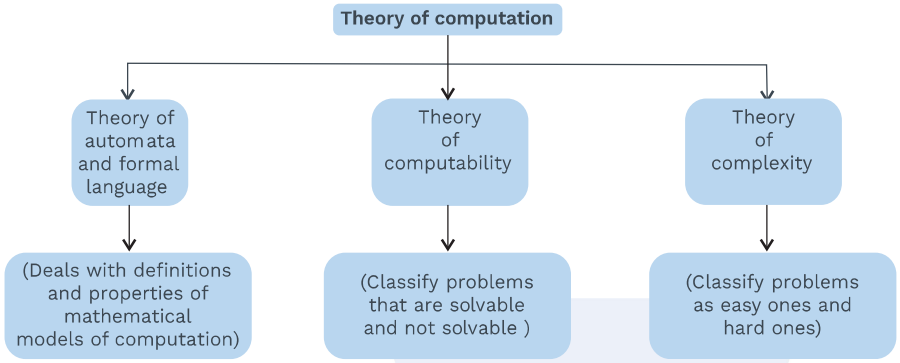
1 Introduction of Theory of Computation

1.1 INTRODUCTION OF THEORY OF COMPUTATION:

Theory of computation: The study of mathematical representation of computing machines & their capabilities.



Why study theory of computation?

It provides mathematical models of a computer:

What can/can't a computer do?

The efficiency of a computer

Finite automata: Useful models for many important kinds of hardware and software.

Pushdown automata: More capable than finite machines but less capable than liner bound automata turning machines.

Turing machines: Mathematical model of computation that defines an abstract machine, which is most powerful as it has unlimited memory.

1.2 INTRODUCTION

Symbol:

A symbol is a member of an alphabet.

OR

Symbol is an abstract entity that we shall not define formally, just as a point and lines which are not defined in geometry.

Letters and digits are examples of frequently used symbols.

Alphabet:

A finite non-empty set of symbols is known as alphabet. It is represented by $\sum$.

Example: $\sum_{binary} = \{0,$

$\sum_{decimal} = \{0, 1, 2, ...$

String (Word):
A string is finite sequence of alphabets. It is denoted by w.

Example: 1001 is a string from binary alphabet $\Sigma = \{0, 1\}$.

xyz is a string from alphabet $\Sigma = \{x, y, z\}$

Length of string:
Number of alphabets in the string.

Example: w = xyzwu
|w| = 5

Empty string:

- This is a string with no symbol at all.
- It is denoted by $\in$ (Epsilon), λ.
- Length of empty string = 0

Substring:
u is a substring of w if $\exists$ xuy = w, where u, x, y, w$\in \Sigma^*$

Example: $\underset{x}{a}\underset{u}{bcd}\underset{y}{e}$

$\underset{x}{\in}\underset{u}{mno}\underset{y}{pq}$

Note:

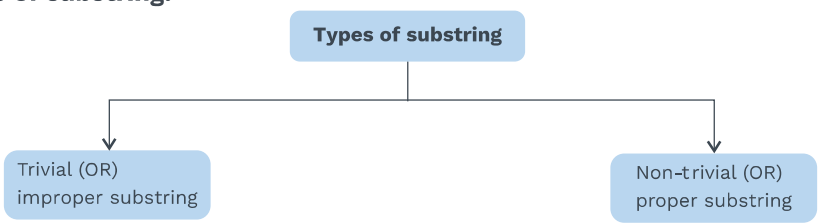
a) Every string is a substring to itself.

b) '$\in$' i.e. empty string is substring to every string.

Rack Your Brain

How many substring are there for word "COMPUTATION"?

Type of substring:



1) Trivial (OR) improper substring:
If 'w' is any string, then '$\in$' and 'w' itself is called trivial substring.

2) Non-Trivial (OR) proper substring:

If 'w' is any string over $\sum$, then any substrings of w other than '$\in$' and 'w' is called non-trivial substring.

