Formal Languages and Automata Theory

Finite Automata

Finite Automata — Defintion

Let $A = (Q, T, q_0, \delta, F)$. It is a finite automaton (recognizer), where Q is the finite set of (inner) states, T is the input (or tape) alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final (or accepting) states and δ is the transition function as follows.

- $\delta : Q \times T \to Q$ (for deterministic[1] finite automata);
- $\delta : Q \times T \to 2^Q$ (for nondeterministic[2] finite automata).

Description One can imagine a finite automaton as a machine equipped with an input tape. The machine works on a discrete time scale. At every point of time the machine is in one of its states, then it reads the next letter on the tape (the letter under the reading head), and then, according to the transition function (depending on the actual state and the letter being read,) it goes to the next state. It may happen in some variations that there is no transition defined for the actual state and letter, then the machine gets stuck and cannot continue its run. This case the automaton is called **partially defined** finite automaton. Otherwise the automaton is called **completely defined** finite automaton.

Example Let the finite automaton A be

$$
\begin{aligned}
&A = (\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, q_0, \delta, \{q_2, q_3\}), \text{ where} \\
&\delta(q_0, a) = \{q_1\} \\
&\delta(q_0, b) = \{q_0\} \\
&\delta(q_0, c) = \{q_0\} \\
&\delta(q_1, a) = \{q_1\} \\
&\delta(q_1, b) = \{q_0\} \\
&\delta(q_1, c) = \{q_2\} \\
&\delta(q_2, a) = \{q_2, q_3\} \\
&\delta(q_3, b) = \{q_3\} \\
&\delta(q_3, c) = \{q_1, q_2, q_3\}.
\end{aligned}
$$

[1] Deterministic: one input, only one result.

[2] Non-deterministic: one input, more than one result.

Representation — Cayley Table:

$T \quad Q$	$\rightarrow q_0$	q_1	$\subset \overline{q_2} \supset$	$(\overline{q_3})$
a	q_1	q_1	q_2, q_3	$-$
b	q_0	q_0	$-$	q_3
c	q_0	q_2	$-$	q_1, q_2, q_3

When an automaton is given by a Cayley table, then the 0th line and the 0th column of the table are reserved for the states and for the alphabet, respectively (and it is marked in the 0th element of the 0th row).

The initial state should be the first among the states (it is advisable to mark it by a $\rightarrow$ sign also). The final states should also be marked, they should be circled.

The transition function is written into the table: the elements of the set $\delta(q, a)$ are written (if any) in the field of the column and row marked by the state q and by the letter a.

Representation — Graph

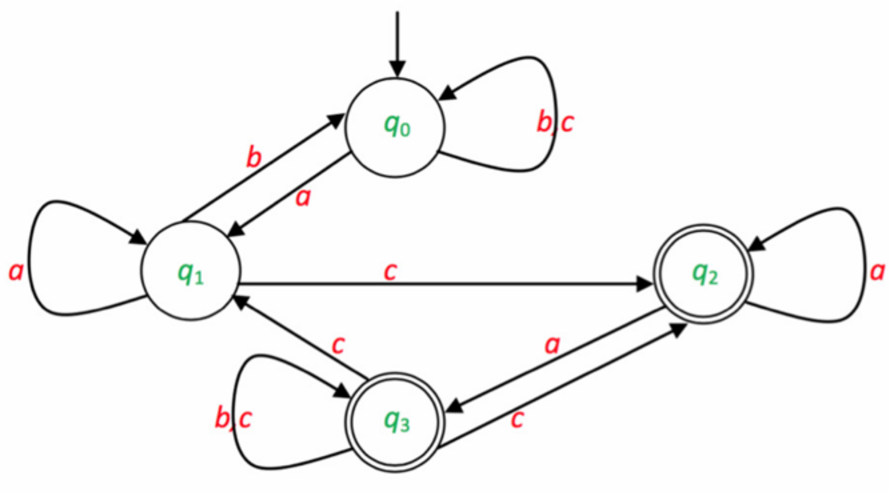


Figure 1: Finite automata graph representation

Figure 1 shows that automata can also be defined in a graphical way: let the vertices (nodes, that are drawn as circles in this case) of a graph represent the states of the automaton (we may write the names of the states into the circles). The initial state is marked by an arrow going to it not from a node. The accepting states are marked by double circles. The labeled arcs (edges) of the graph represent the transitions of the automaton. If $p \in \delta(q, a)$ for some $p, q \in Q, a \in T$, then there is an edge from the circle representing state q to the circle representing state p and this edge is labeled by a.

Language Accepted by Finite Automaton — Definition

Let $A = (Q, T, q_0, \delta, F)$ be an automaton and $w \in T^*$ be an input word. We say that w is accepted by A if there is a run of the automaton, i.e., there is an alternating sequence $q_0 t_1 q_1 \ldots q_{k-1} t_k q_k$ of states and transitions, that starts with the initial state q_0, ($q_i \in Q$ for every i, they are not necessarily distinct, e.g., $q_i = q_j$ is allowed even if $i \neq j$) and for every of its transition t_i of the sequence

- $t_i : q_i \in \delta\left(q_{i-1}, a_i\right)$ in nondeterministic cases,
- $t_i : q_i = \delta\left(q_{i-1}, a_i\right)$ in deterministic cases, where $a_1 \ldots a_k = w$, and $q_k \in F$. This run is called an accepting run.

All words that A accepts form $L(A)$, the language accepted (or recognized) by the automaton A.

Determinization of Finite Automata — Theorem

For every finite automaton there is an equivalent (completely defined) deterministic finite automaton.

