XXI

TWENTY-FIRST LECTURE OF AUTOMATA & FORMAL LANGUAGES

22 NOVEMBER 2022

WHERE ARE WE IN THE OVERALL NARRATIVE OF AUTOMATA & FORMAL LANGUAGES ?

	Chapter 2 type 3 (regular)	**Chapter 3** type 2 (context-free)	**Chapter 1** type 1 (context-sensitive)	**Chapter 4** type 0
Closure properties.				
Concatenation	✓	✓	✓	
Union	✓	✓	✓	
Intersection	✓	×		
Complementation	✓	×		
Difference	✓	×		
Decision problems.				
Word problem	✓	✓	✓	
Emptiness problem	✓	✓		
Equivalence problem	✓	×		

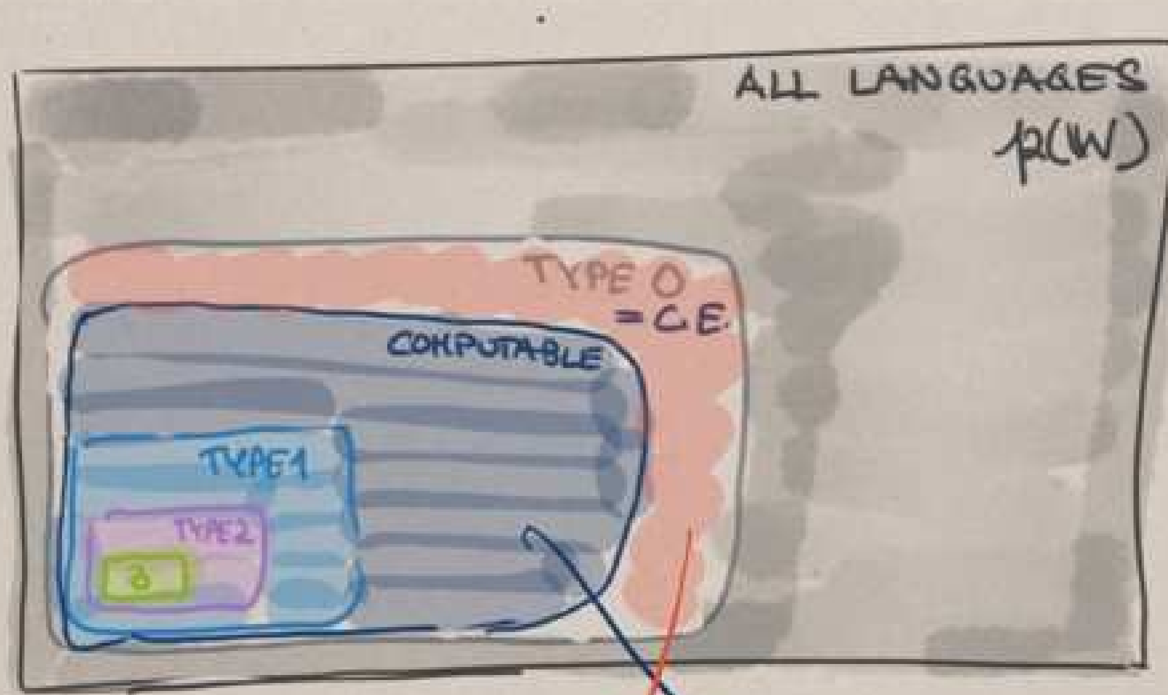
? Type 0 $\longleftrightarrow$ $\dfrac{\text{COMPUTABLE}}{\text{COMPUTABLY ENUMERABLE}}$

Also: The formulation of DECISION PROBLEMS relies on the notion of

ALGORITHM

which is still not properly defined!

RECAP



LECTURE $\overline{\underline{XX}}$: We finally proved that

c.e. $\neq$ computable

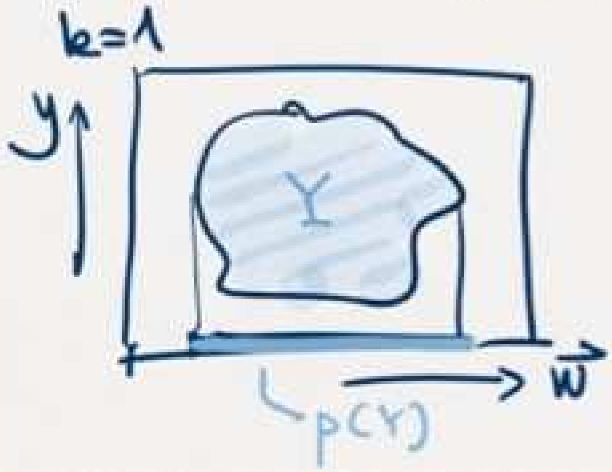
The halting problem $\mathbb{K}$ (or $\mathbb{K}_0$) is computably enumerable, but not computable.