Example: w = GATE

Trivial string = $\in$

Non-trivial string = G, A, T, E, GA, AT, TE, GAT, ATE

Note:

If 'w' is any string with a distinct symbol i.e. |w| = n, then

a) Number of substring = $\sum n+1 = \frac{n(n+1)}{2}+1$

b) Number of trivial substring = 2

c) Number of non-trivial substring = $\sum n-1 = \frac{n(n+1)}{2}-1$

Operation on string:

Prefix:

'u' is a prefix of 'w' if $\exists x$ such that ux = w where $u, x, w \in \sum^*$

Example: abcde

u

w

prefixes = $\in$, a, ab, abc, abcd, abcde

Proper prefix: A prefix of a string other than the string itself is known as a proper prefix.

In other words 'u' is a proper prefix of 'w' if u≠w.

Suffix:

'v' is a suffix of 'w' if $\exists x$ such that xv = w where $v, x, w \in \sum^*$

Example: abcde

v

w

Suffixes = $\in$, e, de, cde, bcde, abcde

Proper Suffix: A suffix of a string other than the string itself is known as a proper suffix.

In other words, 'v' is a proper suffix of 'w' if v≠w.

Rack Your Brain

If Lenth of string i.e. |w| = n, then how many number of prefixes and suffixes are possible?

Concatenation of strings:

The concatenation of two strings w and v is the string obtained by appending the symbols of v to the right end of w, that is, if

$w = a_1a_2.....a_n$

and $v = b_1b_2.....b_m$,

then the concatenation of w and v, denoted by wv, is

$wv = a_1a_2.....a_nb_1b_2.....b_m$

Let x any y be strings, then xy denotes the concatenation of x and y.

Example: Let x = 1101 and y = 011, then

xy = 1101011

and yx = 0111101

Note:

$\in .w = w. \in = w$ holds

Reverse of strings:

The reverse of a string is obtained by writing the symbols in a reverse order. If w is a string where $w = a_1a_2.....a_n$, then its reverse w^R is

$w^R = a_n.....a_2a_1$

Example: If w = abcd, then w^R = dcba

Rack Your Brain

"$W=W^R$ iff w is palindrome"

Is it true?

Power of an alphabet:

If Σ is an alphabet, then Σ^k be the set of strings of length k, each of whose symbol is in Σ.

Example: Σ = {a,b,c}

$\Sigma^0 = \{\in\}$

$\Sigma^1 = \{a,b,c\}$

$\Sigma^2 = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$

$\Sigma^3 = \{aaa, aab, aba, aac, abb, ...ccc\}$

Σ^* = Set of all strings over an alphabet Σ

$$\Sigma^* = \bigcup_{i \geq 0} \Sigma^i = \{w \mid |w| \geq 0\}$$

$= \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots\ldots\ldots\ldots\ldots\ldots$ (Kleene closure)

$$\Sigma^+ = \bigcup_{i \geq 1} \Sigma^i = \{w \mid |w| \geq 1\}$$

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots\ldots\ldots\ldots\ldots\ldots$ (Positive closure)

$$\Sigma^* = \Sigma^+ \cup \{\epsilon\}$$

Rack Your Brain

What will be the power set of Σ^*?

Basic properties of Σ^+ and Σ^*:

1) $\Sigma^+ \subset \Sigma^*$
2) $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$
3) $\Sigma^+ = \Sigma^* - \{\epsilon\}$
4) $\Sigma^* \cap \Sigma^+ = \Sigma^+$
5) $\Sigma^* \cup \Sigma^+ = \Sigma^*$
6) $\Sigma^* \Sigma^+ = \Sigma^+ \Sigma^* = \Sigma^+$

1.3 GRAMMAR:

- Grammar is a set of rules to produce valid strings in a formal language.

OR

The collection of production rules is used for the formation of strings.

- A grammar G is defined as a quadruple:

  G = (V, T, S, P)

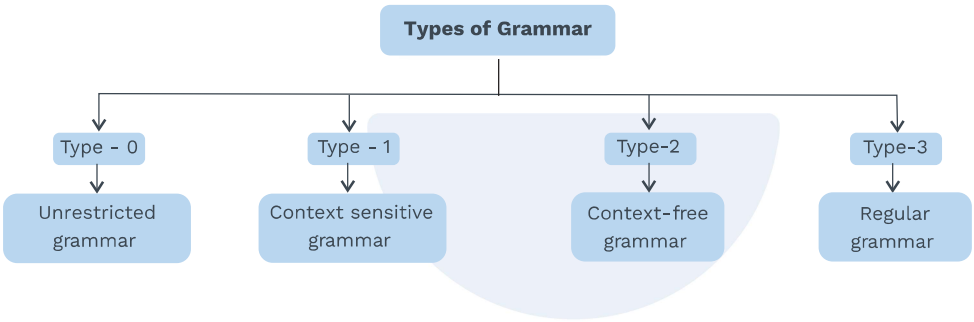
Where V is a finite set of objects called variables.
T is a finite set of objects called terminal symbols.
S $\in$ V is called the start variable.
P is a finite set of productions.