Equivalence of Finite Automata and Regular Grammar

Every language generated by a regular grammar is accepted by a finite automaton.

Regular Languages

The language L is regular, if there exists a regular grammar G such that $L = L(G)$.

Regular Normal Form

Regular Grammars — Definition

A grammar $G = (N, T, S, P)$ is regular if each of its productions has one of the following forms: $A \to u, A \to uB$, where $A, B \in N, u \in T^*$. The languages that can be generated by regular grammars are the regular languages (they are also called type 3 languages of the Chomsky hierarchy).

Regular Normal Form — Definition

A grammar $G = (N, T, S, P)$ is regular if each of its productions has one of the following forms: $A \to a, A \to aB, S \to \lambda$, where $A, B \in N, a \in T$. The languages that can be generated by these grammars are the regular languages.

Context-Free Languages

The language L is context-free, if there exists a context-free grammar G such that $L = L(G)$.

The grammar G is called context-free, if all of its productions have a form

$$A \to p,$$

where $A \in N$ and $p \in V^*$.

Pushdown Automata

Definition

A pushdown automaton (PDA) is the following 7-tuple:

$$PDA = (Q, T, Z, q_0, z_0, \delta, F)$$

where

- Q is the finite nonempty set of the states,
- T is the set of the input letters (finite nonempty alphabet),
- Z is the set of the stack symbols (finite nonempty alphabet),
- q_0 is the initial state, $q_0 \in Q$,
- z_0 is the initial stack symbol, $z_0 \in Z$,
- δ is the transition function having a form

  $Q \times \{T \cup \{\lambda\}\} \times Z \to 2^{Q \times Z^*}$, and
- F is the set of the final states, $F \subseteq Q$.

Description

In order to understand the operating principle of the pushdown automaton, we have to understand the operations of finite automata and the stack memory. The stack is a LIFO (last in first out) memory, which has two operations, PUSH and POP. When we use the POP operation, we read the top letter of the stack, and to the same time we delete it. When we use the PUSH operation, we add a word to the top of the stack.

The pushdown automaton accepts words over the alphabet T. At the beginning the PDA is in state q_0, we can read the first letter of the input word, and the stack contains only z_0. In each step, we use the transition function to change the state and the stack of the PDA. The PDA accepts the input word, if and only if it can read the whole word, and it is in a final state when the end of the input word is reached.

More formally, in each step, the pushdown automaton has a configuration - also called instantaneous description - (q, v, w), where $q \in Q$ is the current state, $v \in T^*$ is the unread part of the input word, and $w \in Z^*$ is the whole word contained by the stack. At the beginning, the pushdown automaton is in its initial configuration: (q_0, p, z_0), where p is the whole input word. In each step, the pushdown automaton changes its configuration, while using the transition function. There are two different kinds of steps, the first is the standard, the second is the so called λ-step.

(1) The standard step is when the PDA reads its current state, current input letter, the top stack symbol, it finds an appropriate transition rule, it changes its state, it moves to the next input letter and changes the top symbol of the stack to the appropriate word. Formally, we can say the PDA can change its configuration from (q_1, av, zw) to (q_2, v, rw) in one step, if it has a transition rule $(q_2, r) \in \delta(q_1, a, z)$, where $q_1, q_2 \in Q, a \in T, z \in Z, v \in T^*$, $w \in Z^*$. Denote this transition $(q_1, av, zw) \vdash_{PDA} (q_2, v, rw)$.

(2) The λ-step is when the PDA reads its current state, it does not read any input letters, it reads the top stack symbol, it finds an appropriate transition rule, and it changes its state, it does not move to the next input letter and it changes the top letter of the stack to the given word. Formally, we can say again that the PDA can change its configuration from (q_1, v, zw) to (q_2, v, rw) in one step, if it has a transition rule $(q_2, r) \in \delta(q_1, \lambda, z)$, where $q_1, q_2 \in Q, z \in Z, v \in T^*$, and $w \in Z^*$. Denote this transition $(q_1, v, zw) \vdash_{PDA} (q_2, v, rw)$.

We can say that the PDA can change its configuration from (q_1, v, w) to (q_2, x, y) in finite many steps, if there are configurations $C_0, C_1, \ldots, C_n$ such that $C_0 = (q_1, v, w), C_n = (q_2, x, y)$, and $C_i \vdash_{PDA} C_{i+1}$ holds for each integer $0 \leq i < n$. Denote this transition $(q_1, v, w) \vdash^*_{PDA} (q_2, x, y)$.

The language accepted by the pushdown automaton

$L(PDA) = \{p \mid p \in T^*, (q_0, p, z_0) \vdash^*_{PDA} (q_f, \lambda, y), q_f \in F, y \in Z^*\}$

Acceptance by Empty Stack

There is another method for accepting words with a pushdown automaton. It is called "acceptance by empty stack". In this case, the automaton does not have any final states, and the word is accepted by the pushdown automaton if and only if it can read the whole word and the stack is empty when the end of the input word is reached. More formally:

Definition

The language accepted by automaton

$$PDA_e = (Q, T, Z, q_0, z_0, \delta)$$

by empty stack is

$$L\left(PDA_e\right) = \left\{p \mid p \in T^*, (q_0, p, z_0) \vdash^*_{PDA_e} (q, \lambda, \lambda), q \in Q\right\}.$$

Lecture review

1. Finite automata : definition

Definition (Finite automata)

Let $A = (Q, T, q_0, \delta, F)$. It is a finite automaton (recognizer), where Q is the finite set of (inner) states, T is the input (or tape) alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final (or accepting) states and δ is the transition function as follows.

