cse303
ELEMENTS OF THE THEORY OF COMPUTATION

Professor Anita Wasilewska

LECTURE 5

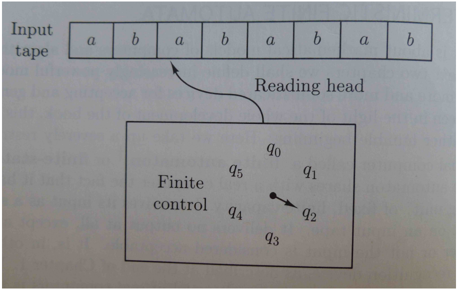
CHAPTER 2
FINITE AUTOMATA

1. Deterministic Finite Automata DFA
2. Nondeterministic Finite Automata NDFA
3. Finite Automata and Regular Expressions
4. Languages that are Not Regular
5. State Minimization

CHAPTER 2
PART 1: Deterministic Finite Automata DFA

Simple Computational Model

Here is a picture



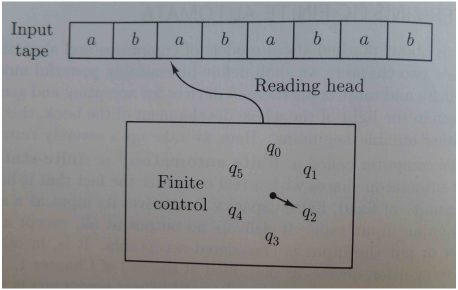
Here are the **components** of the model

C1: Input string on an input tape written at the beginning of the tape

The input tape is divided into squares, with **one symbol** inscribed in each tape square

DFA - A Simple Computational Model

Here is a picture



C2: "Black Box" - called **Finite Control**

It can be in any specific time in **one** of the finite number of **states** $\{q_1, \ldots, q_n\}$

C3: A movable **Reading Head** can sense what symbol is written in any position on the input tape and **moves** only **one square** to the right

DFA - A Simple Computational Model

Here are the **assumptions** for the model

A1: There is no output at all;

A2: DFA **indicates** whether the input is acceptable or not acceptable

A3: DFA is a language **recognition** device

DFA - A Simple Computational Model

Operation of DFA

O1 Initially the **reading head** is placed at left most square at the beginning of the tape and

O2 **finite control** is set on the **initial state**

O3 After reading on the input symbol the **reading head** moves one square to the right and enters a new state

O4 The process is **repeated**

O5 The process **ends** when the reading head reaches the end of the tape

DFA - A Simple Computational Model

The general rules of the operation of DFA are

R1 At regular intervals DFA **reads** only one symbol at the time from the input tape and **enters** a new state

R2: The **move** of DFA depends only on the **current** state and the **symbol** just read

DFA - A Simple Computational Model

Operation of DFA

O6 When the process **stops** the DFA indicates its approval or disapproval of the string by means of the **final state**

O7 If the process **stops** while being in the **final state**, the string is accepted

O8 If the process **stops** while not being in the **final state**, the string is not accepted

Language Accepted by DFA

Informal Definition

Language accepted by a Deterministic Finite Automata is equal to the set of strings accepted by it

DFA - Mathematical Model

To build a mathematical model for DFA we need to include and define the following components

FINITE set of STATES

ALPHABET Σ

INITIAL state

FINAL state

Description of the MOVE of the reading **head** is as follows

R1 At regular intervals DFA **reads** only one symbol at the time from the input tape and **enters** a new state

R2: The MOVE of DFA depends **only** on the current state and the symbol just **read**

DFA - Mathematical Model

Definition

A Deterministic Finite Automata is a quintuple

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

where

K is a finite set of **states**

Σ as an **alphabet**

$s \in K$ is the **initial state**

$F \subseteq K$ is the set of **final states**

δ is a function

$$\delta : \ K \times \Sigma \ \longrightarrow \ K$$

called the **transition function**

We usually use different symbols for $K,\ \Sigma$, i.e. we have that $K \cap \Sigma = \emptyset$

DFA Definition

Definition revisited

A Deterministic Finite Automata is a quintuple

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

where

K is a finite set of **states**

$K \neq \emptyset$ because $s \in K$

Σ as an **alphabet**

Σ can be $\emptyset$ - case to consider

$s \in K$ is the **initial state**

$F \subseteq K$ is the set of **final states**

F can be $\emptyset$ - case to consider

δ is a function

$$\delta : \quad K \times \Sigma \quad \longrightarrow \quad K$$

called the **transition function**

Transition Function

Given DFA

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

where

$$\delta :\quad K \times \Sigma \quad \longrightarrow \quad K$$

Let

$$\delta(q, \sigma) = q' \quad \text{for} \quad q,\ q' \in K, \quad \sigma \in \Sigma$$

means: the automaton M in the state q **reads** $\sigma \in \Sigma$ and **moves** to a state $q' \in K$, which is uniquely determined by state q and σ just **read**

Configuration

In order to define a notion of computation of M on an input string $w \in \Sigma^*$ we introduce first a notion of a **configuration**

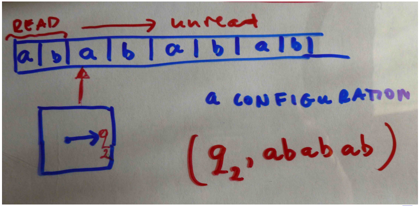
Definition

A configuration is any tuple

$$(q, w) \in K \times \Sigma^*$$

where $q \in K$ represents a **current** state of M and $w \in \Sigma^*$ is **unread part** of the input

Picture



Transition Relation

Definition

The set of all possible configurations of $M = (K, \Sigma, \delta, s, F)$ iis just

$$K \times \Sigma^* = \{(q, w) : \quad q \in K, \quad w \in \Sigma^*\}$$

We **define** move of an automaton M i in terms of a **transition relation**

$$\vdash_M$$

The **transition relation** acts between two **configurations** and hence $\vdash_M$ is a certain binary relation defined on $K \times \Sigma^*$, i.e.

$$\vdash_M \subseteq (K \times \Sigma^*)^2$$

Formal definition follows

Transition Relation

Definition

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

A binary relation

$$\vdash_M\ \subseteq\ (K \times \Sigma^*)^2$$

is called a **transition relation** when for any

$q,\ q' \in K,\ \ w_1, w_2 \in \Sigma^*$ the following holds

$$(q, w_1) \vdash_M (q', w_2)$$

if and only if

1. $w_1 = \sigma w_2$, for some $\sigma \in \Sigma$ (M **looks** at σ)
2. $\delta(q, \sigma) = q'$ (M **moves** from q to q' reading σ in w_1)

