Turing Machines

A turing machines has an infinite tape (memory). There is a finite-state "program" that controls a tape head. The head can read, write and move around (in both directions) on the tape.

A typical program instructions: if the finite control is in state p and the head reads b, then write a, move the head to the left and go to state q

1 Formal Definition of TM

A Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

1. Q is the set of states
2. Σ is the input alphabet not containing the special **blank** symbol $\sqcup$
3. Γ is the tape alphabet satisfying $\Sigma \subset \Gamma$ and $\sqcup \in \Gamma$
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function (If in Q moving to Γ, replace the state and move the head to the **L**eft or **R**ight).This is deterministic
5. $q_0 \in Q$ is the start state
6. $q_{accept} \in Q$ is the accept state, and
7. $q_{reject} \in Q$ is the reject state $q_{accept} \neq q_{accept}$

Important: Blanks

The first $\sqcup$ denotes the start of the blanks

2 Computation of TM

The tape content is unbounded but always **finite**; the first (leftmost) blank marks the end of the tape content

A configuration consists of three items: the current state, the tape content and the head location

The configuration C_1 yields the configuration C_2 if the TM can legally go from C_1 to C_2 in a single step (using the transition function once)

The start configuration on an input $w \in \Sigma^*$ consists of the start state q_0, w as the tape content, and the head location being the first (leftmost) position of the tape

An accepting (rejecting) configuration is a configuration whose state is q_{accept} (q_{reject},respectively). Accepting and rejecting configurations are halting configurations.

A TM $\mathscr{M}$ accepts an input w if there is a sequence of configurations $C_1, C_2, ..., C_k$ such that

1. C_1 is the start configuration of $\mathscr{M}$ on input w
2. C_i yields C_{i+1} for $1 \leqslant i \leqslant k - 1$, and
3. C_k is an accepting configuration

The set of string accepted by $\mathscr{M}$ constitutes the language of $\mathscr{M}$, denoted by $L(\mathscr{M})$

2.1 Example

He did example 3.7 from theory of computation.

Here we describe a Turing machine (TM) M_2 that decides $A = \{0^{2^n} \mid n \geq 0\}$, the language consisting of all strings of 0s whose length is a power of 2.

M_2 = "On input string w:

1. Sweep left to right across the tape, crossing off every other 0.
2. If in stage 1 the tape contained a single 0, *accept*.
3. If in stage 1 the tape contained more than a single 0 and the number of 0s was odd, *reject*.
4. Return the head to the left-hand end of the tape.
5. Go to stage 1."

Each iteration of stage 1 cuts the number of 0s in half. As the machine sweeps across the tape in stage 1, it keeps track of whether the number of 0s seen is even or odd. If that number is odd and greater than 1, the original number of 0s in the input could not have been a power of 2. Therefore the machine rejects in this instance. However, if the number of 0s seen is 1, the original number must have been a power of 2. So in this case the machine accepts.

Now we give the formal description of $M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{\text{accept}}, q_{\text{reject}})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{\text{accept}}, q_{\text{reject}}\}$,
- $\Sigma = \{0\}$, and
- $\Gamma = \{0, x, \sqcup\}$.
- We describe δ with a state diagram (see Figure 3.8).
- The start, accept, and reject states are q_1, q_{accept}, and q_{reject}.

3 Turing-Recognisable and Turing-Decidable languages

Definition 1 - A language $\mathscr{L}$ is Turing-Recognisable, if there is a TM $\mathscr{M}$ that recognises it, i.e. $\mathscr{L} = L(\mathscr{M})$
Definition 2 - A language $\mathscr{L}$ is Turing-Decidable, if there is a TM $\mathscr{M}$ that accepts every $w \in \mathscr{L}$ and reject every $w \notin \mathscr{L}$

Important note 1: If $\mathscr{M}$ recognises $\mathscr{L}$, it may or may not halt on words not in $\mathscr{L}$. However, if $\mathscr{M}$ decides $\mathscr{L}$, it always halts
Important note 2: The standard terminology is "r.e." (stands for recursively enumerable) instead of "Turing-Recognisable" and "recursive" instead of "Turing-Decidable"

4 Multitape TM

A Multitape TM is an ordinary (single tape) TM with several tapes, each of them having its own head. The only difference in the formal definition is the transition function, which is now

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k$$

where k is the number of tapes.

Theorem. - Every Multitape TM has an equivalent single tape TM

Put the encoding from each turing machine in sequence, with a special character to act as a separator between each encoding so that the turing machine knows to jump.

Add $\dot{a}, \dot{b}$... for all the characters in the language, denote the position of the head in the original machine.

A single step in the multi tape TM will take many steps in a single tape TM, up to the total length of all the chunks added together.

5 Non-deterministic TM

A non-deterministic TM has a transition function

$$\delta : Q \times \Gamma \to \mathscr{P}(Q \times \Gamma \times \{L, R\})$$

This transition function creates the power set of all options, including the empty set.

Theorem - Every Non-deterministic TM has an equivalent deterministic TM

Proof. Idea: Consider the tree of all possible computations of the Non-deterministic TM. Start from the root (the start configuration) and do a breadth-first search. Accept only if an accepting configuration is found. Note that:

1. Depth First Search would not work - might go down forever
2. Can use a multitape TM to implement the breadth-first search

6 Enumerators

The concept of an "enumerator" is important as it justifies the term recursively enumerable.

