CS310M: Automata Theory (Minor)

Topic 2: Deterministic Finite State Automata

Paritosh Pandya

Indian Institute of Technology, Bombay
Course URL: https://cse.iitb.ac.in/∼pandya58/CS310M/automata.html

Autumn, 2021

Today's Topics:

- Decision Problems
- Finite word langauges
- Deterministic Finite State Automata (DFA)
- Big Step Semantics
- Examples of DFA

Source: Kozen, Lectures 2 and 3.
Hopcropf, Motwani, Ullman, Chapters 1 and 2

Decision Problems

Decision problem is a problem which has yes or no as answer.

Problem Formulation: pair of sets (A, B)

- inputs A : The set of inputs, candidates or instances.
- yes instances B: Subset of A for which the answer is "yes".
- Set B is usually given implicitely using some properties.

Decision Problem Given $x \in A$ to determine whether $x \in B$.

Example

- Prime: Is given natural number n a prime number?
  $A = \aleph$, the set of natural numbers. $B = \{n \in A \mid n \text{ is prime}\}$
- Power of 2; Is given natural number a power of 2?
  $A = \aleph$ and $B = \{n \mid \exists i \in \aleph.\ n = 2^i\}$.

Automaton for (A, B)

A machine which can solve any given instance of the decision problem

Euler Graphs

Euler Graphs: Given a graph $G = (V, E)$ is it Eulerian, i.e. does it have an Euler circuit.

Euler circuit starts at a node and ends at the same node visitineg each edge exactly once. (Nodes can be revisited).

A set of finite graphs. B, set of Eulerian graphs.

Problem Instances as Strings

- Question: How to give input to the automaton?
  Consider the case when $A = \aleph$ or A is the set of graphs.
- Answer: Encode each input as a string of symbols.
- Alphabet Σ: finite set of allowed symbols.

Example

- $\Sigma = \{0, 1, 2, \ldots, 9\}$, the set of decimal digits.
  A natural number n is given as a sequence of such digits.
- Unary encoding of numbers: $\Sigma = \{a\}$, single letter alphabet.
  Encode number n as string $aaa \ldots a$ where a occurs n times.
  This is denoted as a^n.
- What about number 0 in unary encoding?
  Empty string, i.e. sequence with zero a. This is denoted as ϵ.
- Binary encoding of numbers: $\Sigma = \{0, 1\}$. Encode number n in binary. E.g. 9 can be encoded as string 1001.

Finite Words

- Alphabet Σ: a finite set of symbols (letters). E.g. $\Sigma = \{0, 1\}$. We use $a, b, c \ldots$ to denote letters.
- Word or String over Σ is a finite sequence of letters from Σ. E.g. 10010. We use x, y, z, u, v, w for words.
- Σ^* denotes the set of all words over Σ.
- $|x|$ denotes the length of word x. E.g. $|abaa| = 4$.
- ϵ denotes the empty word. $|\epsilon| = 0$.
- catenation of x and y is denoted $x \cdot y$. E.g. if $x = ab$ and $y = bba$, then $xy = abbba$.
  This gives $|xy| = |x| + |y|$.
- Properties of catenation:
  $x(yz) = (xy)z$ (associativity)
  $\epsilon \cdot x = x = x \cdot \epsilon$ (identity)
  Note that xy is not same as yx in general.
- Define $x^0 = \epsilon$ and $x^{n+1} = (x^n) \cdot x$.
  E.g. $(aab)^3 = aabaabaab$.

Inductive Definition

Define $x^0 = \epsilon$ and $x^{n+1} = (x^n) \cdot x$.

E.g. $(aab)^3 = aabaabaab$.

$= (aab)^2 . aab$

$= ((aab)^1 . aab) . aab$

$= (((aab)^0 . aab) . aab) . aab$

$= (((\epsilon . aab) . aab) . aab$

$|\alpha| : \Sigma^* \to \mathbb{N}$

$|\varepsilon| = 0$

$|u a| = |u| + 1$

Language

A language over alphabet Σ is any subset of Σ^*. We use A, B, C to denote languages. We can define langauges using the set builder notation.