Transition Relation

Definition (Transition relation short definition)

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

For any $q,\ q' \ \in K, \quad \sigma \in \Sigma, \quad w \in \Sigma^*$

$$(q, \sigma w) \vdash_M (q', w)$$

if and only if

$$\delta(q, \sigma) = q'$$

Idea of Computation

We use the transition relation to define a move of M along a given input, i.e. a given $w \in \Sigma^*$

Such a move is called a **computation**

Example

Given M such that $K = \{s,\ q\}$ and let $\vdash_M$ be a transition relation such that

$$(s, aab) \vdash_M (q, ab) \vdash_M (s, b) \vdash_M (q, e)$$

We call a **sequence** of configurations

$$(s,\ aab),\ \ (q, ab),\ \ (s, b),\ \ (q, e)$$

a **computation** from (s, aab) to (q, e) in automaton M

Idea of Computation

Given a a **computation**

$$(s,\ aab),\ \ (q, ab),\ \ (s, b),\ \ (q, e)$$

We write this **computation** in a more general form as

$$(q_1, aab),\ \ (q_2, ab),\ \ (q_3, b),\ \ (q_4, e)$$

for $q_1,\ q_2,\ q_3,\ q_4$ being a specific **sequence of states** from $K = \{s,\ q\}$, namely $q_1 = s,\ q_2 =,\ q_3 = s,\ q_4 = q$ and say that the **length** of this computation is 4

In general we write any **computation of length** 4 as

$$(q_1, w_1),\ \ (q_2, w_2),\ \ (q_3, w_3),\ \ (q_4, w_4)$$

for any **sequence** $q_1,\ q_2,\ q_3,\ q_4$ of states from K and words $w_i \in \Sigma^*$

Idea of the Computation

Example

Given M and the **computation**

$$(s, aab), \quad (q, ab), \quad (s, b), \quad (q, e)$$

We say that the word w= aab is **accepted** by M if and only if

1. the **computation** starts when M is in the initial state

- true here as s denotes the **initial state**

2. the whole word w has been read, i.e. the last configuration of the computation is (q, e) for certain state in K,

- true as $K = \{s, q\}$

3. the **computation** ends when M is in the **final state**

- true only if we have that $q \in F$

Otherwise the word w is **not accepted** by M

Definition of the Computation

Definition

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

A sequence of **configurations**

$$(q_1, w_1),\ (q_2, w_2),\ \ldots, (q_n, w_n), \qquad n \geq 1$$

is a computation of the **length** n in M from (q, w) to (q', w')

if and only if

$$(q_1, w_1) = (q, w), \qquad (q_n, w_n) = (q', w') \quad \text{and}$$

$$(q_i, w_i) \vdash_M (q_{i+1}, w_{i+1}) \quad \text{for} \quad i = 1, 2, \ldots n-1$$

Observe that when $n = 1$ the computation (q_1, w_1) **always** exists . It is a computation of the length 1, called also a trivial computation

We also write sometimes the computations as

$$(q_1, w_1) \vdash_M (q_2, w_2) \vdash_M \ \ldots \ \vdash_M (q_n, w_n) \quad \text{for} \quad n \geq 1$$

Definition of the Computation

Given a computations

$$(q_1, w_1) \vdash_M (q_2, w_2) \vdash_M \ldots \vdash_M (q_n, w_n) \quad \text{for} \quad n \geq 1$$

In the case $n = 1$, we get only **one** configuration (q_1, w_1)

It is a computation of length 1

It is a **ZERO STEP** computation, as we have **zero** applications of the transition relation $\vdash_M$

In the case $n = 2$ (length 2) we get

$$(q_1, w_1) \vdash_M (q_2, w_2)$$

It is a **ONE STEP** computation as we have **one** application of the transition relation $\vdash_M$

In the case $n = 3$ (length 3) , we get

$$(q_1, w_1) \vdash_M (q_2, w_2) \vdash_M (q_3, w_3)$$

It is a **TWO STEPS** computation as we have **two** applications of the transition relation $\vdash_M$, etc, etc...

Words Accepted by M

Definition

A word $w \in \Sigma^*$ is **accepted** by $M = (K, \Sigma, \delta, s, F)$ if and only if **there is** a computation

$$(q_1, w_1), (q_2, w_2), \ldots, (q_n, w_n)$$

such that $q_1 = s$, $w_1 = w$, $w_n = e$ and $q_n = q \in F$

We re-write it as

A word $w \in \Sigma^*$ is **accepted** by $M = (K, \Sigma, \delta, s, F)$ if and only if **there is** a computation

$$(s, w), (q_2, w_2), \ldots, (q, e) \quad \text{and} \quad q \in F$$

When the computation is such that $q \notin F$ we say that the word w is **not accepted** (rejected) by M

Words Accepted by M

In Plain Words:

A word $w \in \Sigma^*$ is **accepted** by $M = (K, \Sigma, \delta, s, F)$

if and only if

there is a computation such that

1. starts with the word w and M in the **initial state** ,
2. ends when M is in a **final state**, and
3. the whole word w has been read

Language Accepted by M

Definition

We define the language **accepted** by M as follows

$$L(M) = \{w \in \Sigma^* : \quad w \text{ is accepted by } M\}$$

i.e. we write

$$L(M) = \{w \in \Sigma^* : (s, w) \vdash_M \dots \vdash_M (q, e) \text{ for some } q \in F\}$$

Examples

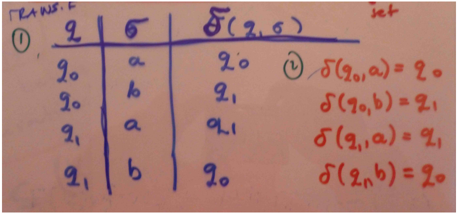
Example 1

Let $M = (K,\ \Sigma,\ \delta,\ s,\ F)$, where

$K = \{q_0, q_1\}, \quad \Sigma = \{a,\ b\}, \quad s = q_0, \quad F = \{q_0\}$

and the **transition function** $\delta :\ K \times \Sigma \ \longrightarrow\ K$

is defined as follows



Question Determine whether $ababb \in L(M)$ or $ababb \notin L(M)$

Examples

Solution

We must evaluate computation that starts with the configuration $(q_0, ababb)$ as $q_0 = s$

