XVIII

Eighteenth Lecture of Automata & Formal Languages

26 November 2024

IN WHICH WE SHALL EXPLAIN HOW THE MODERN WORLD WORKS

§4.6 Coding languages and machines

$$\Sigma := \{ 0, 1, +_0, +_1, -, ?_0, ?_1, ?_\varepsilon, [,], / \}$$

eleven letters $\quad 11 < 2^4$

0	1	$+_0$	$+_1$	$-$	$?_0$	$?_1$	$?_\varepsilon$	$[$	$]$	$/$
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010

A RM looks like this

$q_s \mapsto -(2, q_2, q_H) \quad q_H \mapsto ?_\varepsilon(1, q_H, q_H) \quad q_2 \mapsto +_0(2, q_H)$

Represent states and registers by binary numbers

This allows me to get rid of the "$q \mapsto$" part.

$0, q_s \rightsquigarrow \varepsilon$

$1, q_H \rightsquigarrow 0$

$2, q_2 \rightsquigarrow 1$

$-(1,1,0), ?_\varepsilon(0,0,0), +_0(1,0)$

String in alphabet $\{0, 1, +_0, +_1, -, ?_\varepsilon, ?_0, ?_1, (,), ,\}$ $[\ ]\ /$

Avoid confusion by writing $(\mapsto [$, $) \mapsto]$, comma as $/$.

$-[1/1/0]/?_\varepsilon[0/0/0]/+_0[1/0]$

Instruction codes.

$$R(+_{\mathbb{0}}) := +_{\mathbb{0}}\left[(\mathbb{0}+\mathbb{1})^*/(\mathbb{0}+\mathbb{1})^*\right]$$

$$R := R(+_{\mathbb{0}}) + R(+_{\mathbb{1}}) + R(-) + R(?_{\mathbb{0}}) + R(?_{\mathbb{1}}) + R(?_{\varepsilon}).$$

$$R_{RM} := R(/R)^*$$

These are regular expressions for the codes of $+_{\mathbb{0}}$-instructions, instructions, RMs, respectively.

So, the set of codes of RMs is a regular language and can be computed by a RM.

<u>Configurations</u> $(q, \vec{w})$ encoded as finite sequence of binary numbers separated by /

$$R_C := (\mathbb{0}+\mathbb{1})^*\left(/(\mathbb{0}+\mathbb{1})^*\right)^*$$

This is a regular expression, so again the set of codes of configurations is regular, thus computable

Observations

(1) The sets of codes of RM & configurations are computable.

As per section on coding numbers, a RM can access the information coded inside the code.

Remark on pf of L 4.25:

The computation seq. is defined from the transformation fn (computable by L 4.24) via recursion. Since computable fns are closed under rec., this proves L 4.25. q.e.d

Observation (CONTINUED).

2 & 3

Lemma 4.24. The *transformation function*

$$f_{\mathrm{T}}\colon \mathbb{W}^2 \dashrightarrow \mathbb{W}\colon (\mathrm{code}(M), \mathrm{code}(C)) \mapsto \mathrm{code}(C') \text{ if } M \text{ transforms } C \text{ to } C'$$

is computable.

Proof. Let $C = (q, \vec{w})$. The code of M contains information about what $P(q)$ is; ap
concrete operation corresponding to the instruction $P(q)$ to $\vec{w}$ as given on p. 46.

Lemma 4.25. The *computation sequence function*

$$f_{\mathrm{CS}}\colon \mathbb{W}^3 \dashrightarrow \mathbb{W}\colon (\mathrm{code}(M), \mathrm{code}(q_{\mathrm{S}}, \vec{w}), v) \mapsto \mathrm{code}(C(\#v, M, \vec{w}))$$

is computable.

4.7 Software and universality

Theorem 4.26 (The Software Principle). There is a register machine U, called a *universal register machine*, such that

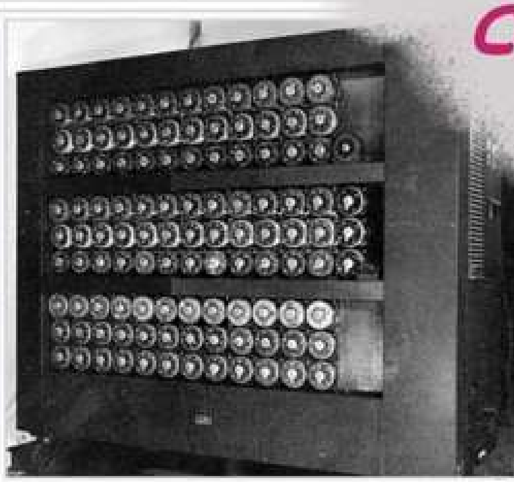
$$f_{U,2}(v,u) = \begin{cases} f_{M,k+1}(\vec{w}) & \text{if } v = \mathrm{code}(M) \text{ for a register machine } M \text{ with upper register} \\ & \quad \text{index } k \text{ and } u = \mathrm{code}(C) \text{ for a configuration with register} \\ & \quad \text{content } \vec{w} \text{ of length } k+1, \\ \uparrow & \text{otherwise.} \end{cases}$$

