# XVII

Seventeenth Lecture of AUTOMATA & FORMAL LANGUAGES

IN WHICH WE SHALL ENCODE MACHINES

23 November 2024

## REGISTER MACHINES & ARITHMETIC

$s : \mathbb{B} \longrightarrow \mathbb{B}$ successor function (encoding $n \longmapsto n+1$)

is computable

⇝ COUNT-THROUGH ARGUMENTS

[One register is counting up using $s$; while performing some task stepwise until the counter satisfies some condition.]


GÖDEL'S PRIMITIVE RECURSIVE FUNCTIONS:

Basic functions: $s$, constant, projections

Closed under COMPOSITION and RECURSION:

$$h(\vec{w}, \varepsilon) = f(\vec{w})$$
$$h(\vec{w}, s(v)) = g(\vec{w}, v, h(\vec{w}, v))$$

Examples:

Addition $n, m \longmapsto +(n, m)$

Multiplication $n, m \longmapsto \times(n, m)$

Exponentiation $n, m \longmapsto \exp(n, m)$

**Example 4.21.** The functions $f_0$ (*signum function*), $f_1$ (*predecessor function*), $f_2$ (*cut-off subtraction function*), $f_3$ (*absolute difference function*), $f_4$ (*difference check function*), and $f_5$ (*remainder mod q function*) defined below are primitive recursive:

$$f_0(n) := \text{signum}(n) := \begin{cases} 0 & \text{if } n = 0 \text{ and} \\ 1 & \text{if } n > 0; \end{cases}$$

$$f_1(n) := p(n) := \begin{cases} 0 & \text{if } n = 0 \text{ and} \\ n-1 & \text{if } n > 0; \end{cases}$$

$$f_2(n,m) := n \dot{-} m := \begin{cases} 0 & \text{if } n < m \text{ and} \\ n-m & \text{if } n \geq m; \end{cases}$$

$$f_3(n,m) := |n-m| := \begin{cases} n-m & \text{if } n \geq m \text{ and} \\ m-n & \text{if } n < m; \end{cases}$$

$$f_4(n,m) := \begin{cases} 1 & \text{if } n \neq m \text{ and} \\ 0 & \text{if } n = m; \text{ and} \end{cases}$$

$f_5(n) := r(n,m) := k$ if and only if $n \equiv k \pmod{m}$.

$\text{signum}(0) := 0$
$\text{signum}(n+1) := 1$

$p(0) := 0$
$p(n+1) := n$

$n \dot{-} 0 := n$
$n \dot{-} (m+1) := p(n \dot{-} m)$

$|n-m| := (n \dot{-} m) + (m \dot{-} n)$

$f_4(n,m) := \text{signum}(|n-m|)$

$r(0,m) := 0$
$r(n+1,m) := \left(r(n,m)+1\right) \cdot f_4(r(n,m), p(m))$

**Theorem 4.22** Every prim. rec. fn is computable.

**Pf.** By induction: all basic fns are computable.

Compositions of comp. fns are computable.

[The Substitution Lemma]

Left: if $f, g$ are computable, then is $h$ with

$$h(\vec{w}, \varepsilon) = f(\vec{w})$$
$$h(\vec{w}, s(v)) = g(\vec{w}, v, h(\vec{w}, v))$$

Count-through argument. Compute $h(\vec{w}, v)$

Two unused reg: $k$, $\ell$ ← counter, ← current info. Copy both. Write $f(\vec{w})$ into $\ell$. If $v = \varepsilon$, output $\ell$ and halt.

REPEAT [ If $v, t$ are content of $\ell, k$
compute $g(\vec{w}, t, v)$ and replace $\ell$ with that.
Count up ]

UNTIL COUNTER HITS $v$. Output $\ell$ and halt.

q.e.d.

# APPLICATIONS

(1)

**Splitting & merging words.** We use our access to arithmetical functions to define splitting and merging operations on binary words. Consider the arithmetical function

$$z : (i, j) \mapsto \frac{(i + j)(i + j + 1)}{2} + j$$

which is the well-known Cantor *zigzag bijection* that Cantor used to prove the countability of the rational numbers $\mathbb{Q}$ (cf. the proof of Lemma 1.1). This function is a composition of the basic arithmetical functions that we already established are primitive recursive, and hence by Theorem 4.22 computable. So, the maps $(v, w) \mapsto u$ if $\#u = z(\#v, \#w)$ is a computable function. We write $v * w := u$ and call this operation *merging $v$ and $w$ into a single word.*

The merging function is a total computable bijection between $\mathbb{B}^2$ and $\mathbb{B}$. It's inverse taking a word $u$ and finding $v$ and $w$ such that $v * w = u$ can also be performed by a register machine: note that we know that these words must exist, since the Cantor zigzag function is a bijection and that if the formula is valid, then $u, v < w$, so we only need to search through finitely many possible values of $u$ and $v$. This operation is called *splitting $u$ into two words.* It gives rise to two computable total functions $\cdot_{(0)} : \mathbb{B} \to \mathbb{B}$ and $\cdot_{(1)} : \mathbb{B} \to \mathbb{B}$ such that $u_{(0)} * u_{(1)} = u$.

$$z(i,j) := \frac{(i+j)(i+j+1)}{2} + j$$

HOMEWORK

Why is $u \longmapsto \frac{u(u+1)}{2}$ primitive recursive?