$(q_0, ababb) \vdash_M$ use $\delta(q_0, a) = q_0$

$(q_0, babb) \vdash_M$ use $\delta(q_0, b) = q_1$

$(q_1, abb) \vdash_M$ use $\delta(q_1, a) = q_1$

$(q_1, bb) \vdash_M$ use $\delta(q_1, b) = q_0$

$(q_0, b) \vdash_M$ use $\delta(q_0, b) = q_1$

$(q_1, e) \vdash_M$ **end** of computation and $q_1 \notin F = \{q_0\}$

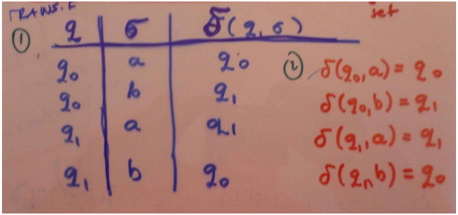
We proved that $ababb \notin L(M)$

Observe that we always get **unique** computations, as δ is a function, hence he name Deterministic Finite Automaton (DFA)

Example 2

Let $M_1 = (K,\ \Sigma,\ \delta,\ s,\ F)$ for all components defined as in M from **Example 1**, except that we take now
$F = \{q_0, q_1\}$

We remind that



Exercise Show that now $ababb \in L(M_1)$

Language Accepted by M
Revisited

We have defined the language **accepted** by M as

$$L(M) = \{w \in \Sigma^* : (s, w) \vdash_M \ldots \vdash_M (q, e) \text{ for some } q \in F\}$$

The question is now- how to write it in a more concise and elegant way

Answer: use the notion (Chapter 1, Lecture 3) of reflexive, transitive closure of $\vdash_M$ denoted by $\vdash_M{}^*$ and now we write

Definition

$$L(M) = \{w \in \Sigma^* : (s, w) \vdash_M{}^* (q, e) \text{ for some } q \in F\}$$

We write it also using the **existential quantifier** symbol as

$$L(M) = \{w \in \Sigma^* : \exists_{q \in F} ((s, w) \vdash_M{}^* (q, e))\}$$

Language Accepted by M Revisited

In order to justify the following I **definition**

$$L(M) = \{w \in \Sigma^* : \ (s, w) \vdash_M{}^* (q, e) \quad \text{for some} \quad q \in F\}$$

We bring back the general notion of a **path** in a binary relation R and its reflexive, transitive closure R^* (Chapter 1)

It follows **directly** from these definitions that

$$(q_1, w_1) \vdash_M{}^* (q_n, w_n)$$

represents a **path**

$$(q_1, w_1), \ (q_2, w_2) \ \ \ldots, \ \ (q_{n-1}, w_{n-1}, \ (q_n, w_n)$$

in the relation $\vdash_M$, which is defined as a **computation**

$$(q_1, w_1) \vdash_M (q_2, w_2) \ \ \ldots, \ \ (q_{n-1}, w_{n-1} \vdash_M (q_n, w_n)$$

in M from (q_1, w_1) to (q_n, w_n)

Language Accepted by M Revisited

Hence

$$(s, w) \vdash_M{}^* (q, e)$$

represent a computation

$$(s, w) \vdash_M (q_1, w_1), \quad \ldots, \quad (q_n, w_n) \vdash_M (q, e)$$

from (s, w) to (q, e),

So define the language L(M) as

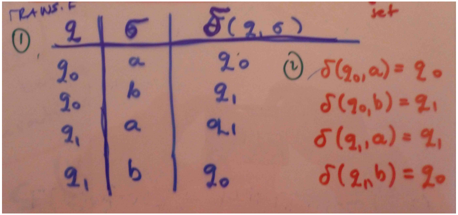
$$L(M) = \{w \in \Sigma^* : \ (s, w) \vdash_M{}^* (q, e) \ \text{ for some } \ q \in F\}$$

Example

Let $M = (K, \Sigma, \delta, s, F)$ be automaton from our **Example 1**, i.e. we have

$K = \{q_0, q_1\}, \quad \Sigma = \{a, b\}, \quad s = q_0, \quad F = \{q_0\}$

and the **transition function** $\delta : K \times \Sigma \longrightarrow K$ is defined as follows



Question Show that $aabba \in L(M)$

Example

We evaluate

$$(q_0, aabba) \vdash_M (q_0, abba) \vdash_M (q_0, bba) \vdash_M$$

$$(q_1, ba) \vdash_M (q_0, a) \vdash_M (q_0, e) \quad \text{and} \quad q_0 = s, \quad q_0 \in F = \{q_0\}$$

This proves that

$$(s, aabba) \vdash_M{}^* (q_0, e) \quad \text{for} \quad q_0 \in F$$

By definition

$$aabba \in L(M)$$

General remark

To **define** or to give an example of

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

means that one has to **specify all** its components $K,\ \Sigma,\ \delta,\ s,\ F$

We usually use different symbols for $K,\ \Sigma$, i.e. we have that $K \cap \Sigma = \emptyset$

Exercise

Given $\Sigma = \{a,\ b\}$ and $K == \{q_0,\ q_1\}$

1. Define 3 automata M

2. Define an automaton M, such that $L(M) = \emptyset$

3. How many automata M can one define?

Exercise

1. Here are 3 automata $M_1 - M_3$

M₁ : $M_1 = (\ K = \{q_0,\ q_1\},\ \Sigma = \{a,\ b\},\ \ \delta,\ s = q_0,\ F = \{q_0\})$

$\delta(q_0, a) = q_0,\ \ \delta(q_0, b) = q_0,\ \ \delta(q_1, a) = q_0,\ \ \delta(q_1, b) = q_0$

M₂ : $M_2 = (\ K = \{q_0,\ q_1\},\ \Sigma = \{a,\ b\},\ \ \delta,\ s = q_0,\ F = \{q_1\})$

$\delta(q_0, a) = q_0,\ \ \delta(q_0, b) = q_0,\ \ \delta(q_1, a) = q_0,\ \ \delta(q_1, b) = q_1$

M₃ : $M_3 = (\ K = \{q_0,\ q_1\},\ \Sigma = \{a,\ b\},\ \ \delta,\ s = q_0,\ F = \{q_1\})$

$\delta(q_0, a) = q_0,\ \ \delta(q_0, b) = q_1,\ \ \delta(q_1, a) = q_1,\ \ \delta(q_1, b) = q_0$

Exercise

2. Define an automaton M, such that $L(M) = \emptyset$

Answer: The automata M_2 is such that $L(M_2) = \emptyset$ as there is no computation that would **start at initial state** q_0 and **end in the final state** q_1 as in M_2 we have that $\delta(q_0, a) = q_0$, $\delta(q_0, b) = q_0$, so we will never reach the **final state** q_1