- It will be assumed that the sets V and T are non-empty and disjoint.

Example: 1) $S \rightarrow AB$ $A \rightarrow a$ $B \rightarrow b$

2) $S \rightarrow AB$ $A \rightarrow a$ $B \rightarrow b|c$

Types of grammar:



1.3.2 Chomsky hierarchy:

Chomsky's hierarchy represents the classes of formal grammar.

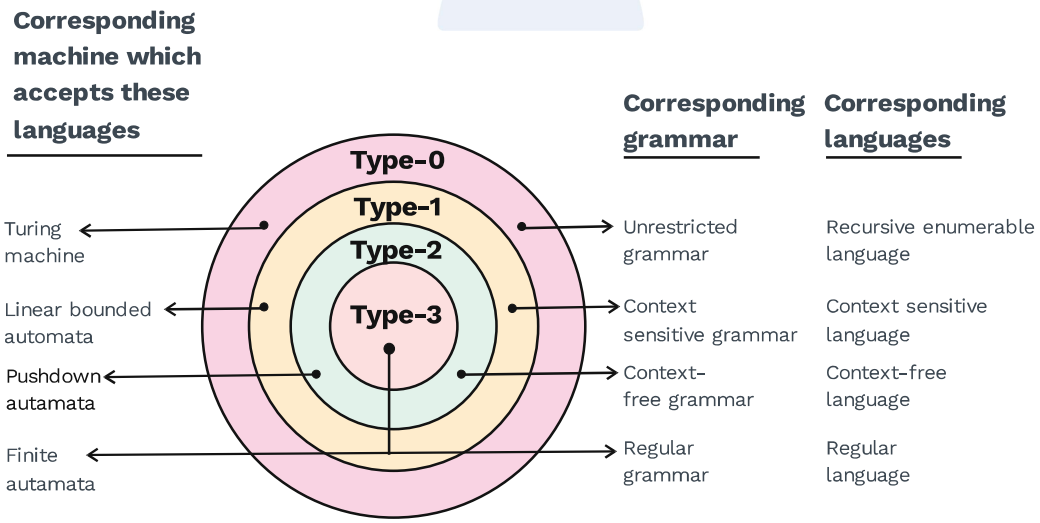


Fig. 1.1 Diagramatic Representation of Chomsky Hierarchy

1.4 LANGUAGES:

A set of strings, all of which are chosen from some Σ^*, where Σ is a particular alphabet is called a language.

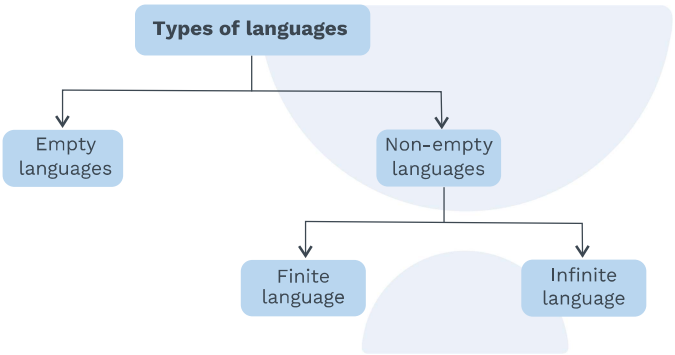
OR

Σ is any alphabet, then the set of all strings from the alphabet 'Σ' is called language.

Example: The language of all strings consisting of n number of 0's followed by n number of 1's for some $n \geq 0$.

$$L = \{\in, 01, 0011, 000111,.....\}$$

Types of languages:



Note:

Σ^* is a universal language.

Empty languages:

- Language which contains nothing i.e. $L = \phi$ does not even contain $\in$.
- Also called null language.
- Represented by ϕ.
- Cardinality of smallest language i.e. $L = \phi$ is $|L|$

Non-empty languages:

- Language which contains atleast one string is called non-empty Language.

