XV

Fifteenth Lecture of
AUTOMATA & FORMAL LANGUAGES

8 November 2023

Concepts from Lecture XIV:

REGISTER MACHINE $M = (\Sigma, Q, P)$

CONFIGURATION $(q, \vec{w})$

M TRANSFORMS C INTO C'

COMPUTATION SEQUENCE $C(k, M, \vec{w})$

HALTING

§4.2 Performing operations & answering questions

Partial functions

Function $f : X \longrightarrow Y$ $\quad dom(f) = X$, $ran(f) \subseteq Y$

Partial function $f : X \dashrightarrow Y$ $\quad dom(f) \subseteq X$, $ran(f) \subseteq Y$

OUR NOTATION FOR PARTIAL FUNCTIONS

We write $f(x)\downarrow$ for $x \in dom(f)$ CONVERGES

$f(x)\uparrow$ for $x \notin dom(f)$ DIVERGES

How can we think of RM M as a partial function?

Call a partial fn $F: \mathbb{N}^{n+1} \dashrightarrow \mathbb{N}^{n+1}$ an operation

Define

$$F_M : \mathbb{N}^{n+1} \dashrightarrow \mathbb{N}^{n+1}$$

$F_M(\vec{w})\uparrow$ iff M does not halt on input $\vec{w}$

$F_M(\vec{w})\downarrow = \vec{v}$ iff M halts on input $\vec{w}$ and $\vec{v}$ is the register content at time of halting

---

<u>Definition</u> M performs $F: \mathbb{N}^{n+1} \dashrightarrow \mathbb{N}^{n+1}$ if $F = F_M$.

<u>Example 1</u> Operation: NEVER HALT

$$F: \mathbb{N}^{n+1} \dashrightarrow \mathbb{N}^{n+1}$$

$$\mathrm{dom}(F) = \emptyset$$

This is performed by

$$q_S \longmapsto +(0, q, q_S)$$

Remark 1. By the Padding Lemma, there are ∞ many strongly equivalent machines performing F; but there are also lots of them producing very diff. computation sequences.

2. We have to commit to use a concrete one.

Example 2 ALWAYS HALT, DON'T CHANGE INPUT

$$F = id : \mathbb{W}^{n+1} \longrightarrow \mathbb{W}^{n+1}$$

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

ANSWERING QUESTIONS

A *question with* $k+1$ *answers* is a partition of $\mathbb{W}^{n+1}$ into $k+1$ disjoint sets $A_0, ..., A_k$. E.g., the question "does the second register end with a?" is the partition $A_0 := \{\vec{w};\ \exists v(w_2 = va)\}$ and $A_1 := \mathbb{W}^{n+1} \backslash A_0$. A register machine M *answers a question with* $k+1$ *answers* if it has $k+1$ designated *answer states* $\widehat{q}_0, ..., \widehat{q}_k$ and for each $\vec{w}$, the computation of M with input $\vec{w}$ produces in finitely many steps a configuration $(\widehat{q}_i, \vec{w})$ if and only if $\vec{w} \in A_i$.

<u>Question</u> partition ANSWER SETS

$$\mathbb{W}^{n+1} = \bigcup_{\ell \leq k} A_\ell$$

Answering a question with k+1 answers:

k+1 ANSWER STATES $\hat{q}_i$

M answers $(A_\ell;\ \ell \leq k)$ if on input $\vec{w}$ it produces in finitely many steps a configuration

$$(\hat{q}_i, \vec{w})$$

the same as input

$$\iff \vec{w} \in A_i$$

Important: answering the question should not destroy the input.

Example 3

Is register i empty?

Two answers YES : $A_0 := \{\vec{w}\,;\, w_i = \varepsilon\}$

NO : $A_1 := \{\vec{w}\,;\, w_i \neq \varepsilon\}$

$$q_s \longmapsto ?(i, \varepsilon, \hat{q}_0, \hat{q}_1).$$

Example 4

Does register i end in a?

As before:

YES $A_0 := \{\vec{w}\,;\, \exists w\,(w_i = wa)\}$

$A_1 := W^{n+1} \setminus A_0$

$$q_s \longmapsto ?(i, a, \hat{q}_0, \hat{q}_1).$$

THE SUBROUTINE LEMMA (L 4.6)

If M performs F and M' performs F', then there is a machine $\hat{M}$ performing $F' \circ F$.