- $\delta : Q \times T \to Q$ *(for deterministic finite automata);*
- $\delta : Q \times T \to 2^Q$ *(for nondeterministic finite automata).*

(1) Definition : $A = (Q, T, q_0, \delta, F)$

Q: set of all states

T: set of input letters of alphabet

q_0: initial state

δ: transition function

F: set of final states

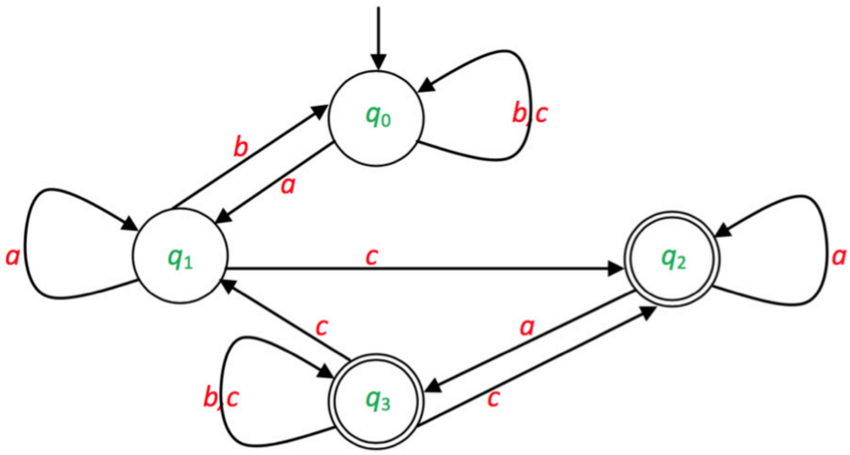
(2) example :

$A = (\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, q_0, \delta, \{q_2, q_3\})$, *where*
$\delta(q_0, a) = \{q_1\}$,
$\delta(q_0, b) = \{q_0\}$,
$\delta(q_0, c) = \{q_0\}$,
$\delta(q_1, a) = \{q_1\}$,
$\delta(q_1, b) = \{q_0\}$,
$\delta(q_1, c) = \{q_2\}$,
$\delta(q_2, a) = \{q_2, q_3\}$,
$\delta(q_3, b) = \{q_3\}$,
$\delta(q_3, c) = \{q_1, q_2, q_3\}$.

(3) representation – Cayley table

T \ Q	$\to q_0$	q_1	$\leftarrow q_2$	$\leftarrow q_3$
a	q_1	q_1	q_2, q_3	–
b	q_0	q_0	–	q_3
c	q_0	q_2	–	q_1, q_2, q_3

(4) representation – graph



2. Finite automata : deterministic v.s. non-deterministic

determistic (DFA) : one input , only one result .

non-determistic (NDFA) : one input, more than one result.

e.g. $\delta(q_2, a) = \{q_2, q_3\}$ NDFA

3. Finite automata : partially defined v.s. fully defined

patially defined : get stuck when no δ for the actual state and letter

fully defined : otherwise

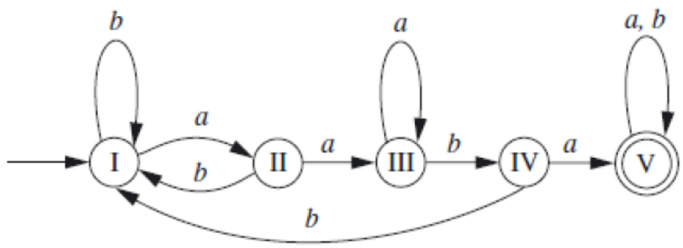
Finite automata – description

One can imagine a finite automaton as a machine equipped with an input tape. The machine works on a discrete time scale. At every point of time the machine is in one of its states, then it reads the next letter on the tape (the letter under the reading head), and then, according to the transition function (depending on the actual state and the letter being read,) it goes to the next state. It may happen in some variations that there is no transition defined for the actual state and letter, then the machine gets stuck and cannot continue its run. This case the automaton is called **partially defined** finite automaton. Otherwise the automaton is called **completely defined** finite automaton.

Practice

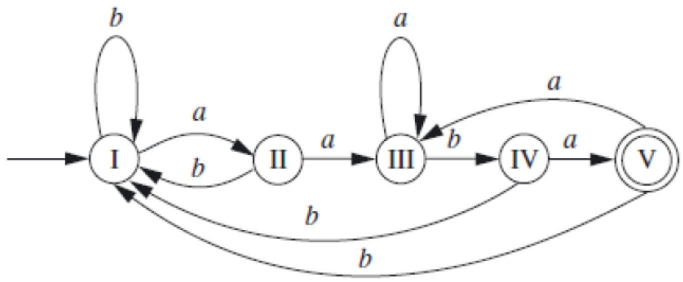
Set 1.

For each of the FAs pictured in Fig. 2.43, give a simple verbal description of the language it accepts.



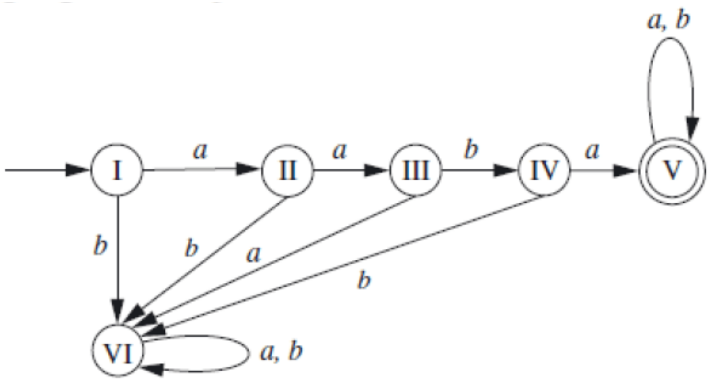
(a)

(a) ... aaba ...