COMPUTING MACHINES

Alan Turing
OBE FRS

Turing c. 1928 at age 16

Born	Alan Mathison Turing 23 June 1912 Maida Vale, London, England
Died	7 June 1954 (aged 41) Wilmslow, Cheshire, England



A wartime picture of a Bletchley Park Bombe

Wilhelmus Schickart (painted 1632)

Replica of Leibniz's stepped reckoner in the Deutsches Museum.

> ... it is beneath the dignity of excellent men to waste their time in calculation when any peasant could do the work just as accurately with the aid of a machine.
>
> —Gottfried Leibniz[1]

Gottfried Wilhelm Leibniz

Portrait, 1695

Born	1 July 1646 Leipzig, Electorate of Saxony, Holy Roman Empire
Died	14 November 1716 (aged 70) Hanover, Electorate of Hanover, Holy Roman Empire

Charles Babbage
KH FRS

Babbage in 1860

Born	26 December 1791 London, England
Died	18 October 1871 (aged 79) Marylebone, London, England
Alma mater	Peterhouse, Cambridge

The London Science Museum's difference engine, the first one actually built from Babbage's design. The design has the same precision on all columns, but in calculating polynomials, the precision on the higher-order columns could be lower.

Proof of Thm 4.26

Input v, u. Need to compute $f_{M,k+1}(\vec{w})$

where $v = \text{code}(M)$

and u is a code for $\vec{w}$

Let q_S, q_H be the start and halt state of M.

Scratch registers n, m, ℓ. Empty them.

[This is a count-through argument, reg. ℓ counter.]

Repeat

- Let t be the content of reg. ℓ
- By L 4.25 calculate $C(\#t, M, \vec{w}) = (q, \vec{s})$. $\quad \vec{s} = (s_0, \dots, s_k)$
- Write a code of q into reg n.
- Write s_0 into reg m.

until register n contains the code of q_H.

If that happens, output content of reg m and halt.

q.e.d.

Remark The URI of a machine is one of the reasons why it is powerful.

U has a fixed URI, but can perform the task of all machines!

[Shift of the complexity from "hardware" (U) to "software" (v).]

NOTATION

$v \in \mathbb{B}$ $\quad f_{v,k}(\vec{w}) := f_{U,2}(v, \mathrm{code}(q_s, \vec{w}))$

$v \in \mathbb{B}$ $\quad W_v = \mathrm{dom}(f_{v,1})$ $\longleftarrow$ This is a list of all c.e. subsets of $\mathbb{B}$, indexed by elements of $\mathbb{B}$.

Similarly, $T_v := T_M$ where $v = \mathrm{code}(M)$.

Theorem (s-m-n Theorem)

If $g: \mathbb{B}^{k+1} \dashrightarrow \mathbb{B}$ computable, there there is a total $h: \mathbb{B} \longrightarrow \mathbb{B}$ s.t.

$$\forall v \forall \vec{w} \quad f_{h(v),k}(\vec{w}) = g(\vec{w}, v).$$

Proof. Clearly $\forall v \; g_v(\vec{w}) := g(\vec{w}, v)$ is computable.
[That's not enough to prove the theorem.]
$\vec{w} \longmapsto (\vec{w}, v)$ explicitly performed by some RM M_v.
If M is the RM for g [$f_{M,k+1} = g$], the proof of subroutine lemma provides us with a concrete machine N_v s.t.

$$f_{N_v,k}(\vec{w}) = f_M(\vec{w}, v) = g(\vec{w}, v).$$

Thus $h(v) := \mathrm{code}(N_v)$ gives the s-m-n theorem. q.e.d

STUPID NAME: Original notation for h was S^n_m.

The process of moving a parameter into the index is called

CURRYING

The Recursion Theorem. We close this section by mentioning another important result that follows from the conceptual work performed here: the *Recursion Theorem* or *Fixed Point Theorem*. If $\varphi\colon \mathbb{B} \dashrightarrow \mathbb{B}$ and $w \in \mathbb{W}$, we call w a *fixed point of* φ if $f_{\varphi(w),1} = f_{w,1}$.

Theorem 4.28 (Recursion Theorem or Fixed Point Theorem). If $\varphi\colon \mathbb{B} \to \mathbb{B}$ is total, then φ has a fixed point.