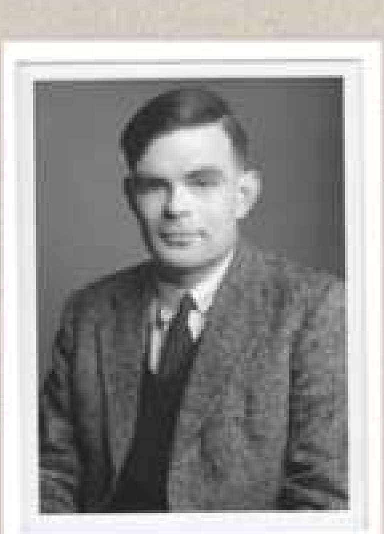
XV

Automata & Formal Languages

Fifteenth Lecture – 8 November 2022



REGISTER MACHINES

Instructions

Let Σ be an alphabet and Q a non-empty finite set whose elements we shall call *states*. A tuple of the form

$$(0, k, a, q) \in \mathbb{N} \times \mathbb{N} \times \Sigma \times Q,$$
$$(1, k, a, q, q') \in \mathbb{N} \times \mathbb{N} \times \Sigma \times Q \times Q,$$
$$(2, k, q, q') \in \mathbb{N} \times \mathbb{N} \times \times Q \times Q \text{ or}$$
$$(3, k, q, q') \in \mathbb{N} \times \mathbb{N} \times Q \times Q$$

is called a (Σ, Q)-*instruction*. For improved readability, we write

$$+(k, a, q) := (0, k, a, q), \qquad (\text{"add"})$$
$$?(k, a, q, q') := (1, k, a, q, q'), \qquad (\text{"check"})$$
$$?(k, \varepsilon, q, q') := (2, k, q, q') \text{ and} \qquad (\text{"check"})$$
$$-(k, q, q') := (3, k, q, q') \qquad (\text{"remove"})$$

$\mathrm{Instr}(\Sigma, Q)$

Σ-Register machine (Σ, Q, P) — PROGRAM consisting of PROGRAM LINES

$P : Q \longrightarrow \mathrm{Instr}(\Sigma, Q)$

Configuration $Q \times W^{n+1}$ of length $n+1$.

UPPER REGISTER INDEX

RM transforms configurations

We say that a sequence $C := (q, w_0, \ldots, w_n) \in Q \times \mathbb{W}^{n+1}$ is a *configuration* or *snapshot* of length $n+1$. In such a configuration, the first entry q is called the *state* of the configuration and the rest is called the *register content* of the configuration. If M is a register machine with upper register index n and C is any configuration of length $m \geq n+1$, then we can define the action of M on C: we say that M *transforms C to C'* if the following is true:

Case 1. If $P(q) = +(k, a, q')$ and $C' = (q', w_0, \ldots, w_{k-1}, w_k a, w_{k+1}, \ldots, w_m)$.

Case 2. If $P(q) = ?(k, a, q', q'')$,

Subcase 2a. $w_k = wa$ for some w and $C' = (q', w_0, \ldots, w_m)$ or

Subcase 2b. $w_k \neq wa$ for any w and $C' = (q'', w_0, \ldots, w_m)$.

Case 3. If $P(q) = ?(k, \varepsilon, q', q'')$,

Subcase 3a. $w_k = \varepsilon$ and $C' = (q', w_0, \ldots, w_m)$ or

Subcase 3b. $w_k \neq \varepsilon$ and $C' = (q'', w_0, \ldots, w_m)$.

Case 4. If $P(q) = -(k, q', q'')$,

Subcase 4a. $w_k = \varepsilon$ and $C' = (q', w_0, \ldots, w_m)$ or

Subcase 4b. $w_k = wa$ for some a and $C' = (q'', w_0, \ldots, w_{k-1}, w, w_{k+1}, \ldots, w_m)$.

COMPUTATION SEQUENCE of M w/ INPUT $\vec{w}$

Computation halts at time k with register content v

M & M' are STRONGLY EQUIVALENT
if for all $\vec{w}$
M halts at time k $\Longleftrightarrow$ M' halts at time k w/ input $\vec{w}$ w/ input $\vec{w}$
& for all $\vec{w}$ & i, the register contents of the computation sequences at time i are the same.

PADDING LEMMA
For each RM there are ∞ many distinct strongly eq. RM.

Observation: If $|Q| = |Q'|$, then for each (Σ, Q, P) there is P' s.t. (Σ, Q, P) & (Σ, Q', P') are strongly equivalent.

Prop. 4.3 Up to strong equivalence, there are only countably many RM.

Proof. By Obs, only $|Q|$ matters up to strong equivalence.
Let $M_{k,n}$ be the collection of RM with a fixed state set Q with $|Q| = n$ and upper register index $\leq k$.

Let us count instructions:

$\mathrm{Instr}(\Sigma, Q)$ consists of four types of instructions

$+(\ell, a, q)$	at most $(k+1)\lvert\Sigma\rvert \cdot n$.
$?(\ell, a, q, q')$	at most $(k+1)\lvert\Sigma\rvert \cdot n^2$
$?(\ell, \varepsilon, q, q')$	at most $(k+1) \cdot n^2$
$-(\ell, q, q')$	at most $(k+1)\, n^2$

In particular, $\lvert \mathrm{Instr}(\Sigma, Q) \rvert =: N_{n,k}$ is finite.

So there are $N_{n,k}^{\,n}$ many programs.
Thus: finitely many!