A language $\mathscr{L}$ is enumerated by a TM $\mathscr{M}$ if $\mathscr{M}$ starts on an empty input and outputs a (potentially infinite) list of words that contains every word in $\mathscr{L}$ and nothing else

Theorem - A language is R.E iff some enumerator enumerates it

7 Church-Turing Thesis

The intuitive notion of algorithm is equivalent to the mathematical concept of algorithm defined by Turing Machines (or any other formal model of computation)

8 Universal Turing Machine

Proposition - Every TM $\mathscr{M}$ can be encoded as a word over a finite alphabet, We shall use $\langle \mathscr{M} \rangle$ to denote the encoding of a Turing machine $\mathscr{M}$
Theorem - There is a TM $\mathscr{U}$ that takes a two-part input, the encoding of a TM $\mathscr{M}$, $\langle \mathscr{M} \rangle$, and a word w, and simulate $\mathscr{M}$ on w. $\mathscr{U}$ is called a universal turing machine.

9 The Halting Problem

Definition: The halting problem

Given a (encoding of a) TM $\mathscr{M}$, and a word w, does $\mathscr{M}$ terminate on w?

Proposition: The Halting Problem is **Turing-recognisable**
Proof: Run a Universal TM on the pair $(\langle \mathscr{M} \rangle, w)$. Accept if the computation eventually terminates.
Proposition: The halting problem is not **Turing-decidable**

Proof: Assume for contradiction that there is a TM $\mathscr{H}$ that decides the halting problem

$$\mathscr{H}(\langle \mathscr{M} \rangle, w) = \begin{cases} \text{accept} & \text{if } \mathscr{M} \text{ terminates on } w \\ \text{reject} & \text{if } \mathscr{M} \text{ does not terminate on } w \end{cases}$$

Consider the TM $\mathscr{D}$ that takes a TM $\mathscr{M}$ as an input does the following:

$$\mathscr{D}(\langle \mathscr{M} \rangle) = \begin{cases} \text{accept} & \text{if } \mathscr{H}(\langle \mathscr{M} \rangle, \langle \mathscr{M} \rangle) \text{ rejects} \\ \text{loop} & \text{if } \mathscr{H}(\langle \mathscr{M} \rangle, \langle \mathscr{M} \rangle) \text{ accepts} \end{cases}$$

What happens when $\mathscr{D}$ runs on its own encoding $\langle \mathscr{D} \rangle$?! There are two possibilities:

1. $\mathscr{D}$ terminates on $\langle\mathscr{D}\rangle$. By the construction of $\mathscr{D}$, we have that $\mathscr{H}(\langle\mathscr{D}\rangle, \langle\mathscr{D}\rangle)$ rejects, and by the definition of $\mathscr{H}$, it follows that $\mathscr{D}$ does not terminate on $\langle\mathscr{D}\rangle$
2. $\mathscr{D}$ does not terminate on $\langle\mathscr{D}\rangle$. By the construction of $\mathscr{D}$, we have that $\mathscr{H}(\langle\mathscr{D}\rangle, \langle\mathscr{D}\rangle)$ accepts, and by the definition of $\mathscr{H}$, it follows that $\mathscr{D}$ terminates on $\langle\mathscr{D}\rangle$

10 Turing-Recognisable vs Turing-Decidable

Theorem: A language $\mathscr{L}$ is Turing-Decidable iff both $\mathscr{L}$ and its complement $\overline{\mathscr{L}}$ are Turing-recognisable

Proof (of the "interesting" direction only): Suppose $\mathscr{M}_1$ recognises $\mathscr{L}$ and $\mathscr{M}_2$ recognises $\overline{\mathscr{L}}$. On an input w, run $\mathscr{M}_1$ and $\mathscr{M}_2$ "in parallel" (i.e. simulate alternating steps of $\mathscr{M}_1$ and $\mathscr{M}_2$ on a multitape TM). Either $\mathscr{M}_1$ or $\mathscr{M}_2$ must eventually accept; accept it if $\mathscr{M}_1$ accepts and reject it is $\mathscr{M}_2$ accepts

11 Exercise

Given a TM $\mathscr{M}$ an input w and a number K. Does $\mathscr{M}$ terminate on w after at most K steps?

This is decidable:

$$StepCounter(\mathscr{M}, w, k)$$

This is undecidable as a quantifier has been added:

$$\exists k \quad StepCounter(\mathscr{M}, w)$$

$\mathscr{M}$ eventually terminates on w

co-Halt($\mathscr{M}, w$) - is it true that $\mathscr{M}$ doesn't terminate on w? This is undecidable

Semi Decidable - is there a program that would terminate on the yes instances?

12 Context Sensitive Languages

Definition: Linear Bounded Automaton (LBA)

A non-deterministic single-tape TM that can use only the part of the tape on which the input is initially written.

Definition: Context-Sensitive language

Some LBA recognises the language

Examples of languages that are context-sensitive but not context-free

$$\{a^n b^n c^n | n \in \mathbb{N}\}$$
$$\{ww | w \in \{0,1\}^*\}$$

13 Chomsky Hierarchy

1. **Regular**: Rules of the form $U \to aV$ or $U \to a$, where U and V are variables and a is either a terminal of ϵ
2. **Context-Free**: Rules of the form $U \to s$, where u is a variable and s is any string of variables and terminals
3. **Context-Sensitive**: Rules of the form $t \to s$, where t, s are string of variables and terminals, t contains at least one variable and $|t| \leqslant |s|$
4. **General (Turing-recognisable)**: rules of the form $t \to s$, where t, s are string of variables and terminals and t contains at least one variable

13.1 Example $a^n b^n c^n$

$$
\begin{aligned}
S &\to aSBC \\
S &\to \varepsilon \\
CB &\to BC \\
aB &\to ab \\
bB &\to bb \\
bC &\to bc \\
cC &\to cc
\end{aligned}
$$

14 Real world

1. **Regular**: (advanced) text search of regular expressions (grep), lexical analysers (lex)
2. **Context-Free**: the syntax of programming languages, i.e. (yacc, bison)
3. **Context-Sensitive**: none?
4. **General (Turing-recognisable)**: everything a computer can do