IV FOURTH LECTURE

CDI : Computability
Decidability
Incompleteness

5 May 2023

Recap

REGISTER MACHINE M
input $\vec{w} \in \mathbb{W}^4$
Computation $C(\vec{w}, i)$
halting, halting times, output : $F: \mathbb{W}^4 \dashrightarrow \mathbb{W}^4$

→ Operations performed
→ Questions answered

$f_{M,k}: \mathbb{W}^k \dashrightarrow \mathbb{W}$ $f_{M,k}(\vec{w}) = v$ if M halts at input $\vec{w}$ and outputs $\vec{v} = (v_0, \dots, v_n)$ with $v_0 = v$

Computable (partial) function

→ χ_A characteristic fu
→ ψ_A pseudo characteristic fu (partial)

Computable / computably enumerable (c.e) set

Questions for LIV

Which ordinary mathematical fns are computable?

Problem Ordinary math. fns do not usually live on W^k.

So, if $f: \mathbb{N} \longrightarrow \mathbb{N}$
$\mathbb{Z} \longrightarrow \mathbb{Z}$
$\mathbb{Q} \longrightarrow \mathbb{Q}$
is an "ordinary math fn", we need to interpret it as a function $W \longrightarrow W$.

First idea

Since W is countable, there is a bijection

$g: W \longrightarrow \mathbb{N}$.

So, we could say that $A \subseteq \mathbb{N}$ is computable if and only if $g^{-1}(A)$ is computable

Problem There is no reason to believe that an arbitrarily picked bijection g is even remotely "computable", so this notion might not reflect the intuitive concept.

C.4 The shortlex ordering –

Assume that $\Sigma = \{a_0, \dots, a_n\}$ is coming with a total order

$$a_0 < a_1 < \dots < a_n .$$

Let's define for

$$w = b_0 \dots b_m$$
$$v = c_0 \dots c_\ell$$

$$w < v \quad :\Longleftrightarrow \quad |w| < |v| \quad \text{OR}$$
$$|w| = |v| \quad \text{AND} \quad w \neq v$$
$$\text{AND if } i \text{ is least s.t. } b_i \neq c_i, \text{ then } b_i < c_i$$

Q1: length of sequence

SHORTLEX order

Q2: earliest [illegible]

Example $\Sigma = \{0,1\}$ with $0 < 1$.

$$\varepsilon < 0 < 1 < 00 < 01 < 10 < 11$$
$$< 000 < 001 < 010 < 011$$
$$< 100 < 101 < 110 < 111$$
$$< 0000 \dots$$

Cf. p.5 for a brief explanation why SHORTLEX is not the usual lexicographic order.

PROPERTIES

$<$ is irreflexive: $\forall w \quad w \not< w$

$<$ is transitive: $\forall v, w, u$

$v < w \;\&\; w < u \longrightarrow v < u$

$<$ is trichotomous: $\forall v, w$

$v < w \vee v = w \vee w < v$.

Proposition There is an isomorphism between $(\mathbb{W}, <)$ and $(\mathbb{N}, <)$.

Proof. Write $\mathrm{pred}_<(w) := \{ v \in \mathbb{W} ;\ v < w \}$

By construction $\mathrm{pred}_<(w)$ is always a finite set [this is where the finiteness of Σ is very important].

So define

$$\# : \mathbb{W} \longrightarrow \mathbb{N} \quad \text{by}$$

$$w \longmapsto |\mathrm{pred}_<(w)|.$$

I claim that this is an isomorphism.

By transitivity, we get

$$w < v \Longrightarrow \mathrm{pred}_<(w) \subseteq \mathrm{pred}_<(v),$$

so $\#w \leq \#v$.

By irreflexivity, we have

$$w < v \Longrightarrow w \in \mathrm{pred}_<(v)$$
$$w \notin \mathrm{pred}_<(w)$$
$$\Longrightarrow \#w < \#v.$$

Here is an immediate successor operation: how do we calculate the "next" word?

```
  1 1 1                  1 1 1
  1010111                3750999
  0000001                      1
  -------                -------
  1011000                3751000
```

This is the ordinary addition algorithm we know from elementary school.

q.e.d.

Shortlex ≠ lexicographic

$$010 < 011 < 0110 < 01100 < \ldots$$
$$< 0111$$

In lexicographic order 0111 has ∞ many predecessors!

Theorem ① The set $\{(w,v);\ w<v\}$ is computable.

② The function $s : W \longrightarrow W$ s.t. $\#(s(w)) = \#(w) + 1$ is computable.

Proof. Idea Try to answer the two questions in the definition of the shortlex order.

Q1 What is the relationship between $|w|$ & $|v|$?

Three possible answers:

A_1 $|w| < |v|$

A_2 $|v| < |w|$

A_3 $|w| = |v|$.

Copy w, v into unused registers; remove symbols from w & v one-by-one until at least one of them is empty.

