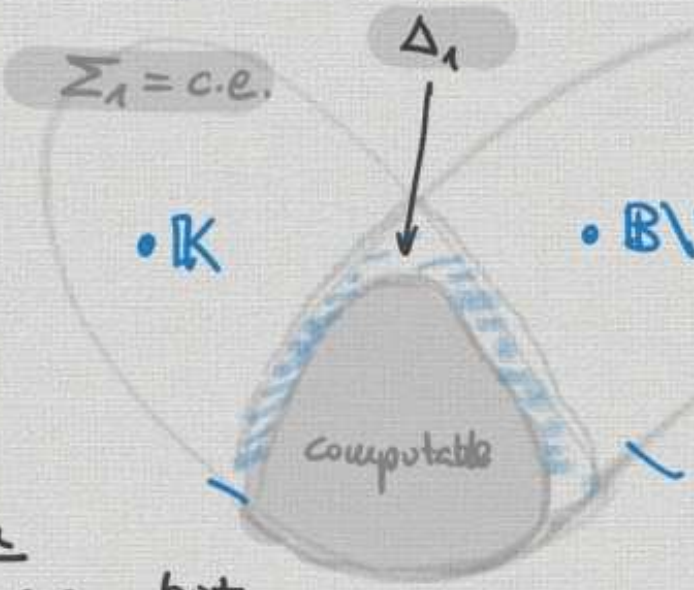
AUTOMATA AND FORMAL LANGUAGES

XX

TWENTIETH LECTURE 30 November 2024



Theorem
$\mathbb{K}$ is c.e., but not computable

Theorem TFAE
(i) X is c.e.
(ii) X is Σ_1

Characterization of c.e. sets
$X \subseteq \mathbb{B}$; $X \neq \emptyset$ TFAE
(i) X is c.e.
(ii) $X = \mathrm{dom}(f)$; f computable
(iii) X is Σ_1
(iv) $X = \mathrm{ran}(g)$; g computable

Remaining questions
1. Where precisely is $\mathbb{K}$ in the above picture?
2. What's the relationship between Δ_1 and computable sets?
3. Where do Type 0 sets fit in?

NOTE: Blackboard proof of P 4.35 had an error. Handwritten notes on Moodle corrected.

Prop 4.36 TFAE (i) X is computable
(ii) X is Δ_1

Pf. (i) $\Rightarrow$ (ii) is P 4.32.

(ii) $\Rightarrow$ (i). If X is Δ_1, then both X and $\mathbb{B}^k \setminus X$ are Σ_1.

Let M, M' be RM s.t.

$$\vec{w} \in X \iff \exists v \ (\vec{w}, v) \in T_M$$
$$\vec{w} \notin X \iff \exists v \ (\vec{w}, v) \in T_{M'}$$

} computable

Define $Y := \{(\vec{w}, v)$; $\#v_{(0)}$ is even and $(\vec{w}, v_{(0)}) \in T_M$ OR $\#v_{(0)}$ is odd and $(\vec{w}, v_{(1)}) \in T_{M'}\}$

Y computable

Use minimisation to get that

$$h(\vec{w}) := \begin{cases} \text{the least } v \text{ s.t. } (\vec{w}, v) \in Y \\ \uparrow \quad \text{if no such } v \text{ exists} \end{cases}$$

So, h is computable. By choice of M, M', h is a total function.

But then $\vec{w} \in X \iff \#h(\vec{w})_{(0)}$ is even. So X is computable.

Corollary The c.e. sets are not closed under complementation. In particular, $\overline{K}$ is not c.e. q.e.d.

Corollary 4.38 Every type 0 language is c.e.

Proof If $G = (\{0,1\}, V, P, S)$ is any grammar, consider $\Omega := \{0,1\} \cup V$.

The set $\mathcal{D} := \{(w,v)_{v \in \mathbb{B}} : w \in T$ and w codes a G-derivation starting from S and ending in $v\}$ is computable.

Then $T := (\Omega^* / \!\!\upharpoonright)^* \Omega^*$ is the set of "putative derivations".

Then $v \in \mathcal{L}(G) \iff \exists w \, ((w,v) \in \mathcal{D})$

This is Σ_1, so c.e. q.e.d.

Thm (without proof) The converse of C 4.38 also holds:

i.e., X is c.e. iff X is Type 0.

