1 Non-deterministic Turing Machine

A **nondeterministic Turing machine** is a generalization of the standard TM for which every configuration may yield none, or one or more than one next configurations.

In contrast to the deterministic Turing machines, for which a computation is a sequence of configurations, a computation of a nondeterministic TM is a tree of configurations that can be reached from the start configuration.

In this tree, the children-nodes of a node are its next configurations. Thus, the configuration, whose state is either $q_a$, or $q_r$ has no children-nodes.
A **nondeterministic Turing machine**, written $NTM$, is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \Delta, q_0, q_a, q_r),$$

where all ingredients except for $\Delta$ are defined as before for the deterministic TM.

$$\Delta : (Q \times \Gamma) \longrightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\}).$$

The transition function of an NTM may correspond, for a given pair $(q, \sigma)$, a set of triples $\{(p, \sigma', D)\}$; this set can be empty.

The transition function is sometimes presented as a set $\Delta$ of pairs:

$$((q, \sigma)(p, \sigma', D)).$$

While for a deterministic TM, there can be just one pair with a given first term $(q, \sigma)$, for a non-deterministic TM, $\Delta$ may have more than one, or none, pair with a given first term.

Thus, for each combination of a state and a tape-symbol, there can be more than one appropriate steps, or none at all. A configuration of an NTM may yield several, but a finite configurations in one step.
An input to an NTM is said to be **accepted** if there exists at least one node of the computation tree which is an accept-configuration.

The path from the root to the accept-configuration is said to be selected **non-deterministically**.

A non-deterministic Turing Machine is called a **decider** if all branches halt on all inputs.

If, for some input, all branches are rejected, then the input is rejected.

NDTM computation tree: schematic representation
Example. A high-level description of an NTM which accepts composite numbers $L$ in the unary representation.

$$L = \{I I \cdots I : m \text{ is a composite integer.}\}.$$

Given an input $I I \cdots I \equiv I^m$, if $m$ is not a prime, i.e.

$m = p \times q$ for some integers $p, q < m$, the machine performs the following instructions:

1. Non-deterministically choose two integers $p$ and $q$ ($p, q \neq m$);

2. transform the input configuration $(\epsilon \ q_{\text{start}} \ I^m)$ into

   $$(I^m \ # \ I^p \ # \ I^q \ # \ d \ \epsilon)$$

   /* here # serves as a separator; $d$ is a state of the control; */

3. Multiply $p$ and $q$, that is, transform $(I^m \ # \ I^p \ # \ I^q \ # \ d \ \epsilon)$

   into $(I^m \ # \ I^{pq} \ # \ f \ \epsilon)$;

4. Compare the length of $m$ with $p \times q$: accept if they are the same, otherwise reject.
How to interpret the instruction “Non-deterministically choose two numbers $p$ and $q$”?

To **non-deterministically** select $p$ and $q$:

- starting from a given square, repeat $p$ times: write $I$, move to the right.
- write #, move to the right.
- repeat $q$ times: write $I$, move to the right.

```
I I ... I # I ... I # I ... I #
```

1. Non-deterministically write in $I^p I^q$ ($p > 1$, $q > 1$)
2. Deterministically generate in $I^{p \times q}$
3. Deterministically compare $I^m$ with $I^{p \times q}$
Definition 1  Two TM’s $M_1$ and $M_2$ are said to be computationally equivalent, if $L(M_1) = L(M_2)$.

Theorem 1

Every non-deterministic Turing machine has an equivalent deterministic Turing machine. If $L$ is decided by an NDTM $N$ in time $f(n)$, then there is a deterministic TM which decides $L$ in time $O(c^{f(n)})$, for some $c > 1$.

Proof. Let $N = (Q, \Sigma, \Gamma, \Delta, q_0, q_a, q_r)$ and let $d$ be the smallest integer such that for each $(q, \sigma)$ there is at most $d$ choices for $N$.