MERGING

$$z : \mathbb{N}^2 \longrightarrow \mathbb{N}$$

$$z^{\#} : \mathbb{B}^2 \longrightarrow \mathbb{B}$$

$$v * w := z^{\#}(v, w)$$

SPLITTING

Inverse of $z^{\#} : \mathbb{B} \longrightarrow \mathbb{B}^2$

$$w \longmapsto (w_{(0)}, w_{(1)})$$

performed by RM and

$w \longmapsto w_{(0)}$ are computable

$w \longmapsto w_{(1)}$

# APPLICATIONS

## EXAMPLE SHEET #3

(2)

(39) Let $b\colon \mathbb{B}^k \to \mathbb{B}$ be a total computable function and $R \subseteq \mathbb{B}^{k+1}$ be a computable set. We call $h\colon \mathbb{B}^k \to \{\mathbf{0},\mathbf{1}\}$ the *bounded search function with bound $b$ and query $R$* if

$$h(\vec{v}) = \begin{cases} \mathbf{1} & \text{there is } w \in \mathbb{B} \text{ such that } w \leq_{\text{shortlex}} b(\vec{v}) \text{ and } (\vec{v}, w) \in R \text{ and} \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Show that $h$ is computable. Can you do this without having a bound function?

(40) Suppose $f\colon \mathbb{B}^{k+1} \dashrightarrow \mathbb{B}$ is a partial function, then the partial function $h$ defined by

$$h(\vec{w}) := \begin{cases} v & \text{if for all } u \leq_{\text{shortlex}} v, \text{ we have that } f(\vec{w}, u)\downarrow \text{ and} \\ & \quad v \text{ is } <_{\text{shortlex}}\text{-minimal such that } f(\vec{w}, v) = \varepsilon \text{ or} \\ \dagger & \text{otherwise} \end{cases}$$

is called the *minimisation result of $f$*. Prove that if $f$ is computable, then so is $h$.

(41) Show that all noncontracting $L \subseteq \mathbb{B}$ are computable.

A corollary of T 4.22 & ES#3 (39) is:

If you can give a bound in standard arithmetical functions for the length of search, searching for witnesses is computable. (COUNT-THROUGH)

⇝ (41)

Side remark:

This gives computability for Types 1, 2, 3.

Q: What about Type 0?


Example (40): minimisation.

Church's recursive functions: $\mathcal{R}$ ⊆ $\mathcal{P}$

closure of prim. rec. under minimisation

$\mathcal{P} \subseteq \mathcal{R}$, but $\mathcal{P} \subsetneq \mathcal{R}$ since recursive fns can be partial.

## §4.6 Coding languages and machines

Arbitrary $\Sigma$ : suppose $m$ is such that $|\Sigma| < 2^m$, then code elts of $\Sigma$ as $\{0,1\}^m$.

E.g. $\{a,b,c,d\}$
00 01 10 11

Let $i$ be the encoding fn
$i(w) \in (\{0,1\}^m)^k \subseteq \mathbb{B}$

If $f : W^k \dashrightarrow W$
then $f^i = i \circ f \circ i^{-1}$ [if defined]
is encoding of $f$

Then $f$ is computable if $f^i$ is.
$A \subseteq W^k$ is comp./c.e. if $\{(\vec{w})_i \mid \vec{w} \in A\}$ is.

A RM looks like this

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

Represent states and registers by binary numbers

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

$0, q_S \rightsquigarrow \varepsilon$

$1, q_H \rightsquigarrow \emptyset$

$2, q_2 \rightsquigarrow \mathbb{1}$

$-(\mathbb{1}, \mathbb{1}, \emptyset), ?_\varepsilon(\emptyset, \emptyset, \emptyset), +_\emptyset(\mathbb{1}, \emptyset)$

String = alphabet $\{\emptyset, \mathbb{1}, +_\emptyset, +_{\mathbb{1}}, -, ?_\varepsilon, ?_\emptyset, ?_{\mathbb{1}}, [\,], (\,), ,\}$

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

$-[\mathbb{1}/\mathbb{1}/\emptyset]/?_\varepsilon[\emptyset/\emptyset/\emptyset]/+_\emptyset[\mathbb{1}/\emptyset]$