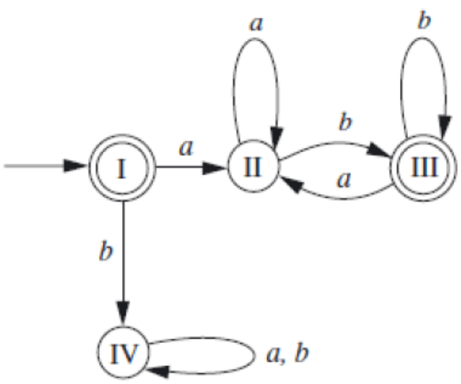
(b)

(b) ... aaba

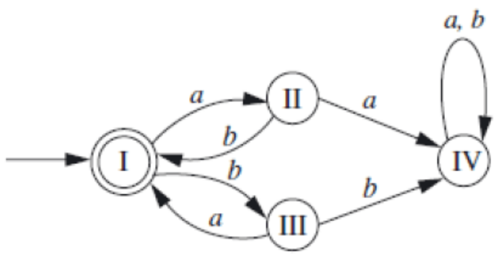


(c)

(c) aaba ...



(d)



(e)

(d) λ, a ... b

(e) $(ab, ba)^*$

$= \{\lambda, ab, ba, abab, \ldots, baba, \ldots,$
$abba, baab, abbaab, \ldots\}$

Set 2: In each part below, draw an FA accepting the indicated language over $\{a, b\}$.

a. The language of all strings containing exactly two a's.

b. The language of all strings containing at least two a's.

c. The language of all strings that do not end with ab.

d. ~~The language of all strings that begin or end with aa or bb.~~

e. The language of all strings not containing the substring aa.

f. The language of all strings in which the number of a's is even.

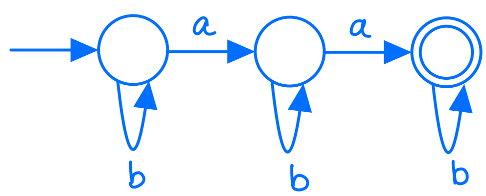
g. The language of all strings in which both the number of a's and the number of b's are even.

h. The language of all strings containing no more than one occurrence of the string aa. (The string aaa contains two occurrences of aa.)

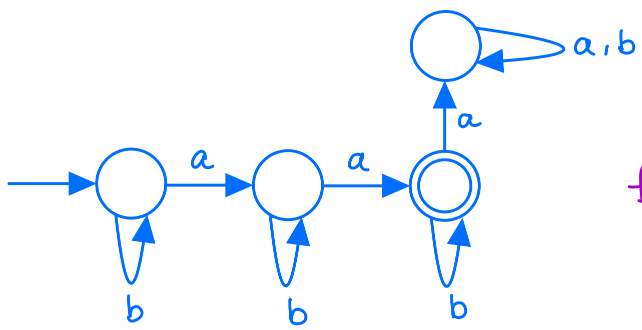
i. The language of all strings in which every a (if there are any) is followed immediately by bb.

j. The language of all strings containing both bb and aba as substrings.

a. containing exactly two a :

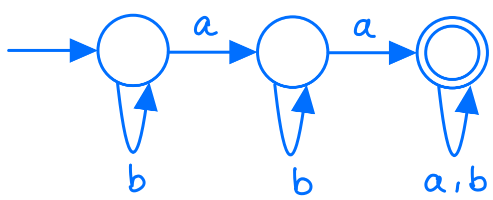
① 

partially defined

② 

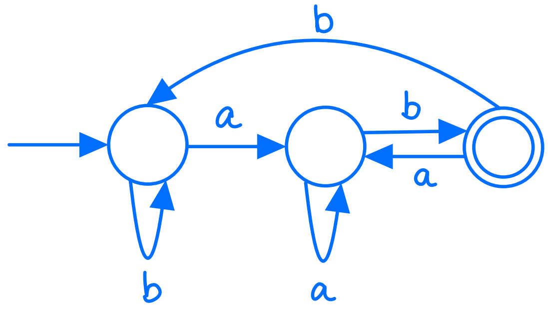
fully defined

b. containing at least two a :

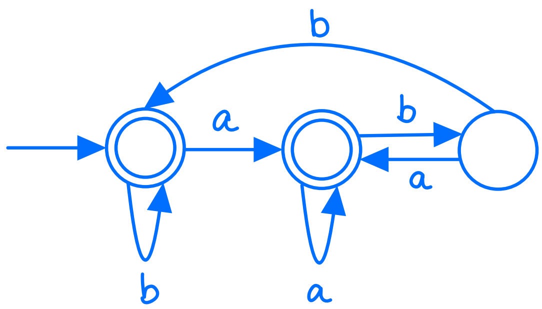


c. do not end with ab :

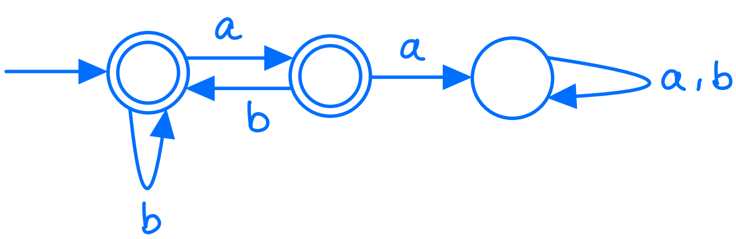
step 1 : create the opposite : end with ab