Example:

$L = \{\in\}, |L| =$ cardinality of language

$L = \{a\}, |L| =$ cardinality of language

$L = \{a^n \mid n \geq 0\} = \{\in, a, aa.....\}$

Rack Your Brain

What will be the kleen star of an empty language?

Finite language:
The language which contains a finite number of strings where the length of each string in a language is finite.

Example:
$L = \{w \mid |w| \leq 2\}$ over $\Sigma = \{a, b\}$

$= \{\in, a, b, aa, ab, ba, bb\}$

$|L|$ = Length of language = 7

Infinite language:
The language which contains an infinite number of strings and cardinality of language is not finite.

Example:

$L = \{a^n b^n \mid n \geq 1\}$

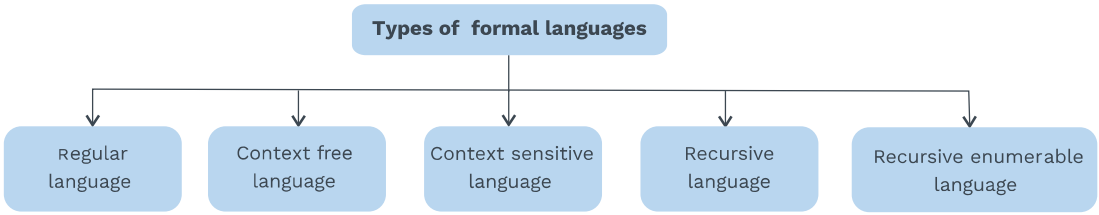
$= \{ab, aabb, aaabbb, ...\}$

Rack Your Brain

Will the complement of finite language always be infinite?

Formal language:
It is a collection of strings where the format of a string is important but meaning of a string is not important.

Types of formal language:



Regular language:

- A language is called a regular language if some finite automaton recognizes it.
- The empty language ϕ is a regular language.
- For each $a \in \Sigma$, the singleton language {a} is a regular language.
- If A is a regular language, then A* (kleen-closure) is a regular language.
- If A and B are regular languages, then $A \cup B$ and $A \bullet B$ (concatenation) are regular languages.

Example: $L = \{a^n \mid n \geq 0\}$ is a regular language.

Deterministic context-free language:

- Deterministic context-free languages (DCFL) are a proper subset of context-free languages.
- They are accepted by a deterministic pushdown automata.
- Deterministic context-free languages (DCFL) are always unambiguous. So, they accepts an unambiguous grammar.

Example: $L = \{a^n b^n c^m \mid m \geq 0 \text{ and } n \geq 0\}$

Context-free language:

- Context-free language is generated by context-free grammar (CFG).
- Most arithmetic expressions are generated by context-free grammar.

Example: $L = \{ww^R \mid w \in \{a.b\}^*\}$, w^R is a reverse of w.

Context sensitive language:

- Context sensitive language is defined by context sensitive grammar.

Example: $L = \{a^n b^n c^n \mid n \geq 1\}$

Recursive language:

- Recursive language is a subset of recursive enumerable languages.
- Recursive language can be decided by the turing machine.
- Turing machine will enter into the final state for the strings which are parts of language and enter into rejecting state for strings not in part of language.

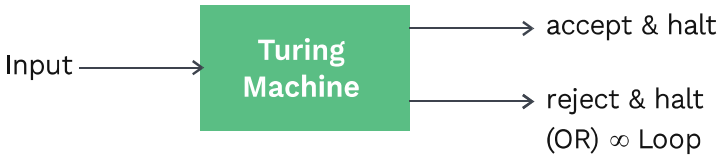
Example: $L = \{ww \mid w \in (a,b)^*\}$

Recursive enumerable language:

Recursive Enumerable Language is a formal language for which there exists a Turing Machine which will-

a) Halt and accept when any string present in the language.

b) But may either halt and reject or loop forever when the string not present in the language.



