RECURSION THEORY

LECTURE II

22 October 2021

COPIED FROM LECTURE I

Def. A <u>MODEL OF COMPUTATION</u> is a triple $\mathbb{M} = (\Sigma, \Phi, h)$

where Σ is an <u>alphabet</u>

Φ is a <u>transition function</u>

h is an <u>output function</u>.

$\Phi : \boxed{\Sigma^*} \times \boxed{\Sigma^*} \longrightarrow \Sigma^*$

where $\mathrm{lh}(\Phi(v,w)) \leq \mathrm{lh}(w) + 1$

PROGRAM

INPUT STORAGE

This already incorporates the software principle.

h is a partial function

$h : \Sigma^* \dashrightarrow \Sigma^*$

$H := \mathrm{dom}(h)$; its elements are called the <u>halting markers</u>

and $h(w)$ is a subword of w.

Recursive definition:

$w_0 \sim C_{\mathbb{M}}(P, w, 0) := w$

$w_{n+1} \sim C_{\mathbb{M}}(P, w, n+1) := \Phi(P, C(P,w,n))$

Called the $\mathbb{M}$-computation with program P and input w.

$\Phi : \boxed{\Sigma^*} \times \boxed{\Sigma^*} \longrightarrow \Sigma^*$

where $\mathrm{lh}(\Phi(v,w)) \leq \mathrm{lh}(w) + 1$

PROGRAM

INPUT STORAGE

This already incorporates the software principle.

Def. The $\mathbb{M}$-computation with program P and input w <u>halts</u> if there is some $i \in \mathbb{N}$ s.t.

$$C_{\mathbb{M}}(P, w, i) \in H.$$

In this case, we say that the $\mathbb{M}$-computation with P/w outputs v if for the least i s.t.

$$C_{\mathbb{M}}(P, w, i) \in H$$

we have

$$v = h(C_{\mathbb{M}}(P, w, i)).$$

Let's write <u>$o_{\mathbb{M}}(P,w)$</u> for this.

The computation gives us a partial function for each program P:

$$f_P : \Sigma^* \dashrightarrow \Sigma^*$$

$$f_P(w) := \begin{cases} o_{\mathbb{M}}(P,w) & \text{if the comp halts} \\ \uparrow & \text{o/w.} \end{cases}$$

MODEL OF COMPUTATION

$$\mathbb{M} := (\Phi, \Sigma, h)$$

Φ — transition function; Σ — alphabet; h — output function

$$\Phi : \Sigma^* \times \Sigma^* \longrightarrow \Sigma^* \times \{0,1\}$$

Σ^* — PROGRAM, Σ^* — INPUT; Σ^* — NEXT STEP, $\{0,1\}$ — WHETHER HALTS OR NOT

$$h : \Sigma^* \longrightarrow \Sigma^*$$

s.t. $h(w)$ is a subword of w

$$C_{\mathbb{M}}(P, w, 0) := w$$

$$C_{\mathbb{M}}(P, w, u+1) := v \quad \text{where}$$

$$\Phi(P, C_{\mathbb{M}}(P, w, u)) = (v, b)$$

If $b = 1$, we say program P halts at input w at time $u+1$.

We say that P halts at input w if there is some i s.t. P halts at w at time i.

If P halts at input w and i is the least such halting time, then

$$o_{\mathbb{M}}(P, w) := h(C_{\mathbb{M}}(P, w, i)).$$

if $P \in \Sigma^*$, we define

$$f_P : \Sigma^* \dashrightarrow \Sigma^*$$

by

$$f_P(w) := \begin{cases} o_{\mathbb{M}}(P,w) & \text{if } P \text{ halts at } w \\ \uparrow & \text{o/w.} \end{cases}$$

REMARKS on the character of the notion of "model of computation".

1. Our notion of "model of computation" is certainly not sufficient for being a reasonable model of computation.

Reason: You can make everything $\mathbb{M}$-computable with the right choice of $\mathbb{M}$.