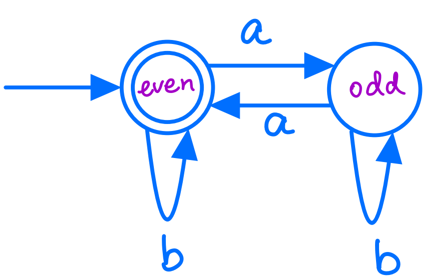
Step 2 : take the opposite states



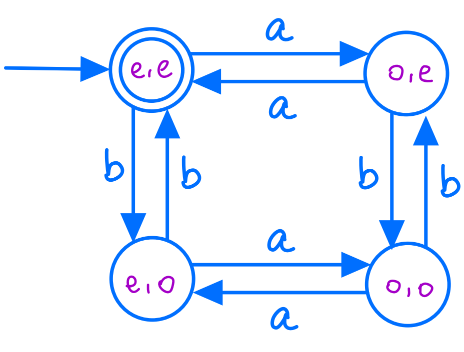
e. not containing the substring aa :



f. the number of 'a' is even :

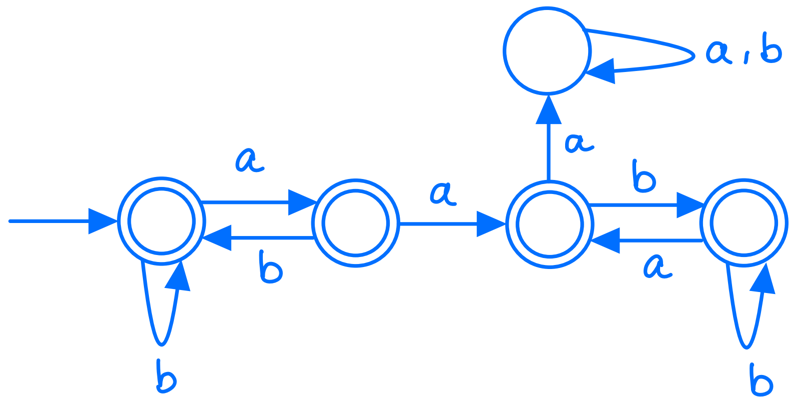


g. the numbers of 'a' and 'b' are even :

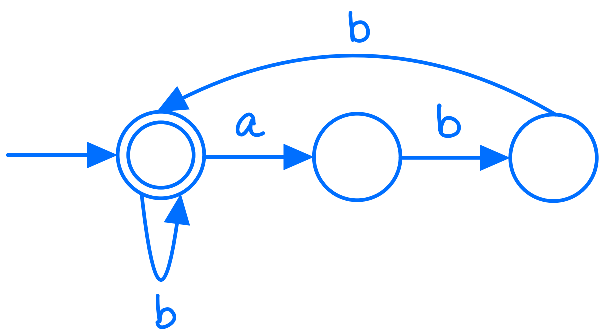


* h. no more than one occurrence of the string aa :

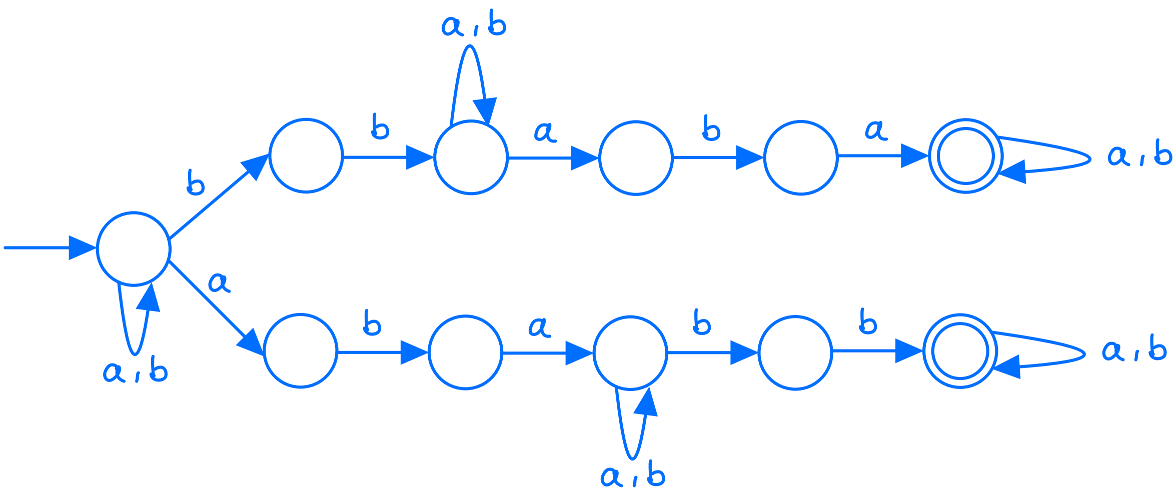
λ , b , aaa , babaabaa , ...



i. every a is followed immediately by bb :



* j. containing both bb and aba as substrings :



␣bb␣aba␣

␣aba␣bb␣

Set 3. For each of the following languages, draw an FA accepting it.