Here is another example:

Let $\mathbf{M_4}$ be defined as follows

$M_4 = (\ K = \{q_0,\ q_1\},\ \Sigma = \{a,\ b\},\ \delta,\ s = q_0,\ F = \emptyset)$

$\delta(q_0, a) = q_0$, $\delta(q_0, b) = q_0$, $\delta(q_1, a) = q_0$, $\delta(q_1, b) = q_0$

$L(M_4) = \emptyset$ as there is no computation that would **start** at initial state q_0 and **end** in the final state as there is no final state

Exercise

3. How many automata M can one define?

Observe that all of M must have $\Sigma = \{a, b\}$ and $K == \{q_0, q_1\}$ so they **differ** on the choices of $\delta : K \times \Sigma \longrightarrow K$

By **Counting Functions Theorem** we have 2^4 possible choices for δ

They also can **differ** on the choices of **final states** F

There as many choices for final states as subsets of $K == \{q_0, q_1\}$, i.e. $2^2 = 4$

Additionally we have to count all combinations of choices of δ with choices of F

Challenge

1. Define an automata M with $\Sigma \neq \emptyset$ such that $L(M) = \emptyset$
2. Define an automata M with $\Sigma = \emptyset$ such that $L(M) \neq \emptyset$
3. Define an automata M with $\Sigma \neq \emptyset$ such that $L(M) \neq \emptyset$
4. Define an automata M with $\Sigma \neq \emptyset$ such that $L(M) = \Sigma^*$
5. Prove that there always exist an automata M such that $L(M) = \Sigma^*$

DFA State Diagram

As we could see the transition functions can be defined in many ways but it is difficult to decipher the workings of the automata they define from their mathematical definition

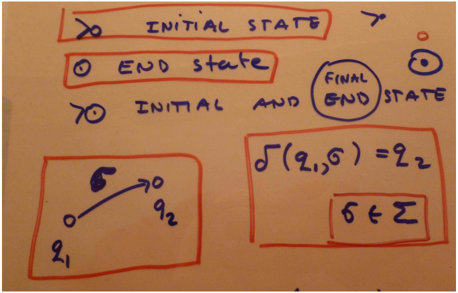
We usually use a much more clear graphical representation of the transition functions that is called a **state diagram**

Definition

The **state diagram** is a directed graph, with certain additional information as shown at the picture on next slide

DFA State Diagram

PICTURE 1



States are represented by the nodes

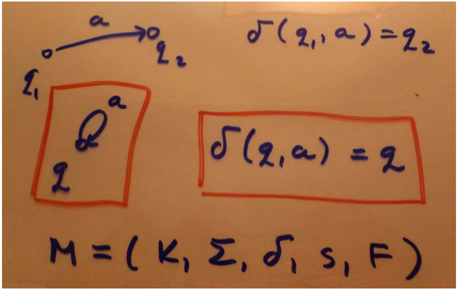
Initial state is shown by a $>\bigcirc$

Final states are indicated by a dot in a circle $\odot$

Initial state that is also a **final state** is pictured as $>\odot$

DFA State Diagram

PICTURE 2



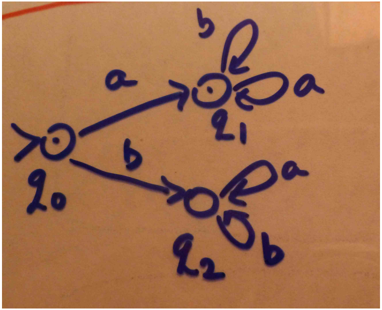
States are represented by the nodes

There is an **arrow labelled** a from node q_1 to q_2 whenever $\delta(q_1, a) = q_2$

A Simple Problem

Problem

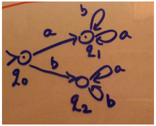
Given $M = (K, \Sigma, \delta, s, F)$ described by the following **diagram**



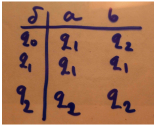
1. List all components of M
2. Describe $L(M)$ as a **regular expression**

A Simple Problem

Given the **diagram**



Components are: $M = (K, \Sigma, \delta, s, F)$ for
$\Sigma = \{a, b\}$, $K = \{q_0, q_1, q_2\}$,
$s = q_0$, $F = \{q_0, q_1\}$ and the **transition function** is given by following table



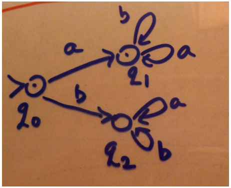
δ	a	b
q_0	q_1	q_2
q_1	q_1	q_1
q_2	q_2	q_2

A Simple Problem

2. Describe $L(M)$ as a **regular expression**, where

$$L(M) = \{w \in \Sigma^* : (s, w) \vdash_M{}^* (q, e) \quad \text{for} \quad q \in F\}$$

Let's look again at the **diagram** of M



Observe that the state q_2 **does not influence** the language L(M). We call such state a **trap state** and say:

The state q_2 is a **trap state**

We read from the **diagram** that

$$L(M) = a(a \cup b)^* \cup e \quad \text{as a regular expression}$$

$$L(M) = \{a\} \circ \{a, b\}^* \cup \{e\} \quad \text{as a set}$$

DFA Theorem

DFA Theorem

For any DFA $M = (K, \Sigma, \delta, s, F)$,

$$e \in L(M) \quad \text{if and only if} \quad s \in F$$

where we **defined** $L(M)$ as follows

$L(M) = \{w \in \Sigma^* : (s, w) \vdash_M{}^* (q, e)$ for some $q \in F\}$

Proof

Let $e \in L(M)$, then by definition $(s, e) \vdash_M{}^* (q, e)$ and $q \in F$

This is possible only when the computation is of the length one (case $n = 1$), i.e when it is (s, e) and $s = q$, hence $s \in F$

Suppose now that $s \in F$

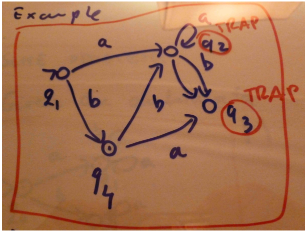
We know that $\vdash_M{}^*$ is reflexive, so $(s, e) \vdash_M{}^* (s, e)$ and as $s \in F$, we get $e \in L(M)$

Definition of TRAP States of M

Definition

A **trap state** of a DFA automaton M is any of its states that **does not influence** the language L(M) of M

