XXI

Twenty-first lecture of
Automata & Formal Languages
22 November 2023

The typed notes have been updated!
Make sure that you have the version dated

21 NOVEMBER 2023

(or later).

Find a description of the changes under ANNOUNCEMENTS on Moodle.

Automata & Formal Languages
Michaelmas Term 2023
Part II of the Mathematical Tripos
University of Cambridge
Prof. Ev. H. Lowe

Automata & Formal Languages

Contents

CHANGES

1. §4.2 (as discussed in lecture XVII)
2. §4.7 (next pages)
3. §4.8
4. §4.11

EXAMPLE SHEET #4 IS CURRENTLY ON MOODLE (not Faculty website)

IMPORTANT CLARIFICATION OF WHAT HAPPENED IN § 4.7

1. A RM $M = (\Sigma, Q, P)$ always has a fixed alphabet and it can only operate on inputs in $W = \Sigma^*$.
2. If $\Sigma \subseteq \Sigma'$, then a Σ'-RM can operate on words in $W = \Sigma^*$, but not vice versa!
3. In § 4.7, we fixed an alphabet Σ and enlarge it to Σ' [by #, =, &, (,), etc.] and created a

   Σ'-REGISTER MACHINE

   that interprets a code

   $$\text{code}(M) \in W' := (\Sigma')^*$$

   as a **Σ-register machine**.
4. In § 4.6 (not lectured), we had said that the choice of alphabet doesn't matter.

   4.6 Remark on the choice of alphabet

   We defined computability for partial functions $f : W^k \dashrightarrow W$ in terms of Σ-register machines: the instructions and behaviour of register machines are closely tied to their alphabet and register machines can only compute partial functions that use the letters that the machines are built for. Clearly, if $\Sigma \subseteq \Sigma'$ and $f : W^k \dashrightarrow W$ is computable by a Σ-register machine, then it is computable by a Σ'-register machine. But could it be that the notion of computability
5. We can encode words in W' by words in W in a computable way, so the universal register machine U is in fact a

   Σ-register machine

6. So, here's what happens:

(a) Let $c : \mathbb{W} \longrightarrow \mathbb{W}'$ be the computable coding function.

(b) Let U be the universal Σ-register machine.

(c) It receives $(w, v) \in \mathbb{W}$ and interprets them as $c(w), c(v) \in \mathbb{W}'$.

(d) It checks whether $c(w)$ is a code for a Σ-RM M& $c(v)$ is a code for a sequence $\vec{w} \in \mathbb{W}^k$.

(e) If so, it runs $f_{M,k}(\vec{w})$.

(f) Thus $f_{v,k}(\vec{w})$ is the function produced by / coded by the machine $c(v)$ on input $\vec{w}$.

In the typed notes, this is now clarified by using code' for the code in $\mathbb{W}'$ and code for $c^{-1} \circ$ code'.

Now back to §4.8 C.e. sets

Recap

(1) We introduced K & K_0 "halting problem"

(2) Proved that they are both c.e., but not computable.

(3) Introduced notions of being $\Sigma_1, \Pi_1, \Delta_1$

Lecture XX, page 9

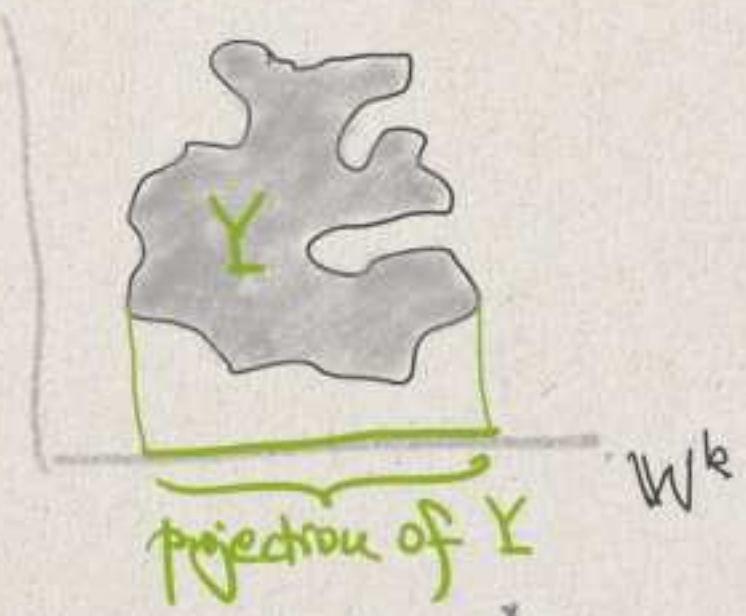
Definition 4.30. A set $X \subseteq \mathbb{W}^k$ is called Σ_1 if there is a computable set $Y \subseteq \mathbb{W}^{k+1}$ such that for all $\vec{w} \in \mathbb{W}^k$, we have