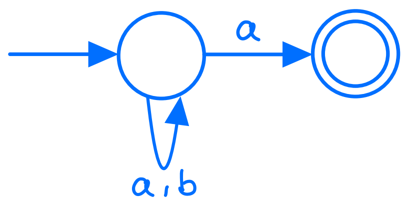
a. $\{a, b\}^*\{a\}$

b. $\{bb, ba\}^*$

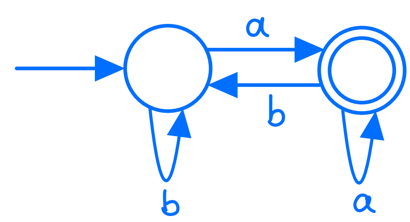
c. $\{a, b\}^*\{b, aa\}\{a, b\}^*$

d. $\{bbb, baa\}^*\{a\}$

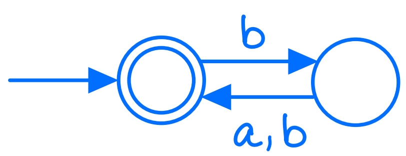
a. $\{a, b\}^*\{a\}$



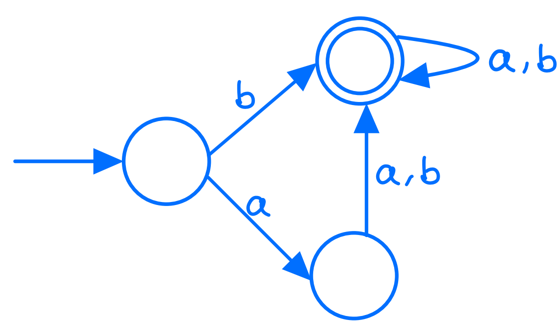
or



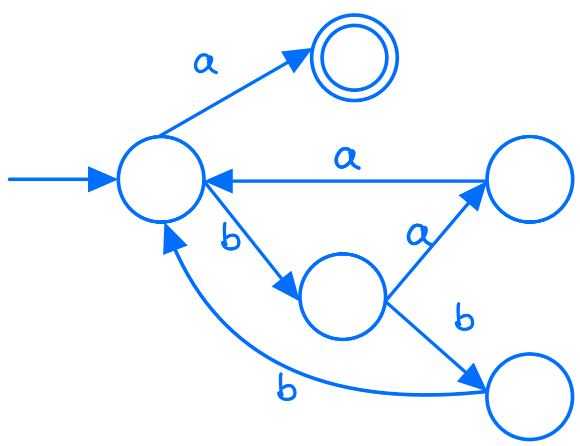
b. $\{bb, ba\}^*$



c. $\{a, b\}^*\{b, aa\}\{a, b\}^*$



d. $\{bbb, baa\}^*\{a\}$



Lecture review (ppt : ICS 7)

1. Definition

Finite automata can accept regular languages, so we have to extend its definition so as it could accept context-free languages. The solution for this problem is to add a stack memory to a finite automaton, and the name of this solution is "pushdown automaton".

• Type 2: context-free grammar

all the production rules should satisfy the following forms:

$A \to p$, $A \in N$ and $p \in V^*$

finite automata: $A = (Q, T, q_0, \delta, F)$

Definition

A pushdown automaton (PDA) is the following 7-tuple:

$$PDA = (Q, T, Z, q_0, z_0, \delta, F)$$

NEW

where

- Q *is the finite nonempty set of the states,*
- T *is the set of the input letters (finite nonempty alphabet),*
- Z *is the set of the stack symbols (finite nonempty alphabet),*
- q_0 *is the initial state,* $q_0 \in Q$,
- z_0 *is the initial stack symbol,* $z_0 \in Z$,
- δ *is the transition function having a form* $Q \times \{T \cup \{\lambda\}\} \times Z \to 2^{Q \times Z^*}$, *and*
- F *is the set of the final states,* $F \subseteq Q$.

~~studied them through many hours~~. The stack is a LIFO (last in first out) memory, which has two operations, PUSH and POP. When we use the POP operation, we read the top letter of the stack, and at the same time we delete it. When we use the PUSH operation, we add a word to the top of the stack.

We can say that the PDA can change its configuration from (q_1, v, w) to (q_2, x, y) in finite many steps, if there are configurations $C_0, C_1, \ldots, C_n$ such that $C_0 = (q_1, v, w)$, $C_n = (q_2, x, y)$, and $C_i \vdash_{PDA} C_{i+1}$ holds for each integer $0 \le i < n$. Denote this transition $(q_1, v, w) \vdash^*_{PDA} (q_2, x, y)$.

"$\vdash$" Turnstile, referred to as "Tee", often read as "yields", "proves"

Definition

$L(PDA) = \{p \mid p \in T^*, (q_0, p, z_0) \vdash^*_{PDA} (q_f, \lambda, y), q_f \in F, y \in Z^*\}.$

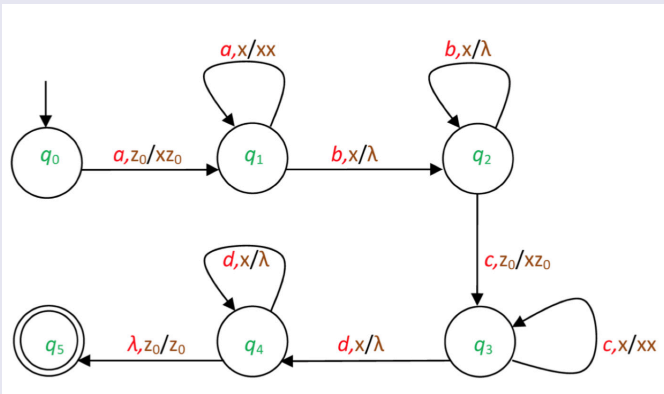
The pushdown automaton accepts words over the alphabet T. At the beginning the PDA is in state q_0, we can read the first letter of the input word, and the stack contains only z_0. In each step, we use the transition function to change the state and the stack of the PDA. The PDA accepts the input word, if and only if it can read the whole word, and it is in a final state when the end of the input word is reached.

