XIV

Fourteenth Lecture

AUTOMATA & FORMAL LANGUAGES

6 November 2023

Classes of languages corresponding to models of computation.

EXAMPLE SHEET #1

(10) For each of the following languages $L \subseteq \{0,1\}^*$, determine whether or not they are regular. Justify your answers.

(a) $\{0^n 1^{2n} ; n \geq 1\}$;
(b) $\{ww ; \varepsilon \neq w \in \{0,1\}^*\}$;
(c) $\{w1w ; w \in \{0\}^*\}$;
(d) $\{v1w ; v, w \in \{0\}^*\}$;
(e) $\{0^n 1^m ; n > m\}$;
(f) $\{0^n 1^m ; n \neq m\}$;
(g) $\{0^n 1^m ; n \geq m \text{ and } m \leq 1000\}$;
(h) $\{0^n 1^m ; n \geq m \text{ and } m \geq 1000\}$;
(i) $\{1^p ; p \text{ is a prime}\}$.

Minimal pair: one of these is regular, the other isn't. Good example to understand the powers & limitations of automata.

Automata cannot store information that is potential unbounded.

EXAMPLE SHEET #2

(27) For each of the following languages, decide whether it is context-free and provide an argument for your claim:

(a) $\{a^n b^m ; n \neq m, n + m > 0\}$;
(b) $\{a^m b^n c^m d^n ; m, n \geq 1\}$;
(c) $\{a^n b^m c^k d^\ell ; 2n = 3m \text{ and } 5k = 7\ell\}$;
(d) $\{a^n b^m c^k d^\ell ; 2n = 3k \text{ and } 5m = 7\ell\}$;
(e) $\{a^n b^m c^k d^\ell ; 2n = 3k \text{ or } 5m = 7\ell\}$;
(f) $\{a^p ; p \text{ is prime}\}$;
(g) $\{a^{2^n} ; n \geq 1\}$;
(h) $\{ww ; w \in \{a,b\}^+\}$;
(i) $\{a,b\}^+ \setminus \{ww ; w \in \{a,b\}^+\}$.

Context-free languages can store information, but only one piece of information.

Languages	Computation Model	Properties
Type 3 Regular	Automata	Only bounded memory
Type 2 Context-free	NOT COVERED IN THIS COURSE: pushdown automata	Can store a single number
Type 1 Context-sensitive	NOT COVERED IN THIS COURSE	
Type 0	?	

Goal for the next 2-3 lectures:
We are going to introduce REGISTER MACHINES

Models of Computation:

Register machines

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

"Father of computing"

TURING machines

230 A. M. TURING [Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

John von Neumann

von Neumann in the 1940s

Member of the	United States Atomic Energy Commission
In office	March 15, 1955 – February 8, 1957
President	Dwight D. Eisenhower
Preceded by	Eugene M. Zuckert
Succeeded by	John S. Graham
Personal details	
Born	Neumann János Lajos December 28, 1903 Budapest, Kingdom of Hungary, Austria-Hungary
Died	February 8, 1957 (aged 53) Washington, D.C., U.S.
Resting place	Princeton Cemetery
Citizenship	Hungary United States

VON NEUMANN ARCHITECTURE

→ modern computers

Storage units

Register machines

Some associated names:

HAO WANG 1921-1995

MARVIN MINSKY 1927-2016

JOACHIM LAMBEK 1922-2014

JOHN SHEPHERDSON 1926-2015

§ 4.1 REGISTER MACHINES

4.1 Register machines

Let Σ be an alphabet and Q a non-empty finite set whose elements we shall call *states*. A tuple of the form

$$(0, k, a, q) \in \mathbb{N} \times \mathbb{N} \times \Sigma \times Q,$$
$$(1, k, a, q, q') \in \mathbb{N} \times \mathbb{N} \times \Sigma \times Q \times Q,$$
$$(2, k, q, q') \in \mathbb{N} \times \mathbb{N} \times Q \times Q \text{ or}$$
$$(3, k, q, q') \in \mathbb{N} \times \mathbb{N} \times Q \times Q$$

is called a (Σ, Q)-*instruction*. For improved readability, we write

$$+(k, a, q) := (0, k, a, q), \qquad (\text{"add"})$$
$$?(k, a, q, q') := (1, k, a, q, q'), \qquad (\text{"check"})$$
$$?(k, \varepsilon, q, q') := (2, k, q, q') \text{ and} \qquad (\text{"check"})$$
$$-(k, q, q') := (3, k, q, q') \qquad (\text{"remove"})$$

This is all psychological blah-blah!

Instruction	Interpretation
$+(k, a, q)$	"Add the letter a to the content of register k and go to state q."
$?(k, a, q, q')$	"Check whether the last letter in register k is a; if so, go to state q; otherwise, go to state q'."
$?(k, \varepsilon, q, q')$	"Check whether register k is empty; if so, go to state q; otherwise, go to state q'."
$-(k, q, q')$	"Check whether register k is empty; if so, go to state q; otherwise, remove the final letter of its content and go to state q'".

Need proper mathematical definition!

Table 1: Interpretations of register machine instructions.