Example

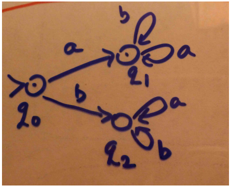


$L(M) = b$ written in shorthand notation, $L(M) = \{b\}$, or $L(M) = \mathcal{L}(b) = \{b\}$

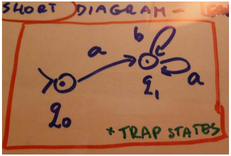
States q_2, q_3 are **trap states**

TRAP States of M

Given a **diagram** of M

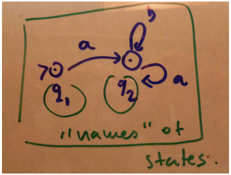


The state q_2 is the **trap state** and we can write a **short diagram** of M as follows

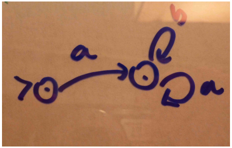


Remember that if you use the **short diagram** you must add statement: " plus **trap states**"

Definition
A diagram of M with some or all of its **trap states** removed is called a **short diagram**
"Our" M becomes



We can "shorten" the diagram even more by removing the **names** of the states

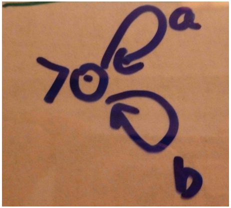


Such diagram, with names of the states removed is called a **pattern diagram**

Pattern Diagrams

Pattern Diagrams are very useful when we want to "read" the language M directly out of the diagram

Lets look at M_1 given by a diagram



It is obvious that (we write a shorthand notion!)

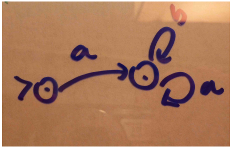
$$L(M_1) = (a \cup b)^* = \Sigma^*$$

Remark that the **regular expression** that defines the language $L(M_1)$ is $\alpha = (a \cup b)^*$

We add the description $L(M_1) = \Sigma^*$ as yet another useful informal **shorthand notation** notation

Pattern Diagrams

The **pattern diagram** for "our" M is



It is obvious that (we write a shorthand notion!) - must add:
plus **trap states**

$$L(M) = aL(M_1) \cup e$$

We must add e to the language by **DFA Theorem**, as we have that $s \in F$
Finally we obtain the following regular expression that defines the language and write it as

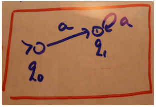
$$L(M) = a(a \cup b)^* \cup e$$

We can also write L(M) in an **informal way** (Σ^* is not a regular expression) as

Trap States

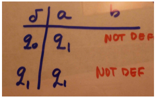
Why do we need trap states?

Let's take $\Sigma = \{a, b\}$ and let M be defined by a diagram



Obviously, the diagram means that M is such that its language is $L(M) = aa^*$

But by definition, $\delta : K \times \Sigma \longrightarrow K$ and we get from the diagram



δ	a	b
q_0	q_1	NOT DEF
q_1	q_1	NOT DEF

We must "complete" definition of δ by making it a function (still preserving the language)

To do so introduce a new state q_2 and make it a **trap state** by defining $\delta(q_0, b) = q_2$, $\delta(q_1, b) = q_2$

Short Problems

For all **short problems** presented here and given on Quizzes and Tests, you have to do the following

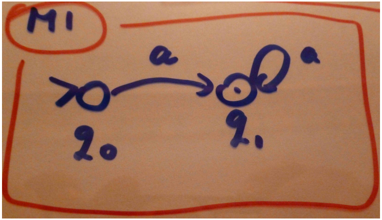
1. Decide and explain whether the given **diagram** represents a DFA or does not, i.e. is not an automatan

2. List all components of M when it represents a DFA

3. Describe $L(M)$ as a **regular expression** when it does represent a DFA

Short Problems

Consider a **diagram** M1



1. Yes, it represents a DFA; δ is a function on $\{q_0, q_1\} \times \{a\}$ and initial state $s = q_0$ exists

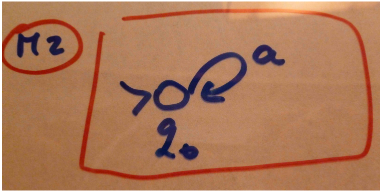
2. $K = \{q_0, q_1\}, \ \Sigma = \{a\}, \ s = q_0, \ F = \{q_1\},$

$\delta(q_0, a) = q_1, \ \ \delta(q_1, a) = q_1$

3. $L(M1) = aa^*$

Short Problems

Consider a **diagram** M2



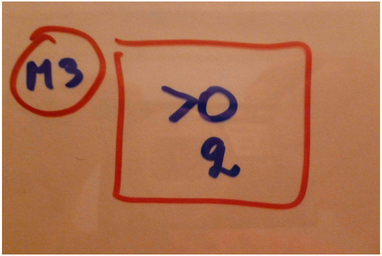
1. Yes, it represents a DFA; δ is a function on $\{q_0\} \times \{a\}$ and initial state $s = q_0$ exists

2. $K = \{q_0\},\ \Sigma = \{a\},\ s = q_0,\ F = \emptyset,\ \delta(q_0, a) = q_0$

3. $L(M2) = \emptyset$

Short Problems

Consider a **diagram** M3



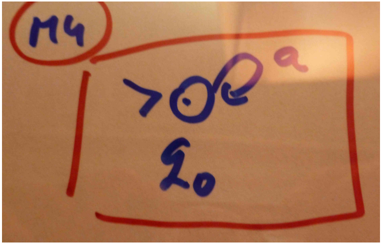
1. Yes, it represents a DFA; initial state $s = q_0$ exists

2. $K = \{q_0\}, \ \Sigma = \emptyset, \ s = q_0, \ F = \emptyset, \ \delta = \emptyset$

3. $L(M3) = \emptyset$

Short Problems

Consider a **diagram** M4



1. Yes, it represents a DFA; initial state $s = q_0$ exists

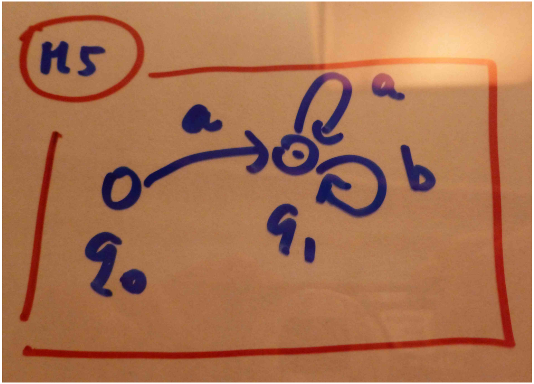
2. $K = \{q_0\},\ \Sigma = \{a\},\ s = q_0,\ F = \{q_0\},\ \delta(q_0, a) = q_0$