If this happens at the same time: A_3;

if one of them is empty before the other A_1/A_2, respectively.

Q2 Which word is lexicographically smaller?

Possible answers:

A_1' w

A_2' v

A_3' w = v

Copy w and v in reverse order into two owned registers; then remove the final symbol one by one, continuing if they are the same.

Three possible outcomes:

- If both registers are empty, at the end, answer A_3.
- At some point I remove different symbols; then one of them is smaller ie < or >; answer A_1' / A_2', respectively.
- If I remove the same symbol, repeat.

So now if (w,v) is given, first answer Q1.

if A_1 ⟶ "Yes".
if A_2 ⟶ "No".
if A_3, continue.

Then answer Q2:

if A_1' ⟶ "Yes".
if A_2'/A_3' ⟶ "No".

[q.e.d. ①]

Proof of ②. Remember:

$s(10101) = 1011000$.
$s(111) = 00000$

Check the last symbol of w. If it is $a_i \neq a_n$, then just change it to a_{i+1}.

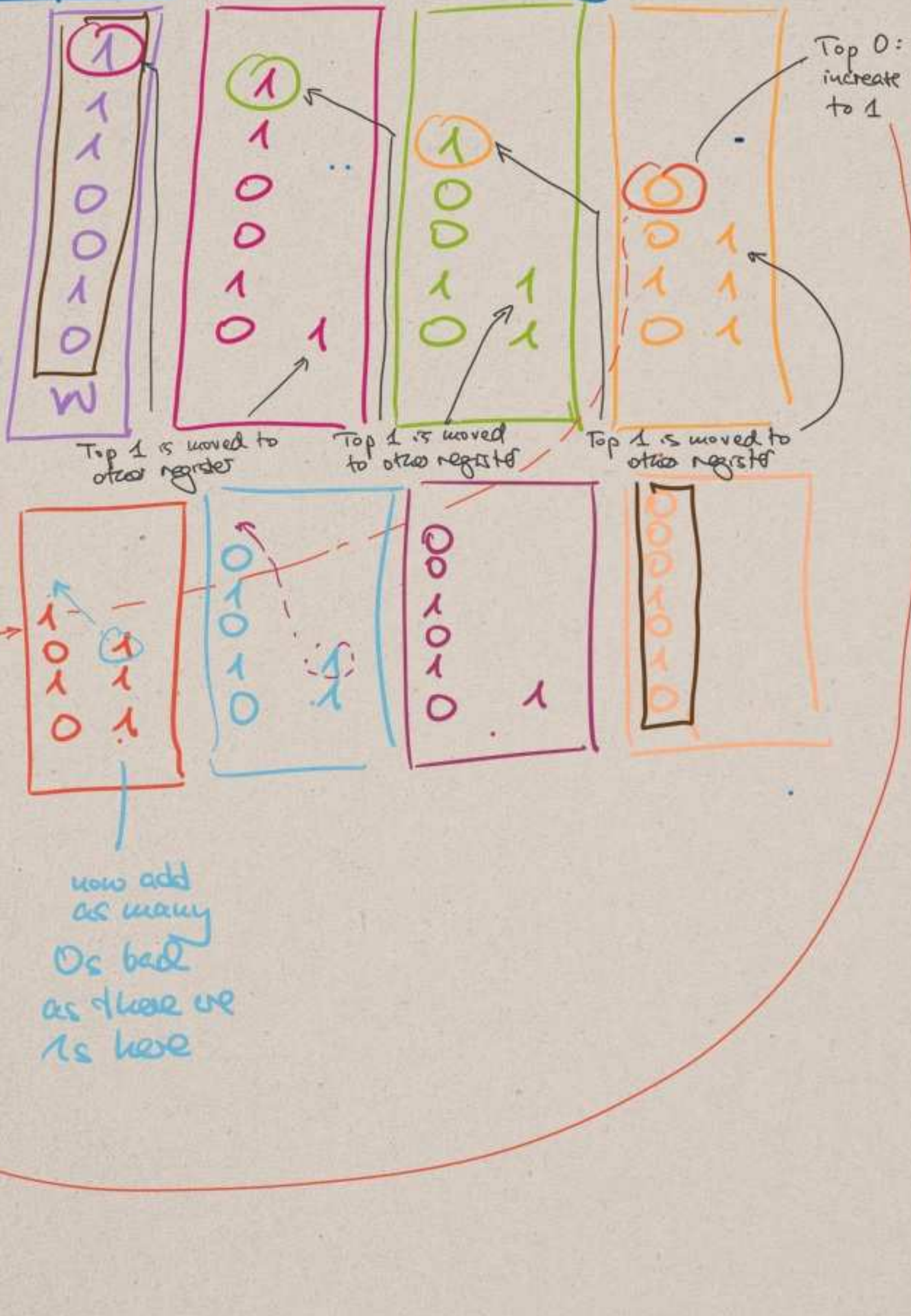
[$s(110) = 111$]

If it is a_n, then remove iteratively all instances of a_n from the end of word, storing them in some register.