Why? Let $A \subseteq \Sigma^*$ arbitrary.
Define transition function Φ by

$$\Phi(P,w) := \begin{cases} (\sigma, 1) & \text{if } w \in A \\ (\varepsilon, 1) & \text{if } w \notin A \end{cases}$$

This is a (strange) model of computation s.t. χ_A is $\mathbb{M}$-computable. → CONTINUED ON P.5.

Def. A partial function $f : \Sigma^* \dashrightarrow \Sigma^*$ is called M-computable if $\exists$ some $P \in \Sigma^*$ s.t. $f = f_P$.

Def. If $A \subseteq \Sigma^*$; fix some $\sigma \in \Sigma$. Then

$$\chi_A : \Sigma^* \longrightarrow \Sigma^*$$

is called the characteristic function of A if

$$\chi_A(w) = \begin{cases} \varepsilon & \text{if } w \notin A \\ \sigma & \text{if } w \in A. \end{cases}$$

Then A is called M-computable if χ_A is M-computable.

2. So, our conditions for models of computation are minimal requirements for something to be a model of computation.

3. Even that is questionable:

Suppose $M = (\Phi, \Sigma, h)$ is a (wc) model of computation. Define

$\overline{\Phi}(P, w) := (v, c)$

Check $\Phi(P, w) =: (u, b)$.

If $b = 1$, let $v := u$ and $c := 1$.

If $b = 0$, check $\Phi(P, u) = (u', b')$ and let $v := u'$ and $c := b'$.

Define $\overline{M} := (\overline{\Phi}, \Sigma, h)$.

By construction, any partial fn

$f : \Sigma^* \dashrightarrow \Sigma^*$

is M-computable iff it is $\overline{M}$-computable.

So $\overline{M}$ is intuitively as good a "model of computation" as M, and yet it may fail the "length increases by at most 1" criterion.

Proposition 1 There are at most countably many computable partial functions.

Proof. Σ is a finite alphabet, so Σ^* is countably infinite.

Thus by definition

$$P \longmapsto f_P$$

is a surjection from Σ^* onto the set of TM-computable partial fns.

q.e.d.

Proposition 2 There are uncountably many partial functions from Σ^* to Σ^* and thus there are non-computable partial functions.

Proof. By Cantor's Thm, already the set of total functions from Σ^* to Σ^* is uncountable.

q.e.d.

THE HALTING PROBLEM

In Computer Science, subsets $A \subseteq \Sigma^*$ are called **problems**:

think of A as a task where you are given $w \in \Sigma^*$ and need to determine whether $w \in A$ or not.

Def. The halting problem is denoted by K and is defined as follows:

$$w \in K :\Longleftrightarrow f_w(w)\downarrow .$$

Goal Show that K is not $\mathcal{M}$-computable.

This will need a few additional properties of $\mathcal{M}$:

COMPOSITIONALITY

If P and Q are programs then there is an $R \in \Sigma^*$ s.t.

$$f_R = f_P \circ f_Q .$$

Remark. Compositionality does not follow from the other properties of being a "model of computation". But it is very reasonable in the sense that "programmable in C++" will have this property.

We define the following partial functions

$$d_{xy} : \Sigma^* \dashrightarrow \Sigma^*$$

where $x, y \in \Sigma^* \cup \{\uparrow\}$ as follows

$$d_{xy}(w) := \begin{cases} x & \text{if } w = \varepsilon \\ y & \text{if } w \neq \varepsilon \end{cases}$$

CASE DISTINCTION FUNCTIONS

CASE DISTINCTION

Every case distinction function d_{xy} is M-computable.

Remark. Again, it doesn't follow from the other properties, but is reasonable since we know how to do it in C++.

Theorem If M is a model of computation satisfying Case Distinction & Compositionality, then K is not M-computable.