2. Example:

Example

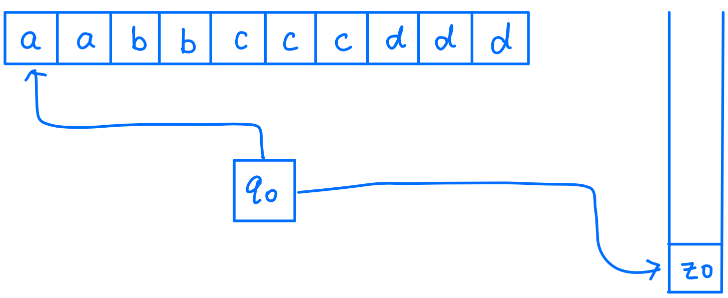
This simple example shows the description of a pushdown automaton which accepts the language $L = \{a^i b^i c^j d^j \mid i, j \geq 1\}$.

$PDA = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b, c, d\}, \{x, z_0\}, q_0, z_0, \delta, \{q_5\}),$
$\delta(q_0, a, z_0) = \{(q_1, xz_0)\},$
$\delta(q_1, a, x) = \{(q_1, xx)\},$
$\delta(q_1, b, x) = \{(q_2, \lambda)\},$
$\delta(q_2, b, x) = \{(q_2, \lambda)\},$
$\delta(q_2, c, z_0) = \{(q_3, xz_0)\},$
$\delta(q_3, c, x) = \{(q_3, xx)\},$
$\delta(q_3, d, x) = \{(q_4, \lambda\},$
$\delta(q_4, d, x) = \{(q_4, \lambda)\},$
$\delta(q_4, \lambda, z_0) = \{(q_5, z_0)\}.$



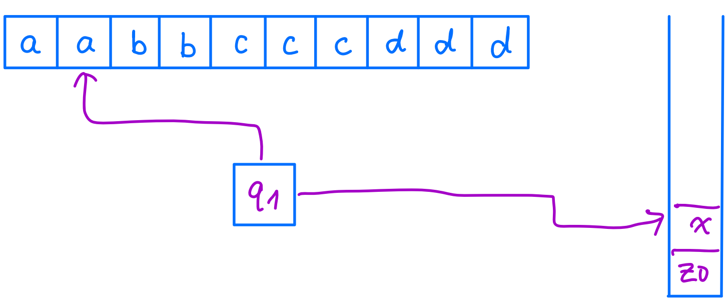
read the word: aabbcccddd

initial state:

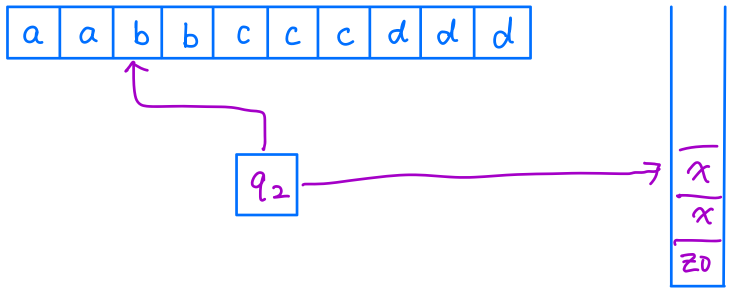


pop z_0 push $x z_0$

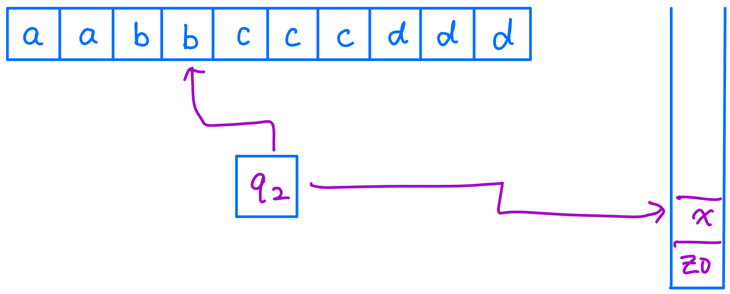
read $\delta(q_0, a, z_0) = \{(q_1, x z_0)\}$



read $\delta(q_1, a, \overset{\text{pop } x}{\underline{x}}) = \{(q_2, \overset{\text{push } xx}{\underline{xx}})\}$

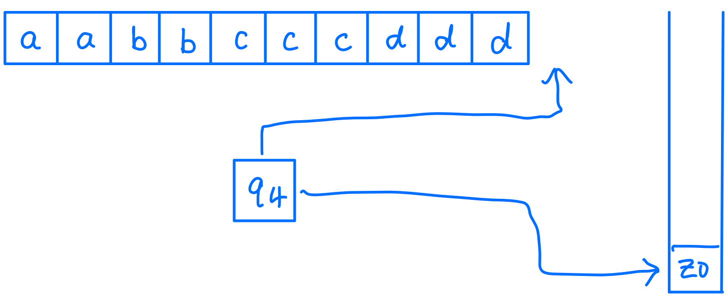


read $\delta(q_2, b, \overset{\text{pop } x}{\underline{x}}) = \{(q_2, \overset{\text{push nothing}}{\underline{\lambda}})\}$