But then the collection of all RM (up to strong eq.) is

$\bigcup_{k,n \geq 0} M_{k,n}$, i.e., a countable union of finite sets, thus countable.

q.e.d.

§4.2 Performing operations & answering questions

By an OPERATION we mean a partial function from $\mathbb{W}^{n+1}$ to $\mathbb{W}^{n+1}$.

My notation for partial functions is

$$f : \mathbb{W}^{n+1} \dashrightarrow \mathbb{W}^{n+1}.$$

We write $f(\vec{w})\downarrow$ for $\vec{w} \in \mathrm{dom}(f)$
is defined
converges

$f(\vec{w})\uparrow$ for $\vec{w} \notin \mathrm{dom}(f)$
is undefined
diverges

A RM M **performs f** if for all $\vec{w}$

$f(\vec{w})\downarrow \iff$ M halts on input $\vec{w}$ and the reg. content at time of halting is $f(\vec{w})$

$f(\vec{w})\uparrow \iff$ M doesn't halt on input $\vec{w}$.

<u>Example</u> The operation "never halt" is performed by a RM.

The function is $f : \mathbb{N}^{n+1} \dashrightarrow \mathbb{N}^{n+1}$
$\operatorname{dom}(f) = \emptyset$

Any program w/o halt state in any RHS of a program line will do that, e.g.

$$q_S \longmapsto +(0, a, q_S).$$
$$q_H \longmapsto +(0, a, q_S)$$

<u>Remark</u> There are many RM that do that; including many that are not strongly eq.

<u>Example</u> The operation "halt w/o doing anything".

The function is $id : \mathbb{N}^{n+1} \dashrightarrow \mathbb{N}^{n+1}$.

$$q_S \longmapsto ?(0, a, q_H, q_H)$$

This halts after one step and keeps the register content intact.

ANSWERING QUESTIONS

A question with $k+1$ answers is a partition of $\mathbb{W}^{n+1}$ into $k+1$ sets A_i; $\mathbb{W}^{n+1} = \bigcup_{i \leq k} A_i$.

A register machine **answers a question** if upon input $\vec{w}$ it does a finite computation at the end of which the configuration is

$$(\overline{q_i}, \vec{w}) \iff \vec{w} \in A_i$$

[the machine has $k+1$ many designated answer states $\overline{q_i}$.]

Examples "Is register i empty?"

$$q_S \mapsto ?(i, \varepsilon, \overline{q_{Yes}}, \overline{q_{No}})$$

"Is the final letter in register i an a?"

$$q_S \longmapsto ?(i, a, \overline{q_{Yes}}, \overline{q_{No}}).$$

Lemma 4.6 (Concatenation Lemma; Subroutine Lemma)

If M performs $F: \mathbb{W}^{n+1} \dashrightarrow \mathbb{W}^{n+1}$

M' performs $F': \mathbb{W}^{n+1} \dashrightarrow \mathbb{W}^{n+1}$:

then we can construct a RM $\hat{M}$ s.t.

$\hat{M}$ performs $F' \circ F$

[Convention: if $F(\vec{w})\uparrow$ then $F' \circ F(\vec{w})\uparrow$.]

<u>Proof.</u> Assume w.l.o.g. $Q \cap Q' = \emptyset$.

$$\hat{Q} := Q \cup Q' \setminus \{q_H\}.$$

- Let P^* be P with $q_H \longmapsto P(q_H)$ removed & all instances of q_H replaced by q'_S.

$$\hat{P} := P^* \cup P'$$

Then $\hat{M} := (\Sigma, \hat{Q}, \hat{P})$ performs $F' \circ F$.

q.e.d.

Lemma 4.7 Case distinction Lemma

Suppose Q is a question w/ $k+1$ answers and $F_i : \mathbb{W}^{n+1} \dashrightarrow \mathbb{W}^{n+1}$ for $i \leq k$.

Let M be a RM that answers Q

M_i —"— performs F_i

Then the operation

$$G(\vec{w}) := F_i(\vec{w}) \text{ if } \vec{w} \in A_i$$

is performed by a register machine.

Proof. Assume w.l.o.g. that Q is disjoint from all Q_i

and that $\bigcap_{i \leq k} Q_i = \{q_H\}$.

Let P_i^* be P_i whose all occurrences of $q_{s,i}$ are replaced with $\overrightarrow{q_i}$

[answer state for A_i]

$$Q^* := Q \cup \bigcup_{i \leq k} Q_i \setminus \{q_{s,i}\}$$

$$P^* := P \cup \bigcup_{i \leq k} P_i^*.$$

Then $M^* = (\Sigma, Q^*, P^*)$ performs G.

q.e.d.

<u>Example</u> $f(\vec{w}) := \begin{cases} \vec{w} & w_i \neq \varepsilon \\ \uparrow & w_i = \varepsilon \end{cases}$

is performed by a RM. ⌟

[Step 1. Answer the Q "Is reg. i empty".

Step 2. If YES: perform "never halt"

If NO: perform "don't change anything & halt."]

This uses the case distinction lemma.

MORE EXAMPLES

(1) "Delete the final letter of i, if it exists"

$$q_S \longmapsto -(i, q_H, q_H)$$

(2) "Add a to the ith register"

$$q_S \longmapsto +(i, a, q_H)$$

Remark ② also performs the operation "guarantee that register i is non-empty".

③ "Empty register i"

$$q_S \longmapsto -(i, q_H, q_S)$$

Together ②, ③ can make sure that we control the emptiness status of a register.

MORE EXAMPLES IN LECTURE XVI.