This cheat sheet is based on the textbook by Papadimitriou (1994).

Deterministic single-tape Turing machine

A Turing machine is a quadruple $M = (K, \Sigma, \delta, s)$ where

- K is a finite set of **states** and $s \in K$ is a designated **initial state**,
- Σ is a finite set of **symbols** (the **alphabet** of M) so that $\triangleright, \sqcup \in \Sigma$
- $\triangleright$ is the **start symbol** and
- $\sqcup$ is the **blank symbol**,
- δ is the **transition function**:

  $$\delta : K \times \Sigma \to (K \cup \{h, yes, no\}) \times \Sigma \times \{\to, \leftarrow, -\}$$

  where the **halting state** h, the **accepting state** *yes*, and the **rejecting state** *no* are not in K, and the symbols $\to$ (right), $\leftarrow$ (left), and $-$ (stay) indicate **cursor directions** on the input tape.

Transition functions

For current state $q \in K$ and current symbol $\sigma \in \Sigma$, $\delta(q, \sigma) = (p, \rho, D)$ where

- p is the new state,
- ρ is the symbol to be replacing σ, and
- $D \in \{\to, \leftarrow, -\}$ is the direction in which the cursor will move.

It is required that $\triangleright$ alway directs the cursor to the right and is never erased. Formally, for any states, p and q we have $\delta(q, \triangleright) = (p, \triangleright, \to)$.

If the machine moves off the right end of the tape, it reads the black symbol $\sqcup$. The string of the tape can become longer, but not shorter. The blanks $\sqcup$ keep track of the space used by the machine.

Starting and Halting

The program starts with

- initial state s,
- the tape contents initialized to $\triangleright\, x$ where the **input** x is a finitely long string in $(\Sigma - \{\sqcup\})^*$ and
- the cursor is pointing to $\triangleright$.

Machine has **halted** when it has reached one of the halting states $\{h, yes, no\}$. On *yes*, machine **accepts** the input, and on *no* machine **rejects** the input.

The **output** $M(x)$ is

- If M accepts/rejects, then $M(x) = yes/no$.
- If M reaches state h, then $M(x) = y$, where $\triangleright\, y\sqcup\sqcup...$ is the string on the tape of M at the time of halting.
- If M does not halt, then $M(x) = \nearrow$.

Operational Semantics

A **configuration** of machine M is a triple (q, w, u), where

- $q \in K$ is the current state,
- $w \in \Sigma^+$ is the string to the left of the cursor, including the symbol scanned by the cursor, and
- $u \in \Sigma^*$ is the string to the right of the cursor

The relation $\to^M$ **yields in one step** $(q, w, u) \to^M (q', w', u')$, where q', w', u' are obtained according to the transition function.

The relation **yields in** k **steps** $(q_1, w_1, u_1) \to^{M^k} (q_k, w_k, u_k)$ if there exists configurations

$$(q_1, w_1, u_1) \to^M (q_2, w_2, u_2) \to^M ... \to^M (q_k, w_k, u_k)$$

The relation **yields** $(q, w, u) \to^{M^*} (q', w', u')$ if there exists some $k \geq 0$ such that $(q, w, u) \to^{M^k} (q', w', u')$.

Decidable and Semidecidable Languages

Let $L \subseteq (\Sigma - \{\sqcup\})^*$ be a **language**.

A Turing machine M **decides** L, if for every string $x \in (\Sigma - \{\sqcup\})^*$,

- if $x \in L$, then $M(x) = yes$ and
- if $x \in L$, then $M(x) = no$.

If L is decided by some Turing machine, L is called a **decidable** language.

A Turing machine M **computes** a function

$$f : (\Sigma - \{\sqcup\})^* \to \Sigma^*,$$

if for every string $x \in (\Sigma - \{\sqcup\})^*$, $M(x) = f(x)$. If such an M exists, f is called a **computable** function.

A Turing machine M **accepts** or **semidecides** L, if for every string $x \in (\Sigma - \{\sqcup\})^*$,

- if $x \in L$, then $M(x) = yes$ and
- if $x \in L$, then $M(x) = \nearrow$.

If L is accepted by some Turing machine, L is called a **semidecidable** language.