§4.9 Closure properties

	concatenation	union	intersection	complement	difference
regular (type 3)	✓	✓	✓	✓	✓
context-free (type 2)	✓	✓	×	×	×
noncontracting (type 1)	✓	✓	✓	✓	✓
computable	✓	✓	✓	✓	✓
computably enumerable (type 0)	✓	✓	✓	×	×

Figure 6: The closure properties of all classes of languages we discussed in an overview.

Corollary just proved

Prop. 4.39 The computable sets are closed under union and intersection.

Concatenation

Typed notes pp. 69–70.

Pf: $\chi_{A\cap B}(w) = \begin{cases} 1 & \text{if } \chi_A(w) = 1 = \chi_B(w) \\ 0 & \text{o/w} \end{cases}$

$\chi_{A\cup B}(w) = \begin{cases} 0 & \text{if } \chi_A(w) = 0 = \chi_B(w) \\ 1 & \text{o/w} \end{cases}$

q.e.d.

Prop 4.40 The c.e. sets are closed under union and intersection.

Pf: $\psi_{A\cap B}(w) = \begin{cases} 1 & \text{if } \psi_A(w) = 1 = \psi_B(w) \\ \uparrow & \text{o/w.} \end{cases}$

For union: zigzag argument — if $w\in A \iff \exists v\,(w,v)\in C$
$w\in B \iff \exists v\,(w,v)\in D$

Then $w\in A\cup B \iff \exists\, v$ s.t.

$\#v_{(0)}$ is even and $(w, v_{(1)})\in C$
OR $\#v_{(0)}$ is odd and $(w, v_{(1)})\in D$

q.e.d.

§ 4.10 The Church-Turing Thesis

Alonzo Church

Alonzo Church (1903–1995)

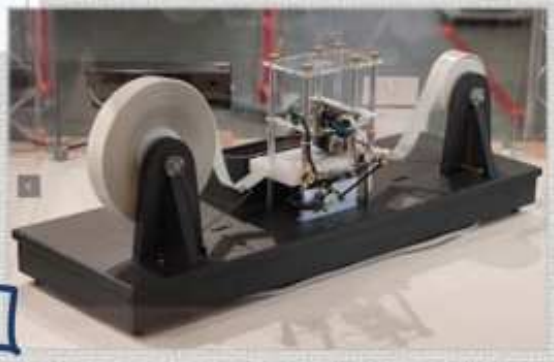
Born	June 14, 1903 Washington, D.C., US
Died	August 11, 1995 (aged 92) Hudson, Ohio, US
Citizenship	United States
Alma mater	Princeton University
Known for	Lambda calculus Simply typed lambda calculus Church encoding Church's theorem Church–Kleene ordinal Church–Turing thesis Frege–Church ontology Church–Rosser theorem Intensional logic

Alan Turing

OBE FRS

Turing c. 1928 at age 16

Born	Alan Mathison Turing 23 June 1912 Maida Vale, London, England
Died	7 June 1954 (aged 41) Wilmslow, Cheshire, England



Thm If $f: \mathbb{B}^n \dashrightarrow \mathbb{B}$, then TFAE:

(i) f is computable [in the sense of AM]

(ii) f is recursive

(iii) f can be performed by a Turing machine.

The Church-Turing Thesis. The mentioned equivalent formal concepts of computability describe the informal notion of computability successfully: any reasonable attempt to describe the informal notion of computability will lead to a formal notion that is equivalent to the ones we have described.

With CTT, we now know what it means to be "algorithmically solvable"!

Take $v \longmapsto G_v$ an encoding of grammars s.t. $\{G_v \,;\, v \in \mathbb{B}^*\}$ is the set of all grammars.

$$WP := \{(w,v) \,;\, w \in \mathcal{L}(G_v)\}$$

$$\not\equiv P := \{w \,;\, \mathcal{L}(G_v) = \emptyset\}$$

$$\not\equiv qP := \{(w,v) \,;\, \mathcal{L}(G_v) = \mathcal{L}(G_w)\}$$

These problems are solvable if those sets are computable.

Application WP is not computable [Use Thm. c.e $\neq$ Type 0.]

If WP comp; then $f(w) := \begin{cases} \uparrow & \text{if } (w,w) \in WP \\ 0 & \text{if } (w,w) \notin WP \end{cases}$ is computable.

Fixed d s.t. $dom(f) = \mathcal{L}(G_d)$.

$d \in \mathcal{L}(G_d) \Leftrightarrow d \in dom(f) \Leftrightarrow (d,d) \notin WP \Leftrightarrow d \notin \mathcal{L}(G_d)$. Contradiction.

q.e.d