$$\vec{w} \in X \iff \exists v (\vec{w}, v) \in Y.$$

It is called Π_1 if it is the complement of a Σ_1 set; it is called Δ_1 if it is both Σ_1 and Π_1.

The terminology derives from the fact that Σ_1 sets are defined using one existential quantifier and logicians tend to think of existential quantifiers as analogues of sums; similarly, Π_1 sets are defined using one universal quantifier and logicians tens to think of these as analogies of products. The letter Δ comes from the German word "*Durchschnitt*" (intersection) since the class of Δ_1 sets is the intersection of the classes of Σ_1 and Π_1 sets.

$X \subseteq \mathbb{W}^k$ is Σ_1 iff it is the projection of a computable $Y \subseteq \mathbb{W}^{k+1}$:



Proposition Every computable set is Δ_1.

Proposition Every computable set is Δ_1.

Proof. Let X be computable. Since the computable sets are closed under complement, only n.t.s. Σ_1.

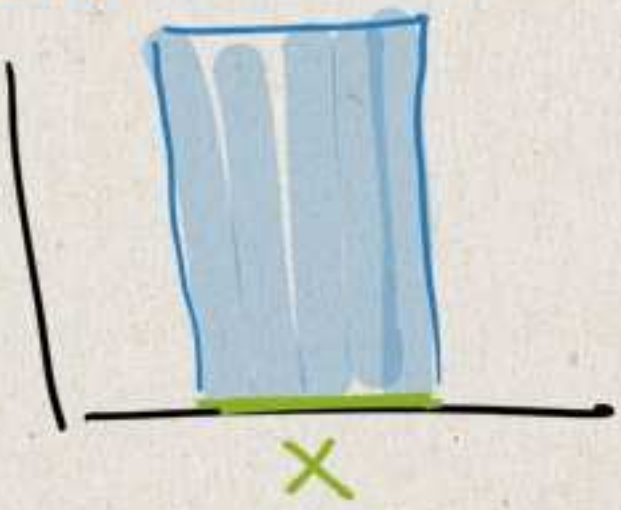
So, we are looking for computable Y s.t.

X is the projection of Y.

$$Y := \{(\vec{w}, v) ; \vec{w} \in X\}$$

CYLINDRE OVER X

Clearly, the projection of Y is X.



The same RM that computes χ_X computes the char. fn of Y.

q.e.d.

Thm If $X \subseteq \mathbb{W}^k$.
Then TFAE:

(i) X is c.e.

(ii) X is Σ_1.

Proof.

(i) $\Rightarrow$ (ii).

Let $X = \mathrm{dom}(f_{M,k})$.

Consider T_M

By observation, we get $\vec{w} \in X \iff \exists v \ (\vec{w}, v) \in T_M$. This shows Σ_1.

Corollary 4.25. The total operation "check whether M has halted with input $\vec{w}$ after at most $\#v$ steps" can be performed by a register machine. We call the corresponding total (characteristic) function

$$t_{M,k}(\vec{w}, v) := \begin{cases} 1 & M \text{ has halted with input } \vec{w} \text{ after at most } \#v \text{ steps and} \\ \varepsilon & \text{otherwise} \end{cases}$$

the *truncated computation of* M.

Note that this is a characteristic function.

TRUNCATED COMPUTATION

$T_M := \{(\vec{w}, v);\ f_{M,k}(\vec{w}) \text{ has halted by time } \#v\}$

$\hat{T}_M := \{(\vec{w}, v, v);\ f_{M,k}(\vec{w}) \text{ has halted by time } \#v \text{ with output } v\}$

Suppose $f : \mathbb{W}^{k+1} \dashrightarrow \mathbb{W}$ is a partial function, then the partial function h defined by

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

is called the *minimisation result of* f.