A deterministic TM $D$ traverses all nodes of the computational tree of $N$. Although the traversal can be exponentially longer than a path from the start configuration to the accepting configuration, it is finite, since every node of the tree has a finite and bounded number of children-nodes. The latter is determined by the cardinality of $|\delta(Q \times \Gamma)|$, which is a finite number since $\delta(Q \times \Gamma) \subseteq Q \times \Gamma \times \{L, R\}$.
\(D\) has three tapes (we already saw that such multi-tape TM is equivalent to a single tape TM):

**input tape:** contains the input \(w\) (never changes);

**simulation tape:** maintains a copy of \(N\)’s tape on a branch of its nondeterministic computation;

**address tape:** keeps track of the current location of \(D\) in the \(N\)’s computational tree; in particular, enumerates tree-paths in the lexicographical order.
$D$ works as follows:

1. Insert the input $w$ to tape 1; empty tapes 2 and 3.
2. Copy tape 1 to tape 2.
3. Simulate one branch of nondeterministic computation. Before each step of $N$, consult the next symbol on tape 3 to determine which choice to make among those allowed by $N$’s transition function.
   - if no more symbols remain on tape 3 or if this nondeterministic choice is invalid, abort this branch by going to stage 4.
   - if a rejection configuration is encountered, go to stage 4.
   - if an accepting configuration is encountered, accept the input.
4. Replace the string on tape 3 with the lexicographically next string. Simulate the next branch of $N$’s computation by going to stage 2.

**Corollary.** A language is Turing-recognizable (acceptable) iff some nondeterministic Turing machine recognizes (accepts) it.
2 Examples of non-deterministic Turing Machines

Example 1

Given a set $S = \{a_1, \ldots, a_n\}$ of integers, determine if there is a subset $T \subseteq S$ such that

$$\sum_{a_i \in T} a_i = \sum_{a_i \in S - T} a_i.$$

The task is to construct an NDTM which accepts a language $L$ corresponding to the problem.

Language:

$L = \{a_1 \# a_2 \# \ldots a_m \#: \exists T \subseteq S, \text{ such that } \sum_{a_i \in T} a_i = \sum_{a_i \in S - T} a_i\}.$

Assume that there are two auxiliary TM (deterministic):

- $C$, a copy machine: copies a specified string on the tape to a specified location;
- $Sum$, a summation machine: sums up specified numbers.
**NDTM** which accepts $L$;

1. Place $S$ onto the tape;
2. sum up the numbers in the input; let the result be $A_1$
3. append the string with $\$);
4. while moving from the left to $\$
   
   nondeterministically copy all $a_i \in T$
   to the right from the rightmost $\#$
   append with $\#$
5. sum up the numbers that were copied; let the result be $A_2$
6. accept if $A_1 = 2A_2$
Example 2
Given a graph $G = (V, E)$ and an integer $k > 0$, determine if there is a subset $C \subseteq V$ such that

- $|C| \geq k$;
- any two vertices in $C$ are adjacent ($C$ is a clique).

Present the problem as one of accepting a language $L$; describe an NDTM to accept $L$.

Solution. Define language $L$ as follows:

$$ L = \{ \langle G, k \rangle : G \text{ has a clique of size } \geq k. \} $$

/* we assume that there is some standard way of presenting $G$ as a string in a finite alphabet */

Assume there is a TM which, given two vertices of $G$, answers if these vertices are adjacent.
**NDTM** which accepts $L$;

1. Place $\langle G, k \rangle$ onto the tape;
2. append the string with $\$$;
3. while moving from the left to $\$$;
   - nondeterministically select some vertices $v_i \in V(G)$
   /*assumption: there is a “finite-choice” computational path which does the selection, e.g.
   it can be just having a vector of length $n$ with components 0 or 1 that define the selection */
4. check if the number of selected vertices $\geq k$;
5. for every two selected vertices check if they are adjacent;
6. accept if all pairs are adjacent
Example 3

Given a graph $G = (V, E)$ and an integer $k > 0$, determine if there is a path $P$ in $G$ such that

- the length of $P \geq k$;
- no two vertices in $G$ are traced twice by $P$.

Present the problem as one of accepting a language $L$; describe an NDTM to accept $L$.

**Solution.** Define language $L$ as follows:

$$L = \{ \langle G, k \rangle : G \text{ there is a path of length } \geq k \}$$

Assume there is a TM which, given two vertices of $G$, answers if these vertices are adjacent.
**NDTM** which accepts $L$;

1. Place $\langle G, k \rangle$ onto the tape;
2. append the string with $\$$;
3. while moving from the left to $\$$;
   
   nondeterministically select some vertices $v_i \in V(G)$
   
   /* we assume here without going into details that there is a “finite-choice” computational path which does the selection; for example, it can be just having a vector of length $n$ with components 0 or 1 that define the selection */

4. check if the number of selected vertices $\geq k$;
5. for every two consequently selected vertices check if they are adjacent;
6. accept if all checked pairs of vertices are adjacent.