Definition $M := (\Sigma, Q, P)$ is called a **Σ-register machine** if

(1) Q is a nonempty finite set (elts called **states**)

(2) $q_S \neq q_H \in Q$

q_S ↑ START STATE; q_H ← HALT STATE

(3) P is a function $\mathrm{dom}(P) = Q$ and $P(q)$ is (Σ, Q)-instruction.

the program

If $q \in Q$ and

$$\boxed{q \longmapsto P(q)}$$

then this is a **program line**

Note Q is finite, therefore P is finite and there is a maximal k s.t. register k is used in any instruction in P. → **UPPER REGISTER INDEX of M**

Example $Q = \{q_S, q_H, q_0, q_1, q_2, q_3, q_4, q_5\}$

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

$q_S : +(2, 0, q_0)$

$q_H : ?(0, \varepsilon, q_H, q_S)$

$q_0 : +(1, 1, q_1)$

$q_1 : ?(2, 1, q_S, q_4)$

$q_2 : ?(1, 0, q_S, q_0)$

$q_3 : +(2, 0, q_H)$

$q_4 : -(1, q_3, q_H)$

$q_5 : ?(2, \varepsilon, q_S, q_H)$

Food for thought

Try to find out what this machine does and on which inputs it halts.

Starts in q_S, adds 0 to register 2, go to q_0.

Adds 1 to register 1, go to q_1.

Check whether last digit of reg 2 is 1 (is it?) ... ?

This example has UPPER REGISTER INDEX 2.

CONFIGURATIONS

A tople $C := (q, w_0, \dots, w_n)$ is called a

CONFIGURATION / SNAPSHOT of length $n+1$

if $q \in Q$, $w_i \in W$.

Write $\vec{w}$ for $w_0, \dots, w_n$

Called the REGISTER CONTENT of C.

We're going to define an operation of M with upper register index n on configuration of length $\geq n+1$.

Intuition If $C = (q, \vec{w})$ and the instruction of P at q is $P(q)$, we want the RM to perform what we described in word in Table 1.

Definition M transforms C to C' if

Case 1. If $P(q) = +(k, a, q')$ and $C' = (q', w_0, ..., w_{k-1}, w_k a, w_{k+1}, ..., w_m)$.

Case 2. If $P(q) = ?(k, a, q', q'')$,

Subcase 2a. $w_k = wa$ for some w and $C' = (q', w_0, ..., w_m)$ or

Subcase 2b. $w_k \neq wa$ for any w and $C' = (q'', w_0, ..., w_m)$.

Case 3. If $P(q) = ?(k, \varepsilon, q', q'')$,

Subcase 3a. $w_k = \varepsilon$ and $C' = (q', w_0, ..., w_m)$ or

Subcase 3b. $w_k \neq \varepsilon$ and $C' = (q'', w_0, ..., w_m)$.

Case 4. If $P(q) = -(k, q', q'')$,

Subcase 4a. $w_k = \varepsilon$ and $C' = (q', w_0, ..., w_m)$ or

Subcase 4b. $w_k = wa$ for some a and $C' = (q'', w_0, ..., w_{k-1}, w, w_{k+1}, ..., w_m)$.

This is precisely the mathematical formulation of the "psychological blah-blah":

Instruction	Interpretation
$+(k, a, q)$	"Add the letter a to the content of register k and go to state q."
$?(k, a, q, q')$	"Check whether the last letter in register k is a; if so, go to state q; otherwise, go to state q'."
$?(k, \varepsilon, q, q')$	"Check whether register k is empty; if so, go to state q; otherwise, go to state q'."
$-(k, q, q')$	"Check whether register k is empty; if so, go to state q; otherwise, remove the final letter of its content and go to state q''."

Table 1: Interpretations of register machine instructions.

Computation Sequence

We define by recursion for RM M and a vector of word $\vec{w}$ a sequence of configurations called the COMPUTATION SEQUENCE of M with input $\vec{w}$

$$C(0, M, \vec{w}) := (q_S, \vec{w}).$$

$C(k+1, M, \vec{w}) = C$ iff M transforms $C(k, M, \vec{w})$ to C.

Note This sequence is always defined for all k, so this is an infinitely long computation.

Definition M halts on input $\vec{w}$ at time k if k is least s.t.

$$C(k, M, \vec{w}) = (q_H, \vec{v}) \text{ for some } \vec{v}.$$

$\vec{v}$: REGISTER CONTENT AT TIME OF HALTING

Definition M, N are strongly equivalent if

$$\forall k, \vec{w} \quad C(k, M, \vec{w}) = (q, \vec{v}) \;\&\; C(k, N, \vec{w}) = (q', \vec{v}') \Longrightarrow \vec{v} = \vec{v}'.$$

Observation In analogy to Prop 1.14, if $|Q| = |Q'|$ and $M = (\Sigma, Q, P)$, there is P' s.t. $M' = (\Sigma, Q', P')$ is str. eq. to M.

(typed notes P 4.3)

Proposition For fixed Σ, there are only ctbly many RM up to strong eq.

Proof Fix n, k and Q. Define

$$RM(Q,k) := \{ M;\ M = (\Sigma, Q, P) \text{ \& upper reg. ind. } \leq k \}$$

$$RM(n,k) := \{ M;\ M = (\Sigma, Q, P) \text{ \& } |Q| = n \text{ \& upper r.i. } \leq k \}$$

By observation, every $M \in RM(Q,k)$ is str. eq. to some $M' \in RM(|Q|, k)$.

Thus EVERY RM is strongly eq.
to one in

$$\bigcup_{n,k\in\mathbb{N}} RM(n,k).$$

This is countable as a ctble union of finite sets. q.e.d.

Proposition (P 4.4)

PADDING LEMMA

For each M there are infinitely many str. eq. RM.

Proof. Add new states (as many as you like) to M without changing P:

$$M' = (\Sigma, Q', P') \quad \text{with } Q \subsetneq Q'$$

$$P' \restriction Q = P.$$

Then by ind., for all k, $C(k, M, \vec{w}) = C(k, M', \vec{w})$,
so M, M' are strongly eq.
There are infinitely many machines (at least one for every k with $|Q|+k+1$ states. q.e.d.