Proof. Assume towards a contradiction that it is:

K is M-computable, i.e., χ_K is M-computable

$$\chi_K(w) := \begin{cases} \varepsilon & \text{if } w \notin K \Leftrightarrow f_w(w)\uparrow \\ \sigma & \text{if } w \in K \Leftrightarrow f_w(w)\downarrow \end{cases}$$

Let P be a program s.t.

$$\underline{\chi_K} = \underline{f_P}.$$

Consider

$$g := d_{\underline{\varepsilon\uparrow}} \circ f_P.$$

(*) By our two extra assumptions, g is M-computable.

$$g(w) = \begin{cases} \uparrow & \text{if } w \in K \\ \varepsilon & \text{if } w \notin K \end{cases}$$

If $w \in K \longrightarrow f_P(w) = \chi_K(w) = \sigma$, so $g(w)\uparrow$.
If $w \notin K \longrightarrow f_P(w) = \chi_K(w) = \varepsilon$, so $g(w)\downarrow = \varepsilon$.
By (*) g is M-computable, so ex. $G \in \Sigma^*$, s.t.

$$g = f_G.$$

Now calculate $f_G(G)$:

$$\left.\begin{array}{ll}\boxed{G \in K \quad \downarrow} \\ \boxed{G \notin K \quad \uparrow}\end{array}\right\} = f_G(G) = g(G) = \begin{cases} \boxed{\uparrow \quad \text{if } G \in K} \\ \boxed{\varepsilon \quad \text{if } G \notin K} \end{cases}$$

Thus, we obtained a contradiction.

q.e.d.

REDUCTION FUNCTIONS

A function $f: \Sigma^* \longrightarrow \Sigma^*$ is called a reduction function if it is total and computable.

