XXII

TWENTY-FIRST LECTURE

AUTOMATA & FORMAL LANGUAGES

24 November 2023

THE GRAND PICTURE

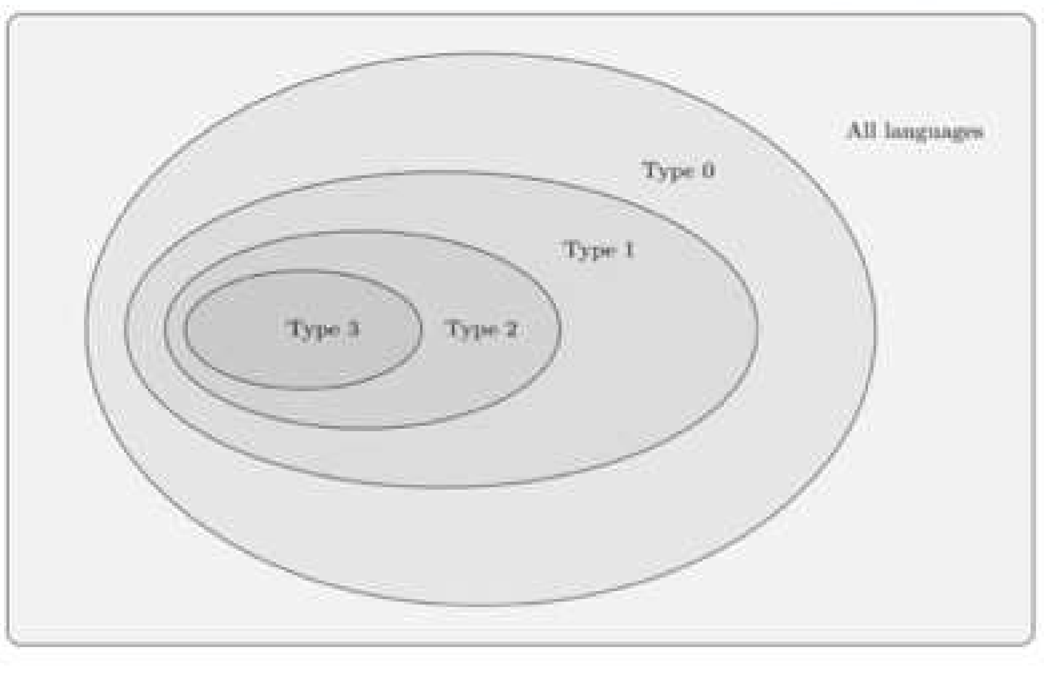

Figure 1: The Chomsky hierarchy

Q1: Separation of classes of languages

Q2: Closure properties

Q3: Decision problems

Underneath Q3: what does it mean to give a negative answer to a decision problem?

Separation of classes:

Figure 1: The Chomsky hierarchy

LECTURE IV page 3

Chomsky defined:

L is type 0 if there is G s.t. $L = \mathcal{L}(G)$.

L is type 1 if it is context-sensitive

— 2 — context-free

— 3 — regular

Observe ① Noncontracting languages are missing. Why?

② Question: Is the diagram PROPER, i.e., is each area in the picture populated by example.

① The reason for this is

THEOREM (Chomsky)

L is noncontracting $\iff$ L is context-sensitive.

Cf. Example Sheet #1.

② Properness of a Venn diagram:

find a language that is not type 0 *

find language type 0, but not type 1 *

find language type 1 not type 2 *

find language type 2 not type 3 *

To prove results like this, we need tools to prove that something cannot be done by a particular type of grammar.

*: follows from the fact that there are unctbly many languages, but only ctbly many type 0 languages.

* Not type 0.

Since $\mathbb{K}$ is not Δ_1, it is not Π_1, so $\mathbb{N} \setminus \mathbb{K}$ is not Σ_1, i.e., not c.e.

* Type 0 but not type 1

Note that type 1 $\Longrightarrow$ computable.

So $\mathbb{K}$ is c.e., but not computable, so type 0, not type 1.

* Type 1 but not type 2.

$\{a^n b^n c^n ; n > 1\}$ Proof by (CF)PL.

* Type 2 but not type 3.

$\{a^n b^n ; n > 1\}$ Proof by (R)PL.

Note that type 0 = c.e.

Closure properties & decision problems:

Lecture XIII, page 12

It turns out that the Equivalence Problem for c-f grammar is unsolvable. This result will not be proved in this course.

DECISION PROBLEMS

	Type 0	Type 1	Type 2	Type 3
WORD P.	?	✓	✓	✓
EMPTINESS P.	?	?	~~?~~ ✓	~~?~~ ✓
EQUIVALENCE P.	?	?	~~?~~ ✗	~~?~~ ✓

CLOSURE PROPERTIES

Proposition on page 7

	Type 0	Type 1	Type 2	Type 3
Concatenation	✓	✓	✓	~~✗~~ ✓
Union	✓	✓	✓	~~✗~~ ✓
Intersection	?	?	~~?~~ ✗	~~?~~ ✓
Complement	?	?	~~?~~ ✗	~~?~~ ✓
Kleene plus	?	?	~~?~~ ✓	✓

Focusing now on the three type 0 question marks.

§ 4.9 Closure properties

4.9 Closure properties

Proposition 4.42. The class of computable languages is closed under union, intersection, complement, and concatenation.