3. $L(M4) = a^*$

Remark $e \in L(M4)$ by **DFA Theorem**, as $s = q_0 \in F = \{q_0\}$

Short Problems

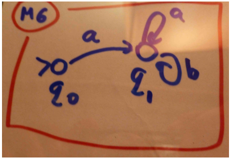
Consider a **diagram** M5



1. NO! it is NOT DFA - **initial state** does not exist

Short Problems

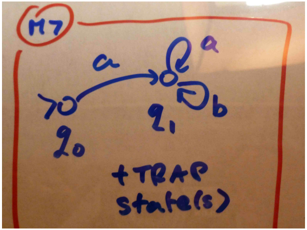
Consider a **diagram** M6



1. NO! Initial state does exist, but δ is not a function; $\delta(q_0, b)$ is **not defined** and we didn't say "plus **trap states**"

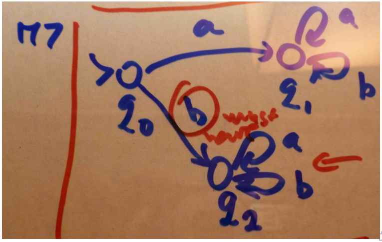
Short Problems

Consider a **diagram** M7

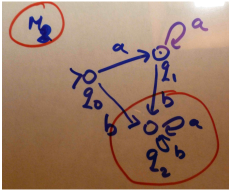


1. Yes! it is DFA

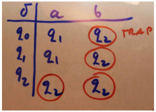
Initial state exists and we can complete definition of δ by adding a **trap state** as pictured below



Consider a **diagram** M8



1. Yes! Initial state exists and it is a **short diagram** of a DFA
We make δ a function by adding a **trap state** q_2



δ	a	b	
q_0	q_1	q_2	TRAP
q_1	q_1	q_2	
q_2	q_2	q_2	

3. $L(M8) = aa^*$
We chose to add one **trap state** but it is possible to add as many as one wishes
Observe that $L(M8) = L(M1)$ and M1, M8 are defined for different alphabets

Two Problems

P1 Let $\Sigma = \{a_1, a_2, \ldots, a_{1025}, \ \ldots, a_{2^{105}} \}$

Draw a **state diagram** of M such that $L(M) = a_{1025(}a_{1025})^*$

P2

1. Draw a **state diagram** of **transition function** δ given by the table below

2. Give an **example** and automaton M with with this δ

q	σ	$\delta(q,\sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_0
q_1	b	q_2
q_2	a	q_0
q_2	b	q_3
q_3	a	q_3
q_3	b	q_3

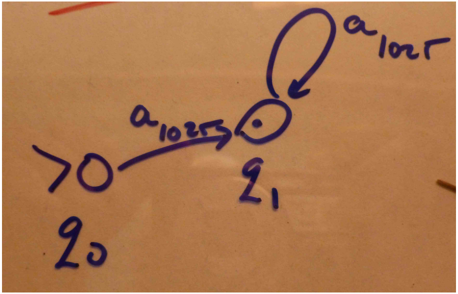
3. Describe the language of M

P1 Solution

P1 Let $\Sigma = \{a_1, a_2, \ldots, a_{1025}, \ldots, a_{2^{105}}\}$

Draw a **state diagram** of M such that $L(M) = a_{1025}(a_{1025})^*$

Solution

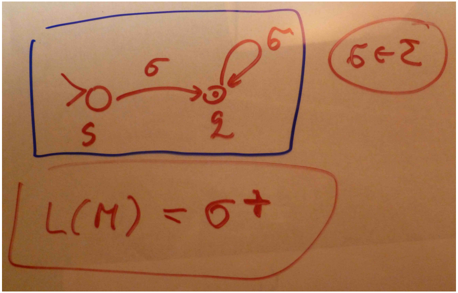


PLUS a LOT of **trap states**!

Σ has 2^{105} elements; we need a **trap state** for each of them except a_{1025}

P1 Solution

Observe that we have a following **pattern** for any $\sigma \in \Sigma$



$$L(M) = \sigma^{+} \quad \text{for any} \quad \sigma \in \Sigma$$

PLUS a LOT of **trap states**! except for the case when $\Sigma = \{\sigma\}$

P2 Solutions

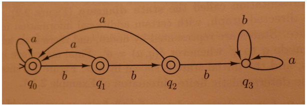
P2

1. Draw a **state diagram** of **transition function** δ given by the table below

2. Give an **example** and automaton M with with this δ

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_0
q_1	b	q_2
q_2	a	q_0
q_2	b	q_3
q_3	a	q_3
q_3	b	q_3

Here is the **example** of M from our book, page 59

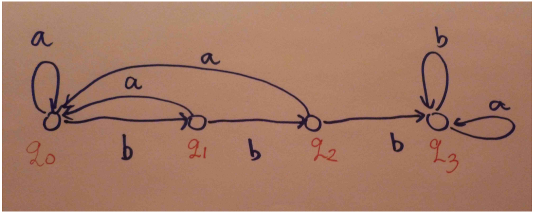


$L(M) = \{w \in \{a, b\}^* :\ w$ does not contain three consecutive $b's\}$

P2 Solution

Observe that the book example is only one of many possible examples of automata **we can define** based on δ with the following

State diagram:



Two more examples follow

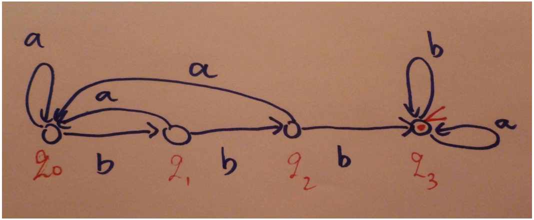
Please invent some more of your own!

Be careful! This diagram is NOT an automaton!!