Proof. W.l.o.g., assume $Q \cap Q' = \emptyset$.

Let P^* be P with all occurrences of q_H replaced by q_s', removing $P(q_H)$.

$\hat{Q} := Q \cup Q' \setminus \{q_H\}$; $\hat{P} := P^* \cup P'$; $\hat{M} = (\Sigma, \hat{Q}, \hat{P})$.

q.e.d.

IMPORTANT REMARK

The subroutine lemma does not just show existence, but gives an algorithm to produce $\hat{M}$ on the basis of M, M'.

The same applies to the CASE DISTINCTION LEMMA proved on the next page.

Lemma 4.7 (Case Distinction Lemma). Let $Q = \{A_i; i \leq k\}$ be a question with $k+1$ answers and $f_i : \mathbb{W}^{n+1} \dashrightarrow \mathbb{W}^{n+1}$ be operations for $i \leq k$. If Q is answered by a register machine $M = (\Sigma, Q, P)$ and f_i is performed by $M_i := (\Sigma, Q_i, P_i)$ (for $i \leq k$), then we can construct a register machine that performs the operation defined by $g(\vec{w}) := f_i(\vec{w})$ if and only of $\vec{w} \in A_i$.

M answers $(A_i ; i \leq k)$.

M_i performs $f_i : \mathbb{W}^{n+1} \dashrightarrow \mathbb{W}^{n+1}$

$g(\vec{w}) := f_i(\vec{w})$ iff $\vec{w} \in A_i$

Then g is performed by a machine.

<u>Proof</u>. W.l.o.g., $Q \cap Q_i = \emptyset$ for all i

$Q_i \cap Q_j = \{q_H\}$

Also assume $P_i(q_H) = P_j(q_H)$ $\forall i, j$.

Take P^* to be P where $\hat{q}_i$ is replaced by q_s^i [the start state of M_i]

$\hat{Q} := Q \cup \bigcup_{i \leq k} Q_i$ $\quad \hat{P} := P^* \cup \bigcup_{i \leq k} P_i$

$\hat{M} := (\Sigma, \hat{Q}, \hat{P})$ performs g.

q.e.d.

Example 5

$$F(\vec{w}) := \begin{cases} \vec{w} & w_i \neq \varepsilon \\ \uparrow & w_i = \varepsilon \end{cases}.$$

"undefined"

can be performed by RM as follows:

This looks like natural language, but the pink references make it precise.

Check WHETHER i IS EMPTY;
if so, HALT W/O CHANGES;
otherwise, NEVER HALT.

VERY IMPORTANT REMARK

This is not an informal hand-waiving description, but a concrete instruction that leads to a unique machine.

up to notation (= isomorphic)

Note that many choices are involved, but they are fixed in our set-up.

LOTS OF EXAMPLES

6. REMOVE FINAL LETTER IN i (IF EXISTS)

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

7. EMPTY REG. i

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

8. ADD a to REG i

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

9. ADD $w \in W$ to REG i

$w = a_0 \dots a_n$

Perform ADD a_0 to REG i (8.)
ADD a_1 to REG i (8.)
⋮
ADD a_n to REG i (8.)

and use the subroutine lemma.

10. What is the final letter of i? (if any)

Question with $k+2$ answers if

$$\Sigma = \{a_0, \ldots, a_k\}.$$

$$A_\ell := \{\vec{w} \,;\, \exists v\,(w_i = v a_\ell)\} \quad \ell \leq k$$

$$A_{k+1} := \{\vec{w} \,;\, w_i = \varepsilon\}$$

Cascade Ex. 4 as follows:

Does i end in a_0 $\xrightarrow{\text{YES}}$ $\hat{q}_0$

$\downarrow$ NO

Does i end in a_1 $\xrightarrow{\text{YES}}$ $\hat{q}_1$

$\downarrow$ NO

$\vdots$

$\downarrow$ NO

Does i end in a_k $\xrightarrow{\text{YES}}$ $\hat{q}_k$

$\downarrow$ NO

$\hat{q}_{k+1}$.

An iteration of k many applications of the case distinction lemma.