Deterministic k-tape Turing machine

A **k-tape Turing machine**, for some integer $k \geq 1$, is a quadruple $M = (K, \Sigma, \delta, s)$ where the transition function is generalized to handle k-tapes simultaneously

$$\delta : K \times \Sigma \to (K \cup \{h, yes, no\}) \times (\Sigma \times \{\to, \leftarrow, -\})^k.$$

Transitions for k-tape machines are of the form

$$\delta(q, \sigma_1, ..., \sigma_2) = (p, \rho_1, D_1, ..., \rho_k, D_k).$$

A **configuration** is defined as a $2k+1$-tuple

$$(q, w_1, u_1, ..., w_k, u_k).$$

A k-tape machine with input x starts from the configuration

$$(s, \rhd, x, \rhd, \epsilon, ..., \rhd, \epsilon),$$

where ϵ is the empty string.

Output is defined as for standard machines

- if $(s, \rhd, x, \rhd, \epsilon, ..., \rhd, \epsilon) \to^{M^*} (h, w_1, u_1, ..., w_k, u_k)$, then $M(x) = y$ where y is $w_k u_k$ with the leading $\rhd$ and trailing $\sqcup$s removed, that is, output is read from the last (kth)tape.

The **runtime** of M on input x is t if

$$(s, \rhd, x, \rhd, \epsilon, ..., \rhd, \epsilon) \to^{M^t} (H, w_1, u_1, ..., w_k, u_k),$$

where $H \in \{h, yes, no\}$. If $M(x) = \nearrow$, then the runtime is considered to be ∞.

Time Complexity

Machine M **operates within time** $f(n)$, if for any input string x, the runtime by M on x is at most $f(|x|)$ where $|x|$ is the size of the input x.

Also, $f(n)$ is **(upper) time bound** for M and the language L decided by M belongs to the **time complexity class TIME**$(f(n))$.

The set of all languages decidable by deterministic Turing machines in polynomial time is defined as:

$$\mathbf{P} = \bigcup_{k>0} \mathbf{TIME}(n^k).$$

Space Complexity

A k-tape Turing machine $k > 2$ **with input and output** is an ordinary k-tape Turing machine with the following restrictions on the transitions function δ:

If $\delta(q, \sigma_1, ..., \sigma_k) = (p, \rho_1, D_1, ..., \rho_k, D_k)$, then

1) $\rho_1 = \sigma_1$ (read-only input string)
2) $D_k \neq \leftarrow$ (write-only output string), and
3) if $\sigma_1 = \sqcup$, then $D_1 = \leftarrow$ (end of input respected).

Space Usage

Suppose for a k-tape Turing machine M and an input x we have

$$(s, \rhd, x, \rhd, \epsilon, ..., \rhd, \epsilon) \to^{M^*} (H, w_1, u_1, ..., w_k, u_k),$$

where $H \in \{h, yes, no\}$ is Halting state. Then, the **space used** is

$$\sum_{i=1}^{k} |w_i u_i|$$

If M is a Turing machine *with input and output*, the space used is

$$\sum_{i=2}^{k-1} |w_i u_i|$$

We exclude the effect of reading the input and writing the output as regards TM space usage.

Let $f : \mathbb{N} \to \mathbb{R}^+$. Turing machine M **operates within space** $f(n)$ if for any input x, M uses space at most $f(|x|)$.

Space Complexity Classes

The **space complexity class SPACE**$(f(n))$ comprises the family of languages L that can be decided by Turing machines with input and output operating within space $f(n)$.

The class **SPACE**$(log(n))$ is denoted by **L**.

Nondeterministic Turing Machines

A nondeterministic Turing machine (NTM) is a quadruple $N = (K, \Sigma, \Delta, s)$ where Δ is a **transition relation**:

$$\Delta \subseteq (K \times \Sigma) \times [(K \cup \{h, yes, no\}) \times \Sigma \times \{\to, \leftarrow, -\}]$$

Yields is a relation $(q, w, u) \vdash_N (q', w', u')$ if there exists a tuple in Δ that makes this a legal transition. We have relations $\vdash_N^k$ and $\vdash_N^*$ defined as previously.

A nondeterministic Turing machine N **decides** a language L if for any $x \in \Sigma^*$, the following holds.