Example: $L = \{a^p \mid p \text{ is a prime number}\}$

1.5 AUTOMATA:

- Singular form of automata is known as automaton.
- Automata theory deals with the definitions and properties of mathematical models of computation.
- Automata theory is the study of abstract computing devices or machines.
- One model, called the finite automaton, is used in text processing, compilers, and hardware design. Another model, called the context-free grammar, is used in programming languages and artificial intelligence

Characteristics of such machines:

Includes:

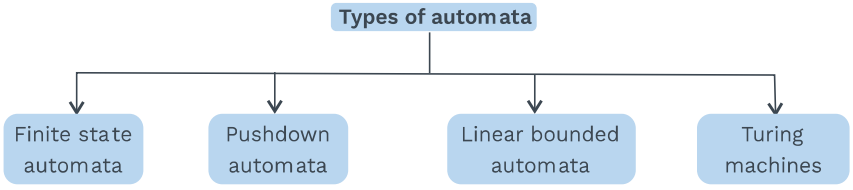
Inputs: It is assumed to be sequences of symbols selected from a finite set of input. Let $\{x_1, x_2, x_3, \ldots\ldots\ldots x_m\}$ is input set where m is the number of inputs.

Outputs: It is sequences of symbols selected from a finite set Z, where $Z = \{y_1, y_2, y_3, \ldots\ldots\ldots\ldots y_n\}$; n is number of outputs.

States: It is a finite set Q, whose definition depends on what type of automata is defined.

Types of automata:

There are 4 major families of automata:



Finite state automata (FA):

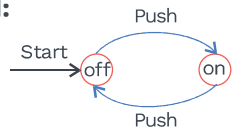
- Simplest machines.
- Finite automata are good models for computers with an extremely limited amount of memory.
- Abstract machines having 5 tuples.
- Formal specification of machines:

$\{Q, \Sigma, q_0, F, \delta\}$

Where
- Q : Finite set of states
- Σ : Set of input symbols
- q_0 : Initial State
- F : Set of final states
- δ : Transition function

Real world examples:

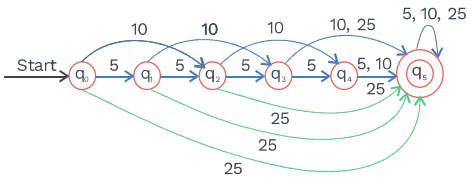
Example 1:



Odd number of pushes = ON State
Even number of pushes = OFF State

Fig. 1.2 A Finite Automaton: An On/Off Electric Switch

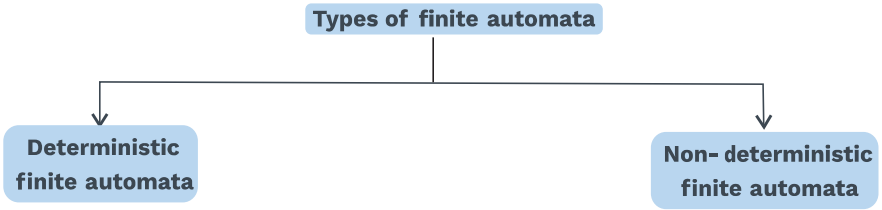
Example 2: Design a computer that controls a "toll gate". Assume that we have only 3 coins 5, 10 and 25 cents i.e. $\Sigma = \{5, 10, 25\}$. Make an assumption that any excess change amount is not refunded back. The driver, upon his arrival at the toll slots into the machine a number of coins. It is the decision of the machine if the way in is to be provided to the driver or not. The machine permits the driver to pave in if the amount is at least 25 cents.



When a car arrives at the toll gate, the gate is closed. The gate opens as soon as the driver paid 25 cents.

Fig. 1.3 A Finite Automata: Controlling a Toll Gate

Types of finite automata:



Pushdown automata (PDA):

- PDA is a finite automata with a stack.

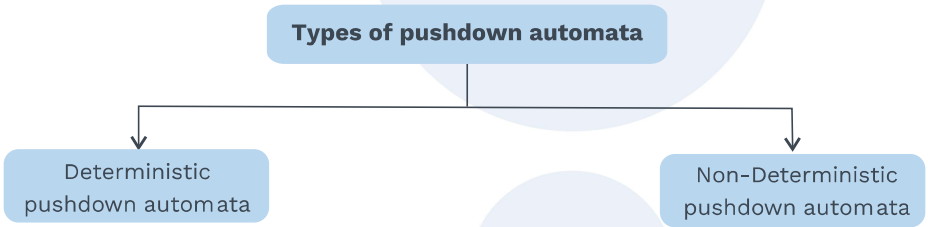
FA+1Stack=PDA

- It is used to recognize context-free languages.
- PDA is defined as:

