{is defined}\ \textbf{then} \\ \quad\quad \textsf{true} \\ \quad \textbf{else} \\ \quad\quad \textsf{false} \end{array}$$

This function is not Turing-computable.

The halting problem

The halting problem can also be viewed as a language:

$$\{ \ulcorner (tm, xs) \urcorner \mid tm \in TM, xs \in List\,\Sigma_{tm}, \\ [\![tm]\!]\ xs \text{ is defined} \}$$

This language is Turing-undecidable.

(Note the difference between this definition and the previous one.)

The halting problem (with self-application)

$\{\ulcorner tm \urcorner \mid tm \in TM, [\![tm]\!] \ulcorner tm \urcorner \text{ is defined}\}$

This language is Turing-undecidable. Proof sketch:

- Assume that the TM *halts* decides it.
- Define a TM *terminv* in the following way:
  - Simulate *halts* with *terminv*'s input.
  - If *halts* accepts, loop forever.
  - If *halts* rejects, halt.
- Note that *terminv* halts when the input string is $\ulcorner terminv \urcorner$ iff it does not halt for this input string.

The halting problem is undecidable

$$\{\ulcorner (tm, xs) \urcorner \mid tm \in TM, xs \in List\, \Sigma_{tm}, \\ [\![tm]\!]\; xs \text{ is defined}\}$$

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?

1. tm
2. $\ulcorner tm \urcorner$
3. $\ulcorner\ulcorner tm \urcorner\urcorner$
4. $tm +\!\!+ \ulcorner tm \urcorner$
5. $\ulcorner tm \urcorner +\!\!+ \ulcorner\ulcorner tm \urcorner\urcorner$
6. $tm +\!\!+ \ulcorner tm \urcorner +\!\!+ \ulcorner\ulcorner tm \urcorner\urcorner$

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 $e \in \mathit{CExp}$.
- 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: $\ulcorner \ulcorner n \urcorner^{\chi} \urcorner^{\mathsf{TM}} \mapsto \ulcorner n \urcorner^{\mathsf{TM}}$.

The Post correspondence problem

The Post correspondence problem

Definition (for a set Σ with at least two members):

- Given: $x_1, ..., x_n \in List\,\Sigma \times List\,\Sigma$.
- Goal: Find $k \geq 1$ and $i_1, ..., i_k \in \{1, ..., n\}$ such that

$$fst\; x_{i_1} + \!\!\!+ \cdots + \!\!\!+ fst\; x_{i_k} = \\ snd\; x_{i_1} + \!\!\!+ \cdots + \!\!\!+ snd\; x_{i_k}.$$

Examples on Wikipedia.

Quiz

Is the Post correspondence problem solvable for the given pairs of strings?

- A: $(001, 00)$, $(01, 10)$.
- B: $(01, 001)$, $(010, 01)$.

The Post correspondence problem

- Undecidable.
- Note that there is no reference to Turing machines (or χ expressions) in the statement of the problem.
- Proof idea:
  - Construct pairs such that a TM halts iff the problem is solvable.
  - The resulting string (if any) encodes the TM's computation history.
- Sipser's *Introduction to the Theory of Computation* (available online via Chalmers' library) contains a readable proof.

Ambiguity

- Undecidable:
  Is a context-free grammar ambiguous?
- The Post correspondence problem can be reduced to this one.

Ambiguity

Proof sketch (taken from Sipser):

- Given: Pairs $(t_1, b_1), \ldots, (t_n, b_n)$.
- Define a CFG with three non-terminals, and Start as the starting non-terminal:

$$
\begin{array}{lcllll}
\mathit{Start} & ::= & \mathit{Top} \mid \mathit{Bottom} & & & \\
\mathit{Top} & ::= & t_1\ \mathit{Top} & \mathtt{1} \mid \ldots \mid & t_n\ \mathit{Top} & \mathtt{n} \\
& \mid & t_1 & \mathtt{1} \mid \ldots \mid & t_n & \mathtt{n} \\
\mathit{Bottom} & ::= & b_1\ \mathit{Bottom} & \mathtt{1} \mid \ldots \mid & b_n\ \mathit{Bottom} & \mathtt{n} \\
& \mid & b_1 & \mathtt{1} \mid \ldots \mid & b_n & \mathtt{n}
\end{array}
$$

  (Here $\mathtt{1}, \ldots, \mathtt{n}$ are fresh terminals.)
- This grammar is ambiguous iff the given instance of the Post correspondence problem has a solution.

Brief and incomplete historical overview

Maybe not entirely correct, I'm not an expert on the history of the subject.

- 1800s, 1900s: Mathematics is made more formal.
- 1900: Hilbert's problems, including the Entscheidungsproblem (mentioned as part of problem ten).
- 1930: Gödel's completeness theorem. Semi-decision procedure.

Brief and incomplete historical overview

- 1931: Gödel's incompleteness theorems.
- 1936, Church: The Entscheidungsproblem is undecidable. The untyped λ-calculus.
- 1937, Turing: Turing machines, equivalence to the λ-calculus.
- 1946, Post: The Post correspondence problem.
- Mid-1900s: The Church-Turing thesis.
- 1970, Matiyasevitch (building on the work of others): Hilbert's tenth problem is undecidable.

Summary

- Representing Turing machines.
- A self-interpreter (a universal Turing machine).
- The halting problem.
- A Turing machine that is a χ interpreter.
- The Post correspondence problem.
- Some history.

Next week

- Summary of the course.
- Old exam questions.