P2 Examples

Example 1

Here is a full **diagram** of M1

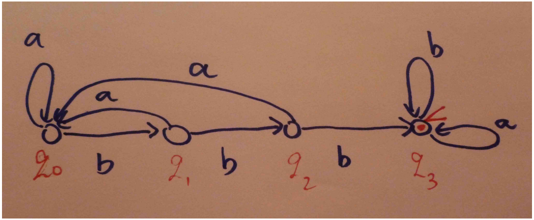


$$L(M) = (a \cup b)^* = \Sigma^*$$

Observe that $e \in L(M1)$ by the DFA **Theorem** and the states q_0, q_1, q_2 are **trap states**

Example 2

Here is a full **diagram** of M1 from **Example 1**



$$L(M) = (a \cup b)^* = \Sigma^*$$

Observe that we can make **all, or any** of the states q_0, q_1, q_2 as **final states** and they will still will remain the **trap states**

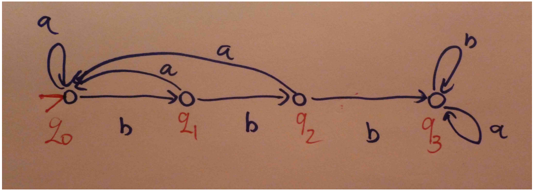
Definition

A **trap state** of a DFA automaton M is any of its states that **does not influence** the language L(M) of M

P2 Examples

Example 3

Here is a full **diagram** of M2 with the same transition function as M1



$$L(M) = \emptyset$$

Observe that $F = \emptyset$ and hence here is no computation that would finish in a **final state**

More Problems

P3 Construct a DFA M such that

$L(M) = \{w \in \{a, b\}^* :\ w \text{ has } abab \text{ as a substring } \}$

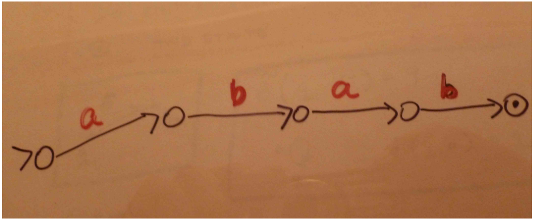
Problems Solutions

P3 Construct a DFA M such that

$$L(M) = \{w \in \{a, b\}^* : \ w \text{ has } abab \text{ as a substring } \}$$

Solution The essential part of the **diagram** must produce abab and it can be surrounded by proper elements on both sides and can be repeated

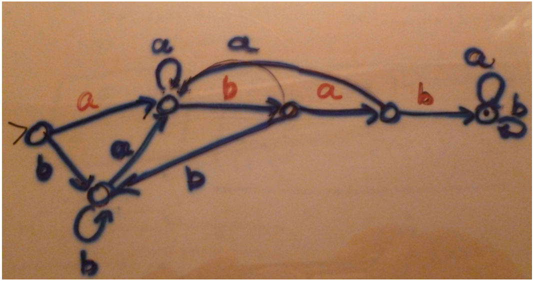
Here is the essential part of the **diagram**



Problems Solutions

We complete the essential part following the fact that it can be surrounded by proper elements on both sides and can be repeated

Here is the **diagram** of M



Observe that this is a **pattern diagram**; you need to add names of states only if you want to list all components

M does not have trap states

More Problems

P4 Construct a DFA M such that

$L(M) = \{w \in \{a, b\}^* :$ every substring of length 4 in word w

contains at least one b $\}$

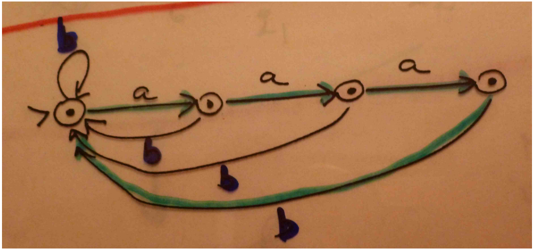
More Problems

P4 Construct a DFA M such that

$L(M) = \{w \in \{a,b\}^* :$ every substring of length 4 in word w

contains at least one b }

Solution Here is a **short pattern diagram** (the trap states are not included)



More Problems

P5 Construct a DFA M such that

$L(M) = \{w \in \{a, b\}^* :$ every word w contains

an even number of sub-strings ba $\}$

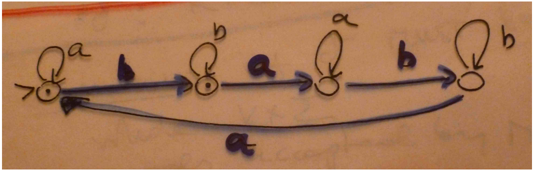
More Problems

P5 Construct a DFA M such that

$L(M) = \{w \in \{a, b\}^* :$ every word w contains

an even number of sub-strings ba $\}$

Solution Here is a **pattern diagram**



Zero is an even number so we must have that $e \in L(M)$, i.e. we have to make the initial state also a final state

More Problems

P6 Construct a DFA M such that

$L(M) = \{w \in \{a,b\}^* :$ each a in w is immediately preceded and immediately followed by b $\}$

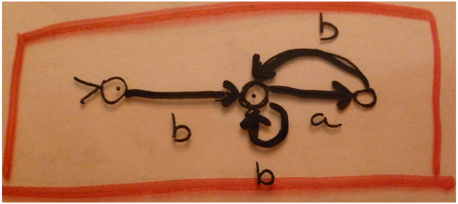
More Problems

P6 Construct a DFA M such that

$$L(M) = \{w \in \{a, b\}^* : \text{ each } a \text{ in } w \text{ is}$$

immediately preceded and immediately followed by b $\}$

Solution: Here is a **short pattern diagram** - and we need to say: plus trap states)

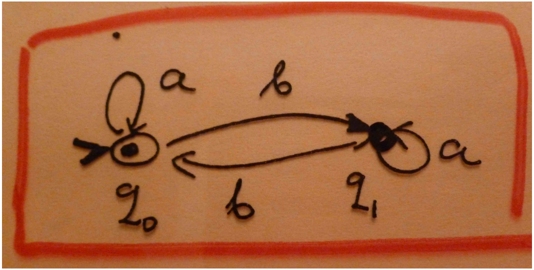


It is a **short diagram** because we omitted needed **trap states** (can be more then one, but one is sufficient)

Complete the diagram as an exercise

More Problems

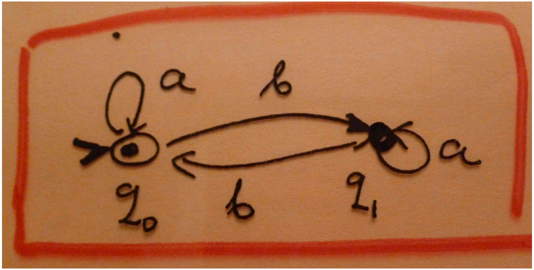
P7 Here is a DFA M defined by the following diagram



