TI2316 Slides week 8

Stefan Hugtenburg, Hans Tonino   Matthijs Spaan

Algorithmics group

Delft University of Technology

19 June 2019

This lecture

- Computable functions
- Mapping / many-to-one reducibility
- Rice's theorem
- Reduction via computation histories
- Linear bounded automata

Reducibility, formal

(See also videos Computable Functions and Reducing One Language to Another)

A function $f : \Sigma^* \to \Sigma^*$ is computable if there exists a TM M that halts on every input w with $f(w)$ on its tape. (Def. 5.17)

Computable functions can be used, for instance, to generate descriptions of TMs, given a certain input.

NB: We assume such a TM always halts in the accept state.

Mapping (many-to-one) reducibility (Def. 5.20)

A language A is mapping reducible to language B, written $A \leq_m B$, if there exists a computable function $f : \Sigma^* \to \Sigma^*$ such that for every $w \in \Sigma^*$:

$$w \in A \quad \Leftrightarrow \quad f(w) \in B.$$

The function f is called the reduction of A to B.

Note Consequentially, for a reduction f, the following holds:

$$\begin{array}{lcl} w \in A & \Rightarrow & f(w) \in B, \text{ and also} \\ w \notin A & \Rightarrow & f(w) \notin B. \end{array}$$

Properties of reduction (1)

If $A \leq_m B$ and B is decidable, then A is decidable. (Th. 5.22)

If $A \leq_m B$ and A is undecidable, then B is undecidable. (Cor. 5.23)

$A \leq_m B$ if and only if $\overline{A} \leq_m \overline{B}$.

Example 1 mapping reduction

Instead of direct reduction $A_{\mathsf{TM}} \leq HALT_{\mathsf{TM}}$, a mapping reduction $A_{\mathsf{TM}} \leq_m HALT_{\mathsf{TM}}$ can be given, where the reduction f is computed by the TM F:

F = "On input $\langle M, w \rangle$:

1. Construct the following TM M':

M' = "On input x x:

1. Run M on x.

2. Accept if M accepts.

3. "Loop" if M rejects."

2. Return $\langle M', w \rangle$."

Example 2 mapping reduction

There exists a (simple) direct reduction $E_{\mathsf{TM}} \leq EQ_{\mathsf{TM}}$.
This reduction can be interpreted as a mapping reduction.

The reduction f is defined here as:

$$f(\langle M \rangle) = \langle M, M_{\emptyset} \rangle,$$

where $M_{\emptyset}$ is a machine that rejects every input.

(See also video The Equivalence of Turing Machines for the direct reduction)

Properties of reduction (2)

If $A \leq_m B$ and B is Turing-recognizable, then A is Turing-recognizable. (Th. 5.28)

If $A \leq_m B$ and A is not Turing-recognizable, then B is not Turing-recognizable. (Cor. 5.29)

Exercise

We have shown that A_{TM} is directly reducible to E_{TM}, i.e., $A_{\mathsf{TM}} \leq E_{\mathsf{TM}}$, which was to say a procedure to decide E_{TM} would give us a procedure to decide A_{TM}.

Show that $A_{\mathsf{TM}} \leq_m E_{\mathsf{TM}}$ does not hold. Or, equivalently, $\overline{A_{\mathsf{TM}}} \leq_m \overline{E_{\mathsf{TM}}}$ does not hold.

Hint: What do you know about $\overline{E_{\mathsf{TM}}}$?

Example 3 mapping reduction

In the direct reduction $A_{\mathsf{TM}} \leq E_{\mathsf{TM}}$, technically there is a mapping reduction:

$$A_{\mathsf{TM}} \leq_m \overline{E_{\mathsf{TM}}},$$

using the reduction f defined as:

$$f(\langle M, w\rangle) = \langle M_1 \rangle,$$

where M_1 is a machine with the property that

$$M \text{ accepts } w \quad \Leftrightarrow \quad L(M_1) \neq \emptyset.$$

Constructing M_1

Suppose we have $\langle M, w \rangle$. To be able to use R we modify M into M_1:

M_1 = "On input x:

1. If $x \neq w$, reject.
2. If $x = w$, run M on input w and accept if M accepts input w."

We now have:

$$M \text{ accepts } w \quad \Leftrightarrow \quad L(M_1) = \{w\} \neq \emptyset.$$

Example 4 mapping reduction

EQ_{TM} is not Turing-recognizable and not co-Turing-recognizable. (Th. 5.30)

Proof

- To prove that EQ_{TM} is not Turing-recognizable, we show: $\overline{A_{\text{TM}}} \leq_m EQ_{\text{TM}}$.
- For the second part, EQ_{TM} is not co-Turing-recognizable, we show: $\overline{A_{\text{TM}}} \leq_m \overline{EQ_{\text{TM}}}$, or, equivalently, $A_{\text{TM}} \leq_m EQ_{\text{TM}}$.

Exercise

Let A be a language. Prove the following statement.

A is Turing-recognizable iff $A \leq_m A_{\mathsf{TM}}$.

Solution

($\Leftarrow$) Suppose $A \leq_m A_{\mathsf{TM}}$. Since A_{TM} is Turing-recognizable, A must be too (Th. 5.28).