(ii) $\Rightarrow$ (i). Let $Y \subseteq \mathbb{W}^{k+1}$ computable s.t.
$\vec{w} \in X \iff \exists v \ (\vec{w}, v) \in Y$.

Define $g(\vec{w}, v) := \begin{cases} \varepsilon & \text{if } (\vec{w}, v) \in Y \\ a & \text{if } (\vec{w}, v) \notin Y. \end{cases}$

This is $\chi_{\mathbb{W}^{k+1} \setminus Y}$, so computable

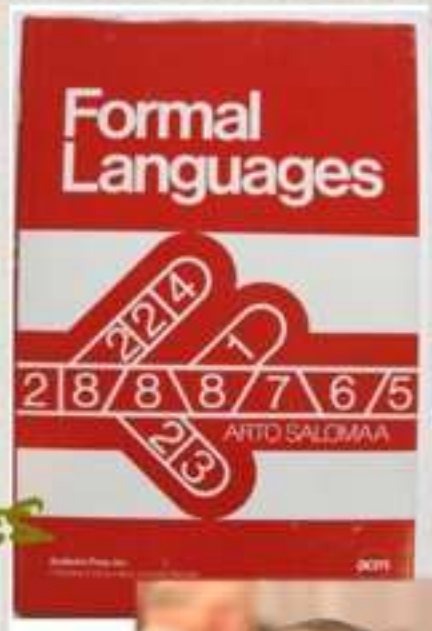
Apply minimisation to g and obtain h, so $h(\vec{w})$ is the least v s.t. $(\vec{w}, v) \in Y$ (if exists)

Thus $X = \mathrm{dom}(h)$. q.e.d.

APPLICATION

Thm $L \subseteq W$ is type 0 iff L is c.e.

Proof. We'll only prove $\Rightarrow$. The other direction is in Salomaa's book FORMAL LANGUAGES [Also proof sketch in typed notes.]



Arto SALOMAA

Let's do $\Rightarrow$.

As before, we enlarge our alphabet. If G is any grammar with alphabet Σ and set of variables V, let

$$\Sigma' := \Sigma \cup V \cup \{\Rightarrow\}.$$

If $W' := (\Sigma')^*$, then every $w \in W'$ can be interpreted as a sequence of strings in Ω^* separated by the symbol $\Rightarrow$.

$$S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \cdots \Rightarrow \alpha_n$$

RM can check whether $w \in W'$ is a code for a G-derivation, whether it starts with S and whether it ends in $v \in W$.

$D := \{(v,w) ;\ v \in W,\ w$ is a G-derivation, starting with S, ending with $v\}$ is computable.

$v \in L \iff \exists w\ (v,w) \in D$. So L is g.e.d. $\Sigma_1 \Rightarrow$ c.e.

ZIGZAG METHOD

Remember zig zag fn

$$* : \mathbb{W} \times \mathbb{W} \longrightarrow \mathbb{W}$$ computable bijection

with components of inverse

$$w \longmapsto w_{(0)} \quad \text{s.t. } w_{(0)} * w_{(1)} = w.$$
$$w \longmapsto w_{(1)}$$

Suppose $Z \subseteq \mathbb{W}^{k+2}$ computable

and $\vec{w} \in X \iff \exists v \exists u \ (\vec{w}, v, u) \in Z$

Then X is c.e.

Two quantifiers can be folded into one

WHY?

$$Y := \{ (\vec{w}, v) \in \mathbb{W}^{k+1} ; (\vec{w}, v_{(0)}, v_{(1)}) \in Z \}$$

then $\vec{w} \in X \iff \exists v \ (\vec{w}, v) \in Y.$

so X is Σ_1, so c.e.

An example Suppose $f : \mathbb{W}^{k+1} \dashrightarrow \mathbb{W}$ computable and $X = \{\vec{w} \,;\, \exists v\, f(\vec{w}, v)\downarrow\}$

CLAIM: X is c.e.

[If $f = f_{M,k}$, consider truncated computation T_M
$T_M = \{(\vec{w}, v, u) \,;\, f(\vec{w}, v)$ has halted by time $\#u\}$

$\vec{w} \in X \iff \exists v \exists u\ (\vec{w}, v, u) \in T_M$.
So by zigzag, X is Σ_1 = c.e.]