Goal : Understand c.e. sets better [and argue that c.e. = type 0]

From Lecture $\overline{XX}$:

Def. $X \subseteq \mathbb{N}^k$ is called Σ_1 if there is a computable $Y \subseteq \mathbb{N}^{k+1}$ s.t.

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



We say

$$X = p(Y) = \{\vec{w} \, ; \, \exists y \; (\vec{w}, y) \in Y\}$$

is the projection of Y.

We say a set X is Π_1 if it is the complement of a Σ_1 set.

We say X is Δ_1 if it is both Σ_1 & Π_1.

Motivation for the names

ANALOGY

Σ_1 since it is written with existential quantifier — SUMS

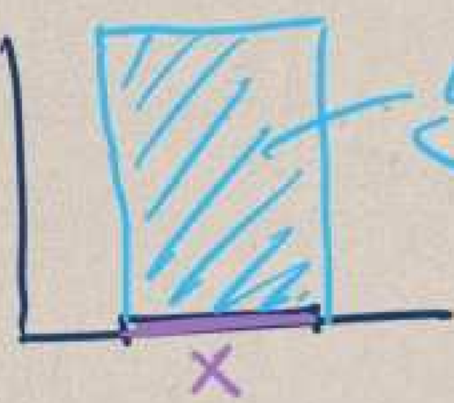
Π_1 --- universal (ES#4) — PRODUCTS

Δ_1 German DURCHSCHNITT (intersection)

Prop 4.30 Every computable set is Δ_1.

Proof. By closure under complement, it's enough to show Σ_1.

Given X, write
$$Y := \{(\vec{w}, y) ; \vec{w} \in X\}$$



cylindre over $X = Y$

Clearly $p(Y) = X$.

q.e.d.

Theorem 4.31 X is c.e. $\Longleftrightarrow$ X is Σ_1.

Proof. "$\Longrightarrow$". If X is c.e., then ψ_X is comp.
So there is M s.t.
$$\psi_X = f_{M,k}$$
$$Y := \{(\vec{w}, y) ; t_{M,k}(\vec{w}, y) = a\}$$
By L 4.24, Y is computable.
$$\vec{w} \in X \Longleftrightarrow \psi_X(\vec{w})\downarrow \Longleftrightarrow \exists y \, (\vec{w}, y) \in Y.$$

"$\Longleftarrow$" Let X be Σ_1. So let Y be computable s.t. $X = p(Y)$.

We know that χ_Y is computable.

Apply minimisation to χ_Y:

$h(\vec{w})$ = the minimal y s.t. $(\vec{w}, y) \in Y$.

Note $\mathrm{dom}(h) = p(Y) = X$.

So X is the domain of a partial comp. function, thus c.e.

q.e.d.

APPLICATION Suppose

$f : \mathbb{W}^2 \dashrightarrow \mathbb{W}$ partial computable

Then $\{w ; \exists v \, f(w,v)\downarrow\}$ is c.e.

Naïve idea Let $Y := \{(w,v) ; f(w,v)\downarrow\}$. But that's not in general computable as $\mathbb{K}_0$ shows.

Let M be s.t. $f = f_{M,2}$.

Let

this is a subset of $\mathbb{W}^3$, not $\mathbb{W}^2$!

$$Z := \{(w, v_0, v_1) ; t_{M,2}(w, v_0, v_1) = a\}$$

Clearly computable by L 4.24.

MERGING & SPLITTING TO THE RESCUE!

Merging & Splitting allows us to view $\mathbb{W}^2$ as $\mathbb{W}$, and thus $\mathbb{W}^3$ as $\mathbb{W}^2$.

$$Y := \{(w, u) ; (w, u_{(0)}, u_{(1)}) \in Z\}$$

By the section on splitting,

$u \longmapsto u_{(0)}$ are computable

$u \longmapsto u_{(1)}$

& thus Y is computable.

$$\exists v\, f(w,v)\downarrow \iff \exists v_0 \exists v_1\ (w, v_0, v_1) \in Z$$
$$\iff \exists u\ (w, u_{(0)}, u_{(1)}) \in Z$$
$$\iff (w, u) \in Y \iff w \in p(Y).$$

So the set

$$\{w ; \exists v f(w,v)\downarrow\}$$ is c.e.