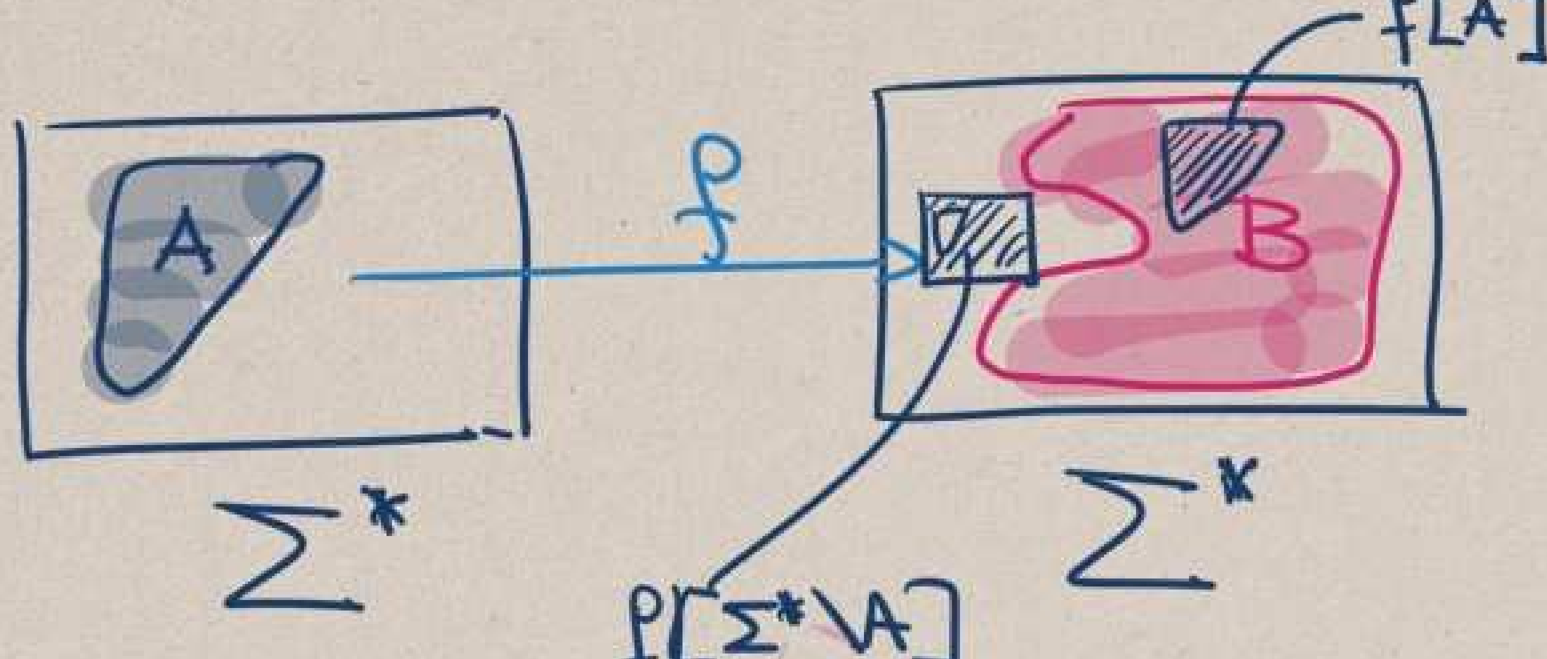
If $A, B \subseteq \Sigma^*$, we write

$$A \leq_m B$$

A is many-one-reducible to B

if there is a reduction fn f s.t.

f.a. w $w \in A \Longleftrightarrow f(w) \in B$.



Remark 1 "many-one" because we're not demanding that f is an injection; there is also a notion called "one-one reducibility" with injections: $\leq_1$.

Remark 2 This general definition occurs in many parts of mathematics, e.g., if you replace "computable" with "continuous", you get the notion of Wadge reduction; if you use "polynomial time computable", you get polynomial reductions, etc.

William W. Wadge

Lemma If $\mathcal{M}$ is compositional and $A, B \subseteq \Sigma^*$ and $A \leq_m B$ and B is $\mathcal{M}$-computable, then A is $\mathcal{M}$-computable.

$\boxed{\begin{array}{l} f: \Sigma^* \to \Sigma^* \\ w \in A \leftrightarrow f(w) \in B \end{array}}$

Proof. If B is computable, that means χ_B is computable. But then

$$\chi_A = \underbrace{\chi_B}_{\text{comp.}} \circ \underbrace{f}_{\text{comp.}}$$

so by compositionality, χ_A is computable.

q.e.d.

Corollary Under the same assumptions, if $A \leq_m B$ and A is not $\mathcal{M}$-computable, then B is not $\mathcal{M}$-computable.

This means: if we wish to show that a set B is not computable, it is enough to show $K \leq_m B$.

[Compositionality + Case Dist.]

Properties of $\leq_m$:

1. Clearly reflexive under the assumption that the identity function $\mathrm{id}: \Sigma^* \longrightarrow \Sigma^*$, $w \longmapsto w$ is $\mathbb{M}$-computable.
2. Transitivity follows from compositionality of $\mathbb{M}$.
3. In general, not anti-symmetric.

<u>Def.</u> A relation R on a set X is called a <u>partial preorder</u> if R is reflexive and transitive.

If we define

$$A \equiv_m B \quad :\Longleftrightarrow \quad A \leq_m B \text{ and } B \leq_m A$$

many-one equivalent

this gives an equivalence relation.

[In general, if R is a partial preorder and $x \equiv x' :\Longleftrightarrow xRx'$ and $x'Rx$, then $(X/_{\equiv}, R)$ is a partial order.]

The $\equiv_m$ - equivalence classes, called many-one degrees are clusters of sets that are computationally the same.

More about the structure of $\leq_m$:

(4) If $A \neq \Sigma^*$, then $\emptyset \leq_m A$.

PROVIDED THAT CONSTANT FUNCTIONS ARE COMPUTABLE

if $w \in \Sigma^*$ $\quad c_w(v) := w$ for all $v \in \Sigma^*$.

[Proof. If $A \neq \Sigma^*$, there is some $w \notin A$.

Consider c_w

$$\underbrace{v \in \emptyset}_{\text{false for all } v} \iff \underbrace{c_w(v) \in A}_{\text{false for all } v.}$$

Thus c_w is a reduction function from $\emptyset$ to A.

(5) $\emptyset \not\leq_m \Sigma^*$.

(6) $\Sigma^* \not\leq_m \emptyset$.