Case 1 Remainder is empty. Then add as many a_0 symbols as there are symbols in the storage register and one extra. *

Case 2 Remainder is nonempty with final symbol $a_i \neq a_n$. Change this to a_{i+1} and add as many a_0's as there are symbols in storage. *

Example $\Sigma = \{0,1\}$



CS Church's Recursive Functions

TURING
1912-1954

Turing's approach to COMPUTABILITY was via closure properties of classes of functions.

CHURCH
1903-1995

BASIC FUNCTIONS

projections

$$\pi_{k,i} : \mathbb{W}^k \longrightarrow \mathbb{W}$$
$$(w_0, \dots, w_{k-1}) \longmapsto w_i$$

Remark: We already saw that these are computable

NOTE
All basic functions are total.

Constant functions

$$c_{k,\varepsilon} : \mathbb{W}^k \longrightarrow \mathbb{W}$$
$$\vec{w} \longmapsto \varepsilon$$

Again: already seen that they are computable.

successor function

$$s : \mathbb{W} \longrightarrow \mathbb{W}$$
$$w \longmapsto v$$
if $\#v = \#w + 1$.

Just prove that it's computable.

Closure properties

(1)

Suppose $f : \mathbb{W}^m \dashrightarrow \mathbb{W}$ and $g_1, ..., g_m : \mathbb{W}^k \dashrightarrow \mathbb{W}$ are partial functions, then the partial function h defined by

$$h(\vec{w}) := f(g_1(\vec{w}), ..., g_m(\vec{w}))$$

COMPOSITION

is called the *composition of f with $(g_1, ..., g_m)$*. The notational convention used for operations applies here as well: if any term on the right hand side is undefined, then so is the left hand side.

(2)

Suppose $f : \mathbb{W}^k \dashrightarrow \mathbb{W}$ and $g : \mathbb{W}^{k+2} \dashrightarrow \mathbb{W}$ are partial functions, then the function h defined by the recursion equations

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

RECURSION

is called the *recursion result of f and g*.

(3)

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

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

MINIMISATION

is called the *minimisation result of f*.

(1) Remark The concatenation lemma tells us that if $f, g_1, ..., g_m$ are all computable, and h is the composition of f with $(g_1, ..., g_m)$, then h is computable.

(2) Compare to standard notation

$$h(\vec{w}, 0) := f(\vec{w})$$
$$h(\vec{w}, n+1) := g(\vec{w}, n, h(\vec{w}, n))$$

(3) $h(\vec{w})$ is the least v s.t. for all $u \leq v$, $f(\vec{w}, v)\downarrow$ and $f(\vec{w}, v) = \varepsilon$. (<-)

Note $h(\vec{w})\uparrow$ if $f(\vec{w}, v) \neq \varepsilon$ for all v, or $f(\vec{w}, v)\uparrow$ before it is empty.

Remark on ① & ②

If $f, g_1, \dots, g_m, g$ are total, then so are the composition of f with $(g_1, \dots, g_m)$ and the recursion result of f and g.

Therefore the closure of the basic functions (all total) under composition and recursion will only contain total fcts.

Remark on ③

Minimisation easily produces partial functions.

E.g. let $v \neq \varepsilon$

$f(w) := v$ for all w

then h is the nowhere defined partial function.

Definition (a) The smallest class containing all basic functions & closed under ① and ② is called the

PRIMITIVE RECURSIVE FUNCTIONS

(b) The smallest class containing all basic fcts & closed under ①, ②, & ③ is called the

RECURSIVE FUNCTIONS

Historical note

The primitive recursive fns were introduced in Gödel's famous 1931 paper, but called "recursive functions".

Goal: Show the following theorem

THEOREM: Every recursive fn is computable.

[We already know that all basic functions are computable; we already know that the computable fns are closed under (1).

If we can show that computable fns are closed under (2), (3), then since recursive fns are smallest such class:

Rec ⊆ Comp.]

Now let's see what is recursive.

$$h : W \times W \longrightarrow W$$

$$h(w, \varepsilon) := w$$
$$h(w, s(v)) := s(h(w, v))$$

By ②, h is recursive.

Think of it as:

$$h : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$$

$$h(n, 0) := n$$
$$h(n, m+1) := h(n, m) + 1.$$

This is the so-called GRASSMANN recursive definition of addition.

$$n + m := h(n, m).$$

Same idea ⇒ GRASSMANN multiplication:

$$h(n, 0) := 0$$
$$h(n, m+1) := h(n, m) + n$$

(using addition)

On W with $h : W \times W \longrightarrow W$ for addition

$$m(w, \varepsilon) := \varepsilon$$
$$m(w, s(v)) := h(m(w, v), w)$$