ZIGZAG TECHNIQUE

The previous argument is sometimes known as a zigzag argument.

You can visualize the $\exists v \iff \exists v_0 \exists v_1$ transformation as searching through $\mathbb{N}^2$ in zigzag fashion.

	ε	$\mathbf{0}$	$\mathbf{1}$	$\mathbf{00}$	$\mathbf{10}$	$\mathbf{01}$	$\mathbf{11}$	$\mathbf{000}$	$\cdots$
ε	$t_{M,k}(\vec{w},\varepsilon,\varepsilon) \to$	$t_{M,k}(\vec{w},\varepsilon,\mathbf{0})$	$t_{M,k}(\vec{w},\varepsilon,\mathbf{1})$	$t_{M,k}(\vec{w},\varepsilon,\mathbf{00})$	$t_{M,k}(\vec{w},\varepsilon,\mathbf{10})$	$\cdots$			
$\mathbf{0}$	$t_{M,k}(\vec{w},\mathbf{0},\varepsilon)$	$t_{M,k}(\vec{w},\mathbf{0},\mathbf{0})$	$t_{M,k}(\vec{w},\mathbf{0},\mathbf{1})$	$t_{M,k}(\vec{w},\mathbf{0},\mathbf{00})$	$t_{M,k}(\vec{w},\mathbf{0},\mathbf{10})$	$\cdots$			
$\mathbf{1}$	$t_{M,k}(\vec{w},\mathbf{1},\varepsilon)$	$t_{M,k}(\vec{w},\mathbf{1},\mathbf{0})$	$t_{M,k}(\vec{w},\mathbf{1},\mathbf{1})$	$t_{M,k}(\vec{w},\mathbf{1},\mathbf{00})$	$t_{M,k}(\vec{w},\mathbf{1},\mathbf{10})$	$\cdots$			
$\mathbf{00}$	$t_{M,k}(\vec{w},\mathbf{00},\varepsilon)$	$t_{M,k}(\vec{w},\mathbf{00},\mathbf{0})$	$t_{M,k}(\vec{w},\mathbf{00},\mathbf{1})$	$t_{M,k}(\vec{w},\mathbf{00},\mathbf{00})$	$\ddots$				
$\mathbf{10}$	$t_{M,k}(\vec{w},\mathbf{10},\varepsilon)$	$t_{M,k}(\vec{w},\mathbf{10},\mathbf{0})$	$t_{M,k}(\vec{w},\mathbf{10},\mathbf{1})$	$\ddots$					
$\vdots$	$\vdots$	$\vdots$	$\vdots$						

This would not work as a SERIAL computation: if you first try to run the first row, it might never halt! So these ∞ly many rows need to be processed in PARALLEL.

In other words: we can do ∞ many computations in parallel.

Corollary 4.33 X is computable $\iff$ X is Δ_1

Proof. "$\Longrightarrow$" Prop. 4.30.

"$\Longleftarrow$" uses zigzag:

Since X is Δ_1, we know that there are machines M, M' s.t.

$$\vec{w} \in X \iff \exists v\ t_{M,k}(\vec{w}, v) = a$$
$$\vec{w} \notin X \iff \exists v\ t_{M',k}(\vec{w}, v) = a$$

$$f(\vec{w}, v) := \begin{cases} t_{M,k}(\vec{w}, v_{(1)}) & \text{if } \# v_{(0)} \text{ is even} \\ t_{M',k}(\vec{w}, v_{(1)}) & \text{if } \# v_{(0)} \text{ is odd} \end{cases}$$

Apply minimisation to f, obtain h

$h(\vec{w}) :=$ the least v s.t. $f(\vec{w}, v) \neq \varepsilon$

Output a if $\# h(\vec{w})_{(0)}$ is even

ε if $\# h(\vec{w})_{(0)}$ is odd

q.e.d.

Corollary 4.34 Σ_1 is not closed under complementation.

[$\mathbb{W} \setminus \mathbb{K}$ is Π_1 and not Δ_1, so not Σ_1]

!! CORRECTED AFTER THE LECTURE

Theorem Every type 0 language is c.e.