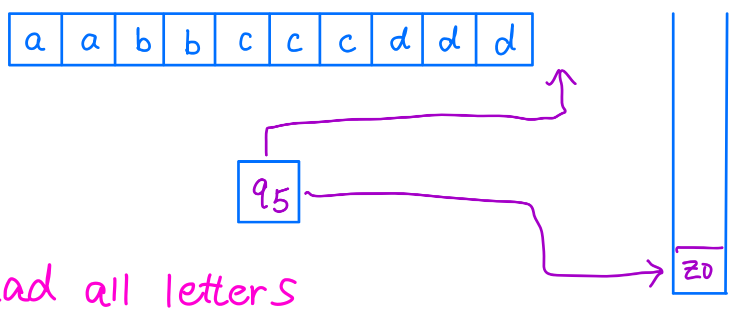
omit some steps :

	q_2	q_2	q_3	q_3	q_3	q_4	q_4	q_4	$q_5 \in F$
...	b	c	c	c	d	d	d	λ	
Z_0	$\cancel{x}$	λ	$\cancel{x}$	$\cancel{x}$	$\cancel{x}$	λ	λ	λ	



read $\delta(q_4, \lambda, Z_0) = \{(q_5, Z_0)\}$

final state :



accepted $\begin{cases} \text{read all letters} \\ q_5 \in F \end{cases}$

Practice

1. create PDA for $L = \{a^i b^i \mid i \geq 1\}$

$PDA = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{x, z_0\}, q_0, z_0, \delta, \{q_3\})$

$\delta(q_0, a, z_0) = \{(q_1, x z_0)\}$

$\delta(q_1, a, x) = \{(q_1, x x)\}$ repeat a

$\delta(q_1, b, x) = \{(q_2, \lambda)\}$

$\delta(q_2, b, x) = \{(q_2, \lambda)\}$ repeat b

$\delta(q_2, \lambda, z_0) = \{(q_3, z_0)\}$

2. $L = \{a^i b^j \mid i, j \geq 1, i > j\}$ e.g. aab, aaab, aaabb, ...

$PDA = (\{q_0, \dots, q_4\}, \{a, b\}, \{x, z_0\}, q_0, z_0, \delta, \{q_4\})$

$\delta(q_0, a, z_0) = \{(q_1, x z_0)\}$

$\delta(q_1, a, x) = \{(q_2, x x)\}$ we need at least 2 "a"

$\delta(q_2, a, x) = \{(q_2, x x)\}$ repeat a

$\delta(q_2, b, x) = \{(q_3, \lambda)\}$ at least 1 "b"

$\delta(q_3, b, x) = \{(q_3, \lambda)\}$ repeat b

$\delta(q_3, \lambda, x) = \{(q_4, x)\}$ $q_4 \in F$

The number of a is more than that of b, x should still exist in the stack when finishing reading all letters. q_4 should be the only final state.

3. $L = \{a^i b^{2i} \mid i \geq 1\}$

easiest idea: when we read one a, we push "xx" to the stack

$PDA = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{x, z_0\}, q_0, z_0, \delta, \{q_3\})$

$\delta(q_0, a, z_0) = \{(q_1, xxz_0)\}$

$\delta(q_1, a, x) = \{(q_1, xxx)\}$ repeat a

$\delta(q_1, b, x) = \{(q_2, \lambda)\}$

$\delta(q_2, b, x) = \{(q_2, \lambda)\}$ repeat b

$\delta(q_2, \lambda, z_0) = \{(q_3, z_0)\}$

Method 2: when we delete b, first we change x to y.
then we delete.

$PDA = (\{q_0, \dots, q_3\}, \{a, b\}, \{x, y, z_0\}, q_0, z_0, \delta, \{q_3\})$

$\delta(q_0, a, z_0) = \{(q_1, xz_0)\}$

$\delta(q_1, a, x) = \{(q_1, xx)\}$ repeat a

$\delta(q_1, b, x) = \{(q_2, y)\}$

$\delta(q_2, b, y) = \{(q_2, \lambda)\}$

$\delta(q_2, b, x) = \{(q_2, y)\}$

$\delta(q_2, \lambda, z_0) = \{(q_3, z_0)\}$

4. $L = \{ a^{2i} b^{i} \mid i \geq 1 \}$ every time we can only delete one letter.

Method 1: when we read the second a, we change the input x to y.

$PDA = (\{q_0, \dots, q_3\}, \{a, b\}, \{x, y, z_0\}, q_0, z_0, \delta, \{q_3\})$

$\delta(q_0, a, z_0) = \{(q_1, x z_0)\}$

$\delta(q_1, a, x) = \{(q_1, y)\}$

$\delta(q_1, a, y) = \{(q_1, xy)\}$

$\delta(q_1, b, y) = \{(q_2, \lambda)\}$

$\delta(q_2, b, y) = \{(q_2, \lambda)\}$

$\delta(q_2, \lambda, z_0) = \{(q_3, z_0)\}$

Method 2: we input one x only after reading two a.

$PDA = (\{q_0, \dots, q_5\}, \{a, b\}, \{x, y, z_0\}, q_0, z_0, \delta, \{q_5\})$

$\delta(q_0, a, z_0) = \{(q_1, z_0)\}$

$\delta(q_1, a, z_0) = \{(q_2, x z_0)\}$

$\delta(q_2, a, x) = \{(q_3, x)\}$

$\delta(q_3, a, x) = \{(q_2, xx)\}$

$\delta(q_2, b, x) = \{(q_4, \lambda)\}$

$\delta(q_4, b, x) = \{(q_4, \lambda)\}$

$\delta(q_4, \lambda, z_0) = \{(q_5, z_0)\}$

aaaabb:

States:	q_0	q_1	q_2	q_3	q_2	q_4	q_4
letters:	a	a	a	a	b	b	
Stack:	z_0	-	x	-	x	λ	λ