($\Rightarrow$) Suppose A is Turing-recognizable. Then there exists a TM M_A recognizing the language A, i.e., $L(M_A) = A$. The function f projecting a word $w \in A$ onto $\langle M_A, w \rangle$ is the reduction we need, since (check this!):

$$w \in A \text{ iff } f(w) = \langle M_A, w \rangle \in A_{\mathsf{TM}}.$$

Of course the requirement for f to be a computable function has been met.

Exercise

Check whether the following statement is true or false:

> *If A and B are both non-Turing-recognizable languages, we have $A \leq_m B$.*

Answer

FALSE!!!

Counterexample: Take $A = \overline{EQ_{\mathsf{TM}}}$ and $B = \overline{A_{\mathsf{TM}}}$.

Both $\overline{EQ_{\mathsf{TM}}}$ and $\overline{A_{\mathsf{TM}}}$ are non-Turing-recognizable.

$\overline{EQ_{\mathsf{TM}}} \leq_m \overline{A_{\mathsf{TM}}}$ cannot hold, however, since this is equivalent to $EQ_{\mathsf{TM}} \leq_m A_{\mathsf{TM}}$. Since A_{TM} is Turing-recognizable, EQ_{TM} would have to be Turing-recognizable as well, but this is not the case.

Rice's theorem

Let P be a set of Turing machines satisfying:

1. P is exclusively defined in terms of input/output behavior of TMs; that is to say, if $L(M_1) = L(M_2)$, then $\langle M_1 \rangle \in P$ iff $\langle M_2 \rangle \in P$;
2. P is not trivial, which is to say, $P \neq \emptyset$ and $\overline{P} \neq \emptyset$.

Then P is not decidable.

Proof (1)

Let $M_\emptyset$ be a TM that recognizes the empty language by rejecting every input. This means that $L(M_\emptyset) = \emptyset$.
If $\langle M_\emptyset \rangle \notin P$, then we perform the reduction $HALT_{\mathsf{TM}} \leq_m P$, otherwise $HALT_{\mathsf{TM}} \leq_m \overline{P}$. In both cases, the reduction is done the same way.
Suppose $\langle M_\emptyset \rangle \notin P$. Also let $\langle M_P \rangle \in P$ (we know that $P \neq \emptyset$).

Proof (2)

The reduction f is defined as follows:

$$f(\langle M, w \rangle) = \langle M_1 \rangle,$$

with:

M_1 = "On input x:

1. Run M on input w.
2. Run M_P on input x.
3. Return the output of M_P on x."

Proof (3)

Now we have:

- If $\langle M, w \rangle \in HALT_{\mathsf{TM}}$, then $L(M_1) = L(M_P)$ (since M_1 always reaches steps 2 and 3). But then $\langle M_1 \rangle \in P$, since $\langle M_P \rangle \in P$.
- If $\langle M, w \rangle \notin HALT_{\mathsf{TM}}$, then M_1 will not reach a halting state while executing step 1, so that $L(M_1) = \emptyset = L(M_\emptyset)$. Therefore, $\langle M_1 \rangle \notin P$, since $\langle M_\emptyset \rangle \notin P$.

Since f is a computable function, we have now presented a reduction. The other case is analogous.

Reduction via computation histories

This technique uses computation histories in the construction of a reduction.

(Sipser p. 220 and following)

Computation histories (Def. 5.5)

Let M be a TM and w an input word. An accepting computation history for M on w is a sequence of configurations $C_1, C_2, \ldots, C_l$ such that C_1 is the start configuration of M on w, C_l is an accepting configuration of M, and each C_i is yielded by C_{i-1}.

A rejecting computation history is defined analogously, except that C_l is a rejecting configuration.

Linear bounded automata (Def. 5.6)

A Linear Bounded Automaton (LBA) is a TM where the read/write head cannot leave the part of the tape that contain(ed) the input.

(See also video Linear Bounded Automata)

Exercise

Explain why for an LBA, the following holds (which explains the name):

Since the tape alphabet can be greater than the input alphabet, the available memory of an LBA is in fact a constant factor times the input length.

Number of configurations of an LBA (Lem. 5.8)

Let M be an LBA with q states and g symbols in the tape alphabet. Then there are exactly qng^n different configurations of M for a tape of length n.

The acceptance problem for LBAs (Th. 5.9)

The acceptance problem A_{LBA} for LBAs, defined as

$$A_{\mathsf{LBA}} = \{\langle M, w\rangle \mid M \text{ is an LBA that accepts input } w\},$$

is decidable.

Undecidability of E_{LBA} (Th. 5.10)

The problem E_{LBA}, defined as

$$E_{\mathsf{LBA}} = \{\langle M \rangle \mid M \text{ is an LBA with } L(M) = \emptyset\}$$

is not decidable.

Proof: reduction via computation history:

$$A_{\mathsf{TM}} \leq E_{\mathsf{LBA}}.$$

Suppose R is a decider that decides E_{LBA}.

Auxiliary construction

We construct an LBA B that recognizes the language of accepting computation histories of M on w for a specific Turing machine M and input w. B is defined such that:

$$L(B) \neq \emptyset \quad \Leftrightarrow \quad M \text{ accepts } w.$$

Note that $L(B)$ contains exactly one string if M accepts input w.

This B can be "fed" to the decider R.