	ε	**0**	**1**	**00**	**10**	**01**	**11**	**000**	⋯
ε	$(\vec{w}, \varepsilon, \varepsilon)$	$(\vec{w}, \varepsilon, \mathbf{0})$	$(\vec{w}, \varepsilon, \mathbf{1})$	$(\vec{w}, \varepsilon, \mathbf{00})$	$(\vec{w}, \varepsilon, \mathbf{10})$				⋯
0	$(\vec{w}, \mathbf{0}, \varepsilon)$	$(\vec{w}, \mathbf{0}, \mathbf{0})$	$(\vec{w}, \mathbf{0}, \mathbf{1})$	$(\vec{w}, \mathbf{0}, \mathbf{00})$	$(\vec{w}, \mathbf{0}, \mathbf{10})$				⋯
1	$(\vec{w}, \mathbf{1}, \varepsilon)$	$(\vec{w}, \mathbf{1}, \mathbf{0})$	$(\vec{w}, \mathbf{1}, \mathbf{1})$	$(\vec{w}, \mathbf{1}, \mathbf{00})$	$(\vec{w}, \mathbf{1}, \mathbf{10})$				⋯
00	$(\vec{w}, \mathbf{00}, \varepsilon)$	$(\vec{w}, \mathbf{00}, \mathbf{0})$	$(\vec{w}, \mathbf{00}, \mathbf{1})$	$(\vec{w}, \mathbf{00}, \mathbf{00})$	⋱				
10	$(\vec{w}, \mathbf{10}, \varepsilon)$	$(\vec{w}, \mathbf{10}, \mathbf{0})$	$(\vec{w}, \mathbf{10}, \mathbf{1})$	⋱					
⋮	⋮	⋮	⋮						

Figure 5: The search space $\mathbb{W}^2$ (or, more precisely, $\{\vec{w}\} \times \mathbb{W}^2$) traversed by Cantor's zigzag function as in the argument of Example 4.37.

If we want to traverse $\mathbb{W} \times \mathbb{W}$, doing it horizontally or vertically doesn't necessarily work: we might run out of time before we're done.

SOLUTION: Diagonal: infinitely processes in parallel !!!

Application 1

Proposition $X \subseteq \mathbb{N}$ X c.e. $\iff$ $\exists$ computable g s.t. $X = \mathrm{ran}(g)$

Proof. $\Rightarrow$ Let f be computable s.t. $X = \mathrm{dom}(f)$.

Define $g(w) := \begin{cases} w & \text{if } f(w)\downarrow \\ \uparrow & \text{o/w} \end{cases}$

Clearly g computable & $X = \mathrm{ran}(g)$.

$\Leftarrow$ Let $X = \mathrm{ran}(f_{M,1})$; consider the truncated computation with output

(on page 6 : ✳)

$\hat{T}_M = \{(v,u,w)$; comp. of M w/ input v has halted by time $\#u$ with output $w\}$

computable

$w \in X \iff \exists v \exists u \ (v,u,w) \in \hat{T}_M$

So Σ_1 = c.e. by the zig-zag method. q.e.d.

Application 2

Proposition X is computable $\iff$ X is Δ_1.

Proof. $\Longrightarrow$ already proved earlier.

If X is Δ_1, i.e., Σ_1 & Π_1.

$$\vec{\omega} \in X \iff \exists v\ (\vec{\omega}, v) \in Z_0$$
$$\vec{\omega} \notin X \iff \exists v\ (\vec{\omega}, v) \in Z_1$$

for Z_0, Z_1 computable.

$Y_0 := \{(\vec{\omega}, v) ;\ \#v_{(0)}$ is even & $(\vec{\omega}, v_{(1)}) \in Z_0\}$
$Y_1 := \{(\vec{\omega}, v) ;\ \#v_{(0)}$ is odd & $(\vec{\omega}, v_{(0)}) \in Z_1\}$
$Y := Y_0 \cup Y_1$. So Y is a computable set.

By minimization, let h be the function s.t.
$h(\vec{\omega})$ is the least v s.t. $(\vec{\omega}, v) \in Y$.
Since every $\vec{\omega}$ is either in X or not, h is a total function.

$$\chi_X(\vec{\omega}) = \begin{cases} a & \text{if } \#h(\vec{\omega})_{(0)} \text{ is even} \\ \varepsilon & \text{if } \#h(\vec{\omega})_{(0)} \text{ is odd} \end{cases}$$

and therefore χ_X is computable, so X is computable. q.e.d.