$(Q, \Sigma, \delta, q_0, Z_0, F, \tau)$

Where
- Q : Finite set of states
- Σ : Input symbol
- δ : Transition function
- q_0 : Initial State
- Z_0 : Bottom of the stack
- F : Set of final states
- τ : Stack alphabets

Types of pushdown automata:

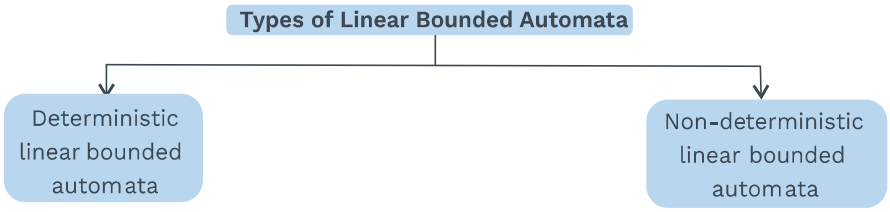


Linear bounded automata (LBA):

- A restrictive approach to the turing machine is depicted by LBA.
- A linear bounded automaton is designed with an unbounded tape. The extent up to which the tape needs to be made use of is purposely an instant of the input. Practically, the unbounded tape is restricted by the length of the input string. The read/write head is utilized to read or retrieve from the tape.
- They are classified as CSL acceptors.
- Linear bounded automata is defined as: $(Q, T, \Sigma, q_0, M_L, M_R, \delta, F)$

Where
- Q : Finite set of states
- T : Tape alphabet
- Σ : Input alphabet
- q_0 : Initial State
- M_L : Left Bound of tape
- M_R : Right Bound of tape
- δ : Transition function
- F : A finite set of final states

- **Types of linear bounded automata:**



Turing machine (TM):

- A Turing Machine is a mathematical model which consists of an infinite length tape.

FA+2Stack=PDA + 1 Stack = TM

- It consists of a head which reads input tape.
- Turing machine is hypothetical machine which can simulate any computer algorithm, however complicated it is.
- Turing machine is as powerful as a computer.

Turing machine is defined as:

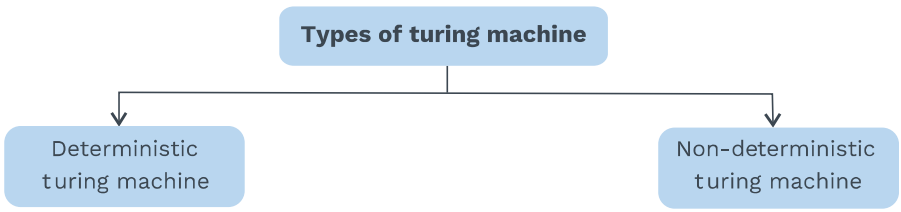
7 tuple ($Q, \Sigma, T, B, \delta, q_0, F$)

Where
- Q : Finite set of states
- T : Tape alphabet
- Σ : Input alphabet
- B : Blank symbol
- δ : Transition function
- q_0 : Initial State
- F : A finite set of final states

Rack Your Brain

Can we make any modification in standard Turing Machine and increase its power?

Types of turing machine:



Expressive power of an automata:
Number of languages accepted by that automata
i.e. E (TM) > E(LBA) > E(PDA) > E(FA)
4 3 2 1

Where E represents expressive power
TM represents turing machines
LBA represents linear bounded automata
PDA represents pushdown automata
FA represents finite automata

Expressive power of DFA and NFA:

Expressive power of DFA = Expressive power of NFA

i.e. DFA and NFA define the same class of languages.

Expressive power of DPDA and NPDA:

Expressive power of DPDA < Expressive power of NPDA

i.e. DPDA defines a class of language which is a proper subset of the class of languages defined by NPDA.

Expressive power of DLBA and NLBA:

Expressive power of DLBA $\overset{?}{=}$ Expressive power of NLBA ↓ Can't say (Not known)

i.e. It is not known whether deterministic linear bounded automata (DLBA) is more powerful or NLBA (Non-deterministic linear bounded automata).

Expressive power of DTM and NTM:

Expressive power of DTM = Expressive power of NTM

i.e. Deterministic turing machine (DTM) and non-deterministic turing machine (NTM) are equivalent in power.