Definition 1. **Deterministic automata:** For each combination of state and input symbol, there is at most one transition. (One transition out of each state for each input symbol.) In other words, it can't be in more than one state at a time.

Definition 2. **Nondeterministic automata:** It is possible to have multiple transitions for the same combination of state and input symbol. In other words, it can be in more than one state at the same time. This situation is common in particle physics and computers.

Remark 0.1. The operation of a Turing machine M is defined using a *configuration* which contains a complete description of the current computation.

Definition 3. A *configuration* is a triple (q, w, u) where q is the current state, w is the string to the left of the cursor (includes also the one currently scanned by the cursor), and u is the string to the right of the cursor.

Definition 4. We say

$$\text{configuration } (q, w, u) \textit{ yields} \text{ configuration } (q', w', u') \text{ in } \textit{one step},$$

denoted by

$$(q, w, u) \xrightarrow{M} (q', w', u')$$

if a step of the machine from (q, w, u) results in (q', w', u').

And we say

$$\text{configuration } (q, w, u) \textit{ yields} \text{ configuration } (q', w', u') \text{ in } k \textit{ steps},$$

denoted by

$$(q, w, u) \xrightarrow{M^k} (q', w', u')$$

if k steps of the machine from (q, w, u) result in (q', w', u').

And finally we say

$$\text{configuration } (q, w, u) \textit{ yields} \text{ configuration } (q', w', u'),$$

denoted by

$$(q, w, u) \xrightarrow{M^*} (q', w', u')$$

if there exists $k \geq 0$ such that

$$(q, w, u) \xrightarrow{M^k} (q', w', u').$$

Definition 5. Let M be a Turing machine and let Σ be an alphabet for M. Let L be a language over $\Sigma - \{\sqcup\}$. We say that M decides L if for any string x over $\Sigma - \{\sqcup\}$, we have that $M(x) = "yes"$ if $x \in L$ and $M(x) = "no"$ if $x \notin L$.

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

Definition 6. With the notation as above, we say that M accepts L if $M(x) = "yes"$ if $x \in L$ and $M(x) = \nearrow$ if $x \notin L$.

If L is accepted by some Turing machine, then L is called *recursively enumerable*.

REMARK 0.2. L recursive $\Longrightarrow$ L recrsively enumerable.

REMARK 0.3. if L is a recursive language, then so is $\overline{L}$.

DEFINITION 7. Let $f : (\Sigma - \{\sqcup\})^* \longrightarrow \Sigma^*$ be a function and let M be as above. We say M computes f if $\forall x \in (\Sigma - \{\sqcup\})^*$, $M(x) = f(x)$. If such M exists, then f is called a *recursive function.*

REMARK 0.4. Complexity theory is independent of input representation.

DEFINITION 8. A $k - string$ Turing machine, where $k \in \mathbb{N}$ is a quadruple

$$M = (K, \Sigma, \delta, s),$$

where K, Σ, and s are as before and δ is a function from

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

In other words,

$$\delta(p, a_1, a_2, ..., a_k) = (q, b_1, D_1, b_2, D_2, ..., b_k, D_k).$$

This means that if M is in state p and if the cursor of the first string is skanning a_1, the cursor of the second string is skanning a_2, and so on, then the machine will change to state q and the first cursor will write b_1 and move in the direction D_1, the second cursor will write b_2 and move in the direction D_2, and so on.

REMARK 0.5. Each one of the k strings starts with $\triangleright$. The input is in the first string and the output is in the last string if the machine halts at h and the output is "yes"/"no" if the machine halts at "yes"/"no".

If the input of a $k - string$ Turing machine is x, then it starts with the configuration

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

If

$$(s, \triangleright, x, \triangleright, \epsilon, ..., \triangleright, \epsilon) \xrightarrow{M^*} (\text{"}yes\text{"}, w_1, u_1, ..., w_k, u_k),$$

then $M(x) = \text{"}yes\text{"}$.

If

$$(s, \triangleright, x, \triangleright, \epsilon, ..., \triangleright, \epsilon) \xrightarrow{M^*} (\text{"}no\text{"}, w_1, u_1, ..., w_k, u_k),$$

then $M(x) = \text{"}no\text{"}$.

If

$$(s, \triangleright, x, \triangleright, \epsilon, ..., \triangleright, \epsilon) \xrightarrow{M^*} (h, w_1, u_1, ..., w_k, u_k),$$

then $M(x) = y$, where y is $w_k u_k$ with $\triangleright$ and all blanks at the end removed.

Note: We mean by the cursor the tapehead.

DEFINITION 9. The time required by a Turing machine M on an input x is the number of steps to halting. If the machine does not halt; i.e. $M(x) = \nearrow$, then the time is ∞.

DEFINITION 10. Let f be a function from the nonnegative integers to the nonnegative integers, then we say that M operates within time $f(n)$ if for any input x, the time required by M on x is less than or equal to $f(|x|)$.

DEFINITION 11. **TIME(f(n))**: Complexity class of languages that are decided by multistring $(k - string)$ $(k - tape)$ Turing machines operating within time $f(n)$.

DEFINITION 12. P**:** set of all languages decidable by Turing machines in polynomial times (i.e. TIME(n^k), $k \geq 1$).

THEOREM 13. *Given any multistring (k-string) Turing machine M_k operating within time $f(n)$, there exists a Turing machine M operating within time $O(f(n)^2)$ such that $M(x) = M_k(x)$ for any input x.*

REMARK 0.6. The above thorem says that

1. single-string Turing machines are as powerful as multistring (k-string) Turing machines. I.e. they decide exactly the same languages and compute exactly the same recursive functions. In other words, they have the same computational capabilities.
2. A multi-string (k-string) Turing machine can be more efficient (faster) than a single-string Turing machine. But, the saving in time is at most polynomially.

EXAMPLE 14. A single-string Turing machine which desides *palindromes* is $O(n^2)$, while a two-string Turing machine that decides palindromes is $O(n)$.