Proof. Let A and B be computable sets, i.e., χ_A and χ_B are computable functions. Then

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

$$\chi_{A\cup B}(w) = \begin{cases} \varepsilon & \text{if } \chi_A(w) = \varepsilon = \chi_B(w) \\ a & \text{otherwise, and} \end{cases}$$

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

are computable functions, and so $A \cap B$, $A \cup B$, and $\mathbb{W}\setminus A$ are computable sets. Also, the

Very easy: closed under union, intersection, and complement.

<u>Concatenation</u> Assume A, B are computable. Show that AB is computable:

given w, check the following $|w|+1$ subwords

$w \restriction k$ where $0 \leq k \leq |w|$

Check whether $w \restriction k \in A$; then determine u s.t. $(w \restriction k)u = w$ and check whether $u \in B$.

If both are "YES" $\longrightarrow$ YES.

o/w go to $k \longmapsto k+1$.

If none of the k's said "YES" $\longrightarrow$ NO.

Proposition 4.43. The computably enumerable languages are closed under union, intersection, and concatenation, but not under complementation and difference.

already proved last time.

Proof. The construction for intersection from the proof of Proposition 4.42 works for pseudo-characteristic functions as well:

$$\psi_{A\cap B}(w) = \begin{cases} a & \text{if } \psi_A(w) = a = \psi_B(w) \\ \uparrow & \text{otherwise.} \end{cases}$$

Intersection same proof as for computable.

This does not work for union.

Here need to use ZIGZAG METHOD:

$$* \quad \boxed{\begin{aligned} w\in A &\iff \exists v\ (w,v)\in C \\ w\in B &\iff \exists v\ (w,v)\in D \end{aligned}} \Bigg\} \ C, D \text{ computable}$$

$$Y := \{(w,v);\ \#v_{(0)} \text{ is even and } (w,v_{(1)})\in C \text{ or } \#v_{(0)} \text{ is odd and } (w,v_{(1)})\in D\}$$

Y is computable &

$$w\in A\cup B \iff \exists v\ (w,v)\in Y.$$

Finishes the union argument.

CONCATENATION:

given w and v, split w as follows:

$$w = \underset{\underset{w\upharpoonright\#v}{\parallel}}{w_0}\ w_1 \leftarrow \text{Rest}$$

initial segment $\rightarrow I(w,v)\ F(w,v) \leftarrow$ final segment

Use the notation of *.

$$w\in AB \iff \exists v_0\exists v_1\exists v_2\ \big((I(w,v_0),v_1)\in C\ \&\ (F(w,v_0),v_2)\in D\big)$$

By a three quantifier version of zigzag, this is Σ_1.

SUMMARY:

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

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

For Type 1, closure properties were an open problem until 1987. Then proved by Immerman & Szelepcsényi.

(Reference to the papers in the typed notes.)

Decision Problems

How do we show that a decision problem has a negative solution?

[i.e., what is an algorithm?]

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

1. Turing 1936 showed unsolvability of ENTSCHEIDUNSPROBLEM

   So: he had an idea what unsolvability means.

2. Church had proved this earlier, so he also had an idea.

3. Turing travelled to Princeton and they found out that their ideas were mathematically equivalent.

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

→ ROBUST MATHEMATICAL NOTION.

Some models of computation:

1. Register machines.
   → computable functions & sets

2. Recursive functions
   [Church's definition of "algorithm"]

3. Turing machines
   TM have a single tape, a READ/WRITE head that sits on the tape & can only read or write in one cell at a time and then move left or right.

4. WHILE programming languages: just variable assignments, basic operations & WHILE loops

→ some informal in the typed notes

Theorem 4.44. If $f : \mathbb{W}^k \dashrightarrow \mathbb{W}$, then the following are equivalent:

(i) the partial function f is computable,

(ii) the partial function f is recursive,

(iii) the partial function f is Turing computable, and

(iv) the partial function f is while computable.

analogue of "computable" for WHILE languages

analogue of "computable" for TM

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.

<u>Definition</u> A decision problem $A \subseteq \mathbb{W}^k$ is <u>SOLVABLE</u> if and only if it is computable.

Theorem The word problem for type 0 languages is unsolvable.

Proof. Take some encoding of grammars into words s.t.

$$w \longmapsto G_w$$

is a surjection from $\mathbb{W}$ onto the set of all grammars (up to isomorphism).

Then $W := \{(u,v) \, ; \, u \in \mathcal{L}(G_v)\}$

is the WORD PROBLEM.

Let's set
$$f(w) := \begin{cases} \uparrow & \text{if } w \in \mathcal{L}(G_w) \\ \varepsilon & \text{if } w \notin \mathcal{L}(G_w) \end{cases}$$

If W is computable (= solvable), then f is computable.

Thus $\mathrm{dom}(f)$ is c.e., so find $d \in \mathbb{W}$ s.t.

$$\mathrm{dom}(f) = \mathcal{L}(G_d)$$ [Possible since Type 0 = c.e.]

$$d \in \mathcal{L}(G_d) \Longleftrightarrow d \in \mathrm{dom}(f) \Longleftrightarrow d \notin \mathcal{L}(G_d)$$

Contradiction!

q.e.d.

Left for Lectures XXIII & XXIV: Emptiness Problem & Equivalence Problem for Type 0 languages.