Proof. Fix $G = (\Sigma, V, P, S)$

Let $\Sigma' = \Omega \cup \{\Rightarrow\}$.

Encode derivation as

$$\sigma_0 \Rightarrow \sigma_1 \Rightarrow \dots \Rightarrow \sigma_n$$ this is a Σ'-word.

Say $w \in (\Sigma')^*$ is a derivation code if

· $w = \sigma_0 \Rightarrow \dots \Rightarrow \sigma_n$

with $(\sigma_0, \dots, \sigma_n)$ a G-derivation.

In this case, we call σ_0 the initial string

σ_n the final string.

$$Y := \{(w, v) ;\ v \text{ is a derivation code with initial string } S \text{ \& final string } w\}$$

Clearly, Y is computable.

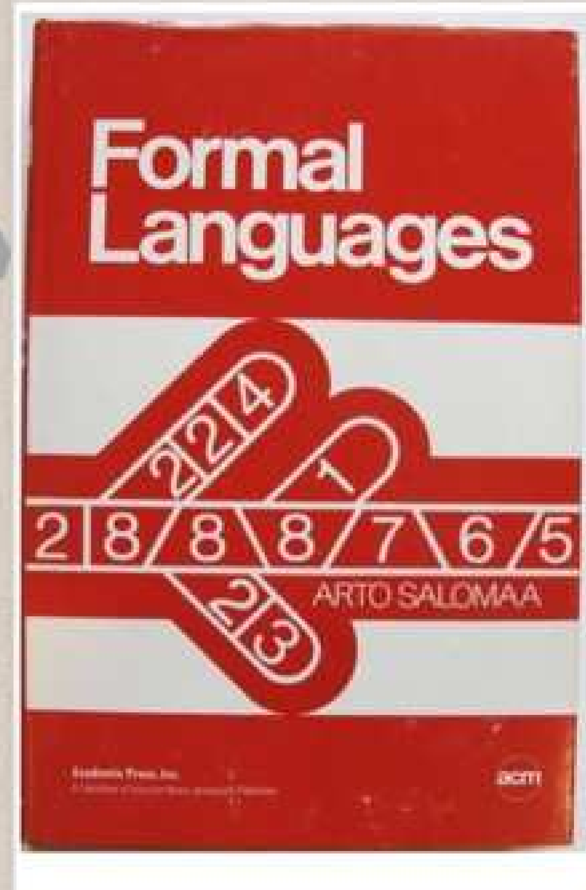
But

$$w \in \mathcal{L}(G) \iff \underbrace{\exists v\, (w, v) \in Y}_{\Sigma_1,\ \text{i.e. c.e.}}$$

q.e.d.

Converse also holds:

$\boxed{\text{Every c.e. set } X \subseteq W \text{ is a type 0 language.}}$



Arto Salomaa (born 1934)

Proof in Salomaa, Formal Languages Theorem 4.4 (p. 37)

Proof sketch in § 4.10.
[Salomaa proves this via Turing machines, so we'll see a sketch after having introduced them.]

---

§ 4.9 Closure properties

Prop. 4.38 The computable set are closed under all five operations.

Proof. Let A, B computable, so therefore χ_A, χ_B are computable functions.

get

$$\chi_{A\cap B}(\vec{w}) = \begin{cases} a & \text{if } \chi_A(\vec{w}) = a = \chi_B(\vec{w}) \\ \varepsilon & \text{o/w} \end{cases}$$

$$\chi_{A\cup B}(\vec{w}) = \begin{cases} \varepsilon & \text{if } \chi_A(\vec{w}) = \varepsilon = \chi_B(\vec{w}) \\ a & \text{o/w} \end{cases}$$

$$\chi_{\mathbb{W}\setminus A}(\vec{w}) = \begin{cases} a & \text{if } \chi_A(\vec{w}) = \varepsilon \\ \varepsilon & \text{o/w} \end{cases}$$

All obviously computable.

[This is essentially the idea of a product machine.]

<u>Homework</u> Try to make this idea of a "product machine" precise: what properties does a product operation need to have to allow the conclusion that the class is closed under all Boolean operations?

<u>Concatenation</u> AB is computable.

$[A, B \subseteq \mathbb{W}]$

Given w, check all possible $w = vu$ for v initial segment of w.

How many? $|w|$ many initial segments.

For each such, check $\chi_A(v) = a = \chi_B(u)$.

If so, output a;

if none of them work, output ε.

q.e.d.