Describe $L(M)$ as a **regular expression**

More Problems

P7 Here is a DFA M defined by the following diagram



Describe $L(M)$ as a **regular expression**

Solution

$$L(M) = a^* \cup (a^*ba^*ba^*)^*$$

Observe that $e \in L(M)$ by the DFA **Theorem**

Short Problems

SP1 Given an automaton M1

$$M1 = (\ K = \{q_0,\ q_1\},\ \Sigma = \{a,\ b\},\ \ \delta,\ s = q_0,\ F = \emptyset)$$

$$\delta(q_0, a) = q_0,\ \ \delta(q_0, b) = q_0,\ \ \delta(q_1, a) = q_0,\ \ \delta(q_1, b) = q_0$$

1. Draw its **state diagram**

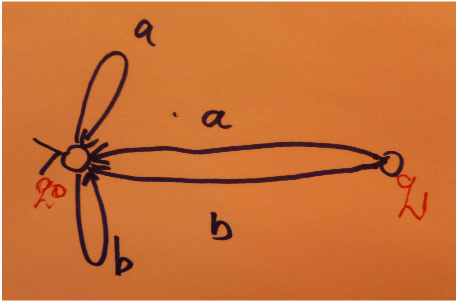
2. List **trap states**, if any

3. Describe L(M1)

SP1 Solution

SP1

1. Here is the **state diagram**



2. q_1 is a **trap state** - M1 never gets there

3. $L(M1) = \emptyset$

Short Problems

SP2 Given an automaton M2

$$M2 = (\ K = \{q_0,\ q_1\},\ \Sigma = \{a,\ b\},\ \ \delta,\ s = q_0,\ F = \{q_1\})$$

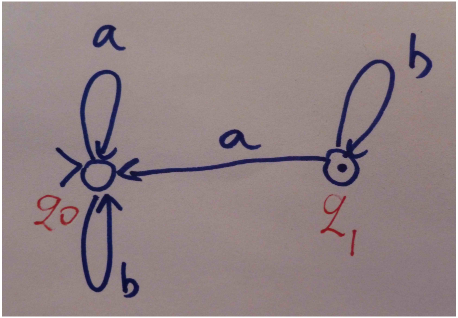
$$\delta(q_0, a) = q_0,\ \ \delta(q_0, b) = q_0,\ \ \delta(q_1, a) = q_0,\ \ \delta(q_1, b) = q_1$$

1. Draw its **state diagram**
2. List **trap states**, if any
3. Describe L(M2)

SP2 Solution

SP2

1. Here is the **state diagram**



2. q_1 is a **trap state** - it does not influence the language of M1

3. $L(M2) = \emptyset$

Short Problems

SP3 Given an automaton M3

$M3 = (K = \{q_0,\ q_1\},\ \Sigma = \{a,\ b\},\ \delta,\ s = q_0,\ F = \{q_1\})$

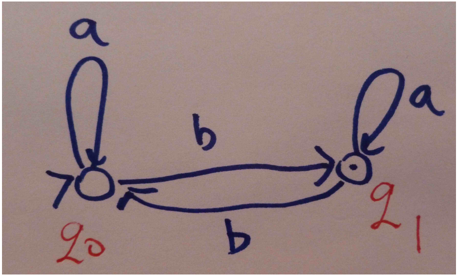
$\delta(q_0, a) = q_0,\ \delta(q_0, b) = q_1,\ \delta(q_1, a) = q_1,\ \delta(q_1, b) = q_0$

1. Draw its **state diagram**
2. List **trap states**, if any
3. Describe L(M3)

SP3 Solution

SP3

1. Here is the **state diagram**



2. There are no **trap states**

3. $L(M3) = a^*b \cup a^*ba^* \cup (a^*ba^*ba^*b)^*$

$L(M3) = a^*ba^* \cup (a^*ba^*ba^*b)^*$

Short Problems

SP4 Given an automaton $M4 = (K, \Sigma, \delta, s, F)$ for $K = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_0, q_1, q_2\}$ and δ defined by the table below

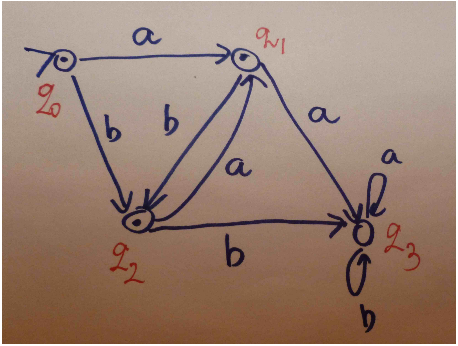
q	σ	$\delta(q, \sigma)$
q_0	a	q_1
q_0	b	q_2
q_1	a	q_3
q_1	b	q_2
q_2	a	q_1
q_2	b	q_3
q_3	a	q_3
q_3	b	q_3

1. Draw its **state diagram**
2. Give a **property** describing L(M4)

SP4 Solution

SP4

1. Here is the **state diagram**

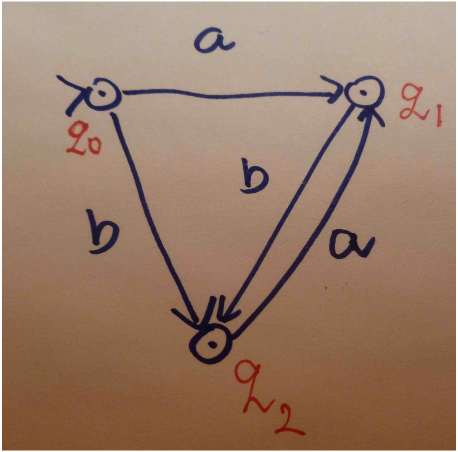


Observe that state q_3 is a **trap state** and the **short diagram** is as follows

SP4 Solution

SP4

1. Here is the **short diagram**



2. The language of M4 is

$L(M4) = \{w \in \Sigma^* :$ neither aa nor bb is a substring of w $\}$

Short Problems

SP5 Given an automaton $M5 = (K, \Sigma, \delta, s, F)$ for $K = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_1\}$ and δ defined by the table below

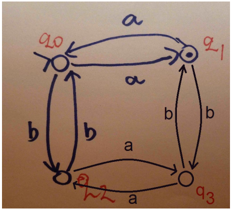
q	σ	$\delta(q, \sigma)$
q_0	a	q_1
q_0	b	q_2
q_1	a	q_0
q_1	b	q_3
q_2	a	q_3
q_2	b	q_0
q_3	a	q_2
q_3	b	q_1

1. Draw its **state diagram**
2. Give a **property** describing L(M5)

SP5 Solution

SP5

1. Here is the **state diagram**



2. $L(M5) = \{w \in \Sigma^* :$ w has an odd number of a 's

and an even number of of b 's $\}$