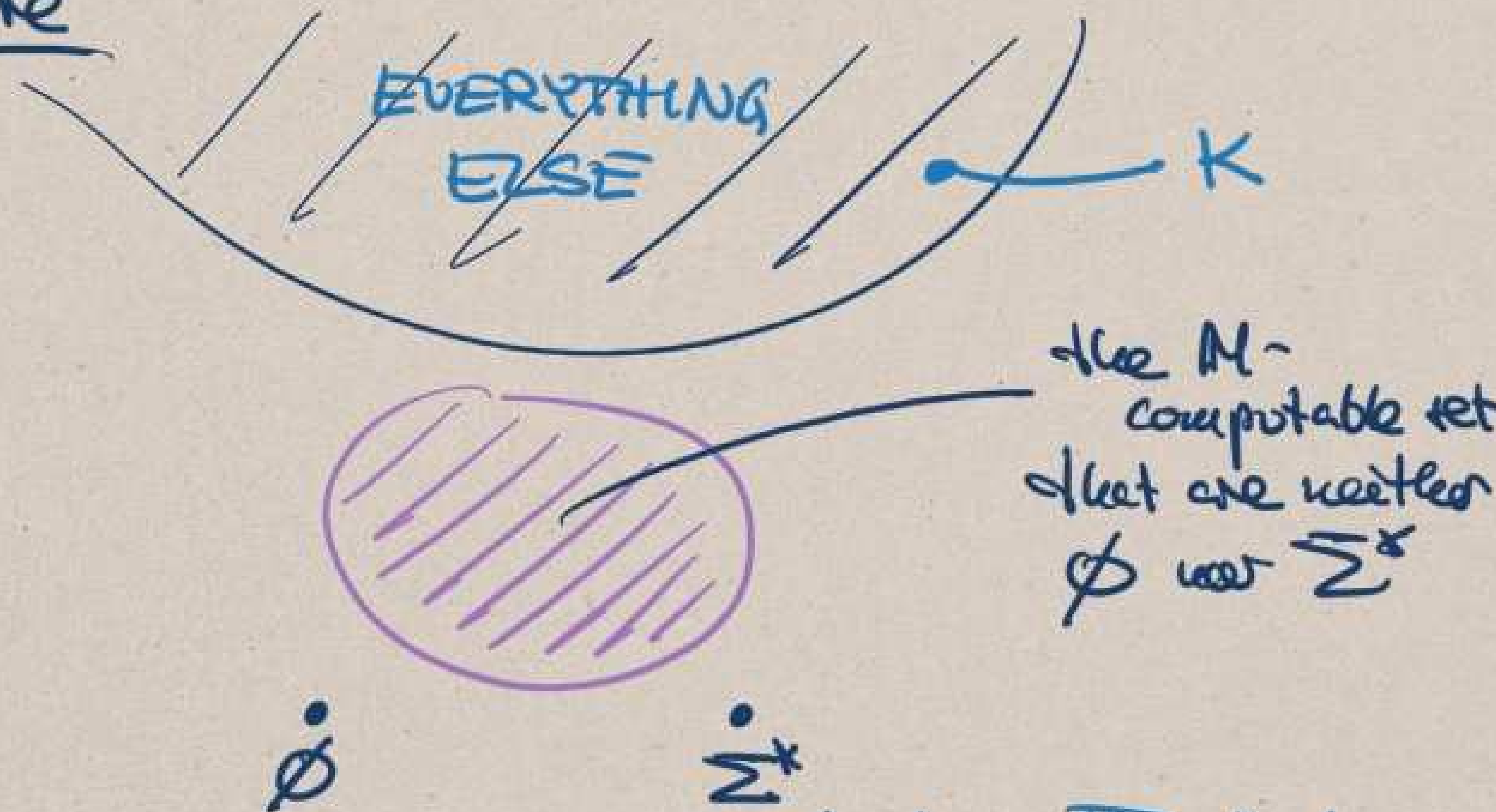
(7) If $A \neq \emptyset$, then $\Sigma^* \leq_m A$.

Provided that constant functions are computable.

[If $A \neq \emptyset$, let $w \in A$.
Then c_w reduces Σ^* to A:

$$\underbrace{v \in \Sigma^*}_{\text{always true}} \Longleftrightarrow \underbrace{\underbrace{c_w(v)}_{w} \in A}_{\text{always true.}}$$]

Picture



We'll prove concretely in lecture III that this picture is correct.