1) all the computation sequences of N on input x halt, and
2) $x \in L$ iff at least one of them ends in-state *yes*

Time Complexity Classes

A nondeterministic Turing machine N decides a language L **in time** $f(n)$ if N decides L and for any $x \in \Sigma^*$, if $(x, \triangleright, x) \vdash_N^k (q, w, u)$, then $k \leq f(|x|)$.

The time complexity class $\mathbf{NTIME}(f(n))$ comprises the family of languages L that can be decided by nondeterministic Turing machines in time $f(n)$.

The family **NP** of all languages decidable by nondeterministic Turing machines in polynomial time is defined as

$$\mathbf{NP} = \bigcup_{k>0} \mathbf{NTIME}(n^k).$$

Space Complexity Classes

Given a k-tape NTM N with input and output, we say that N decides language L **within space** $f(n)$ if N decides L and for any $x \in (\Sigma - \{\sqcup\})^*$, if $(s, \triangleright, x, \triangleright, \epsilon, ..., \triangleright, \epsilon) \vdash_N^* (h, w_1, u_1, ..., w_k, u_k)$, then $\sum_{i=2}^{k-1} |w_i u_i| \leq f(|x|)$.

Universal Turing Machines and Undecidability

Encoding TMs using Integers

Encoding a Turing machine $M = (K, \Sigma, \delta, x)$ using integers:

1) $1, 2, ..., |\Sigma|$ encode symbols Σ
2) $|\Sigma|+1, ..., |\Sigma|+|K|$ encode states K where $s = |\Sigma|+1$
3) $|\Sigma| + |K| + 1, ..., |\Sigma| + |K| + 6$ encode $\leftarrow, \rightarrow, -, h, yes, no$

Turing machine $M = (K, \Sigma, \delta, x)$ is encoded as

$$b(|\Sigma|); b(|K|); e(\delta)$$

where $b(k)$ denotes an encoding of integer k with exactly $\lceil \log(|\Sigma| + |K| + 6) \rceil$ bits and $e(\delta)$ is a sequence of pairs $((q, p), (p, \rho, D))$ describing the transition function δ.

Universal Turing Machine

A **universal Turing machine** U takes as input a description (encoding) of another Turing machine M and an input x for M, and the simulates M on x so that $U(M; x) = M(x)$.

Halting Problem

`HALTING` problem

- Instance: The description of a Turing machine M and its input x.
- Question: Does M halt on x?

The corresponding language is defined as

$$H = \{M; x \mid M(x) = \nearrow\}.$$

The Halting problem (the language H) is semidecidable.

Halting is undecidable.

Undecidability

Assume two languages B and A. A **reduction from** B **to** A is a transformation t of the input y of B to the input $t(y)$ of A such that, for all strings y, it holds that

$$y \in B \text{ if and only if } t(y) \in A.$$

Problem A is undecidable if the algorithm for deciding A implies an algorithm for deciding the halting H. It can be shown by devising a reduction t from halting H to A.

Suppose A were decided by a Turing machine M_A. Then H would be decided by a machine M_H that on input $M; x$.

$M_H(M; x)$:

1) $y \leftarrow M_t(M; x)$
2) $M_A(y)$

First, runs the machine M_t computing the transformation t. Then, runs M_A on the result.

Further Undecidable Problems

The following languages are not decidable:

1) $T = \{M \mid M \text{ halt on all inputs}\}$. Correspond to problem `TOTAL`
2) $\{M; x \mid M(x) = y \text{ for some } y\}$
3) $\{M; x \mid \text{the computation of } M \text{ on input } x$ uses all states of $M\}$
4) $\{M; x; y \mid M(x) = y\}$

A reduction of `HALTING` to `TOTAL`:

- Given input $M; x$, consider a machine M_x that works as follows: $M_x(y)$: if $y = x$ then $M(x)$ else halt.
- Define a reduction mapping $t(M; x) = M_x$. (That is, the input x is hardcoded into the machine code of M and the results is the new code.)
- Now $M; x \in H$ iff M halts on x iff M_x halts on all input iff $M_x \in T$.

References

Papadimitriou, C.H., 1994. *Computational complexity.* Addison-Wesley.pp.I–XV, 1–523.