Example

- Let $\Sigma = \{a\}$. Define

$$A = \{a^p \mid p \text{ is a prime}\}$$

$\{aa, aaa, aaaaa, \ldots\}$

- Let $\Sigma = \{0, 1\}$. For a word $x \in \Sigma^*$ let $\#0(x)$ denote count of 0 in x. Similarly, $\#1(x)$ is count of 1. E.g. $\#0(10011) = 2$ and $\#1(10011) = 3$. Define

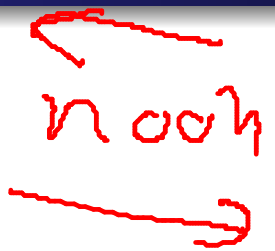
$$A = \{x \in \Sigma^* \mid \#0(x) = \#1(x)\}$$

- Let $\Sigma = \{0, 1\}$. Define

$\{1, 10, 100, 1000, \ldots\}$

$$A = \{x \in \Sigma^* \mid \exists i \in \aleph . . x \text{ is binary representation of } 2^i\}$$

Languages : Examples



- Words over $\Sigma = \{a, b, c, \ldots, z\}$ which are palindromes. E.g. noon, madam.
- Words over ASCII alphabet which are syntactically correct C programs.
- Let $\Sigma = \{0, 1\}$.

$$A = \{0^n 1^n \mid n \geq 0\}$$

$\{\epsilon, 01, 0011, \ldots, 0^n 1^n, \ldots, 0^2 1^2, \ldots\}$

Examples of Languages

Example

Let alphabet $\Sigma = \{a, b\}$. Recall that for $x \in \Sigma^*$ the term $\#a(x)$ gives the count of letter a in x.

- $A_1 = \{x \in \{a, b\}^* \mid \#a(x) = \#b(x)\}$.
- $A_2 = \{x \mid \#a(x) \ \textit{isevenand} \ \#b(x)$ is also even$\}$.
- $A_3 = \{a^i b^j \mid i, j \in \aleph, i > j\}$.

Operations on Languages

Let A, B be languages ove Σ.

- $A \cup B = \{x \in \Sigma^* \mid x \in A \text{ or } x \in A\}$
  $A \cap B = \{x \in \Sigma^* \mid x \in A \text{ and } x \in A\}$.
- Complement $\sim A = \{x \in \Sigma^* \mid x \notin A\}$.

Operations on Languages

$ab.bb = abbb$

Let A, B be languages ove Σ.

- $A \cup B = \{x \in \Sigma^* \mid x \in A \text{ or } x \in A\}$
  $A \cap B = \{x \in \Sigma^* \mid x \in A \text{ and } x \in A\}$.
- Complement $\sim A = \{x \in \Sigma^* \mid x \notin A\}$.
- Catenation of Languages: Given languages A, B over Σ,

$$A \cdot B = \{x \cdot y \mid x \in A \text{ and } y \in B\}$$

Example Let $A = \{a, ab\}$ and $B = \{b, ba\}$. Then,
$A \cdot B = \{ab, aba, abb, abba\}$
$B \cdot A = \{ba, bab, baa, baab\}$

Operations on Languages

Let A, B be languages ove Σ.

- $A \cup B = \{x \in \Sigma^* \mid x \in A \text{ or } x \in A\}$
  $A \cap B = \{x \in \Sigma^* \mid x \in A \text{ and } x \in A\}$.
- Complement $\sim A = \{x \in \Sigma^* \mid x \notin A\}$.
- Catenation of Languages: Given languages A, B over Σ,

$$A \cdot B = \{x \cdot y \mid x \in A \text{ and } y \in B\}$$

  Example Let $A = \{a, ab\}$ and $B = b, ba$. Then,
  $A \cdot B = \{ab,\ aba,\ abb,\ abba\}$
  $B \cdot A = \{ba,\ bab,\ baa,\ baab\}$
- Powers, Kleen Closure and Laws for reasoning about Languages (later lecture).

$A \cdot (B \cup C) = A \cdot B \cup A C$

Language recognition

Language recognition problem for language A Given input word x to determine whether $x \in A$?

All computational decision problems can be encoded as language recognition problems.

Language recognition

Language recognition problem for language A Given input word x to determine whether $x \in A$?

All computational decision problems can be encoded as language recognition problems.

Automaton

Language recognition

Language recognition problem for language A Given input word x to determine whether $x \in A$?

All computational decision problems can be encoded as language recognition problems.

Automaton

- Automaton is a computational device to solve a langauge recognition problems.
  It works by making a sequence of simple moves till answer YES or NO is produced.

Language recognition

Language recognition problem for language A Given input word x to determine whether $x \in A$?

All computational decision problems can be encoded as language recognition problems.

Automaton

- Automaton is a computational device to solve a langauge recognition problems.
  It works by making a sequence of simple moves till answer YES or NO is produced.
- Automaton is a acceptor or recognizer

Language recognition

Language recognition problem for language A Given input word x to determine whether $x \in A$?

All computational decision problems can be encoded as language recognition problems.

Automaton

- Automaton is a computational device to solve a langauge recognition problems.
  It works by making a sequence of simple moves till answer YES or NO is produced.
- Automaton is a acceptor or recognizer
- Each automaton defines a language.

Automaton Example

Let $A_1 = \{x \in \{a, b\}* \mid \#a(x) = \#b(x)\}$.
Automaton for recognition of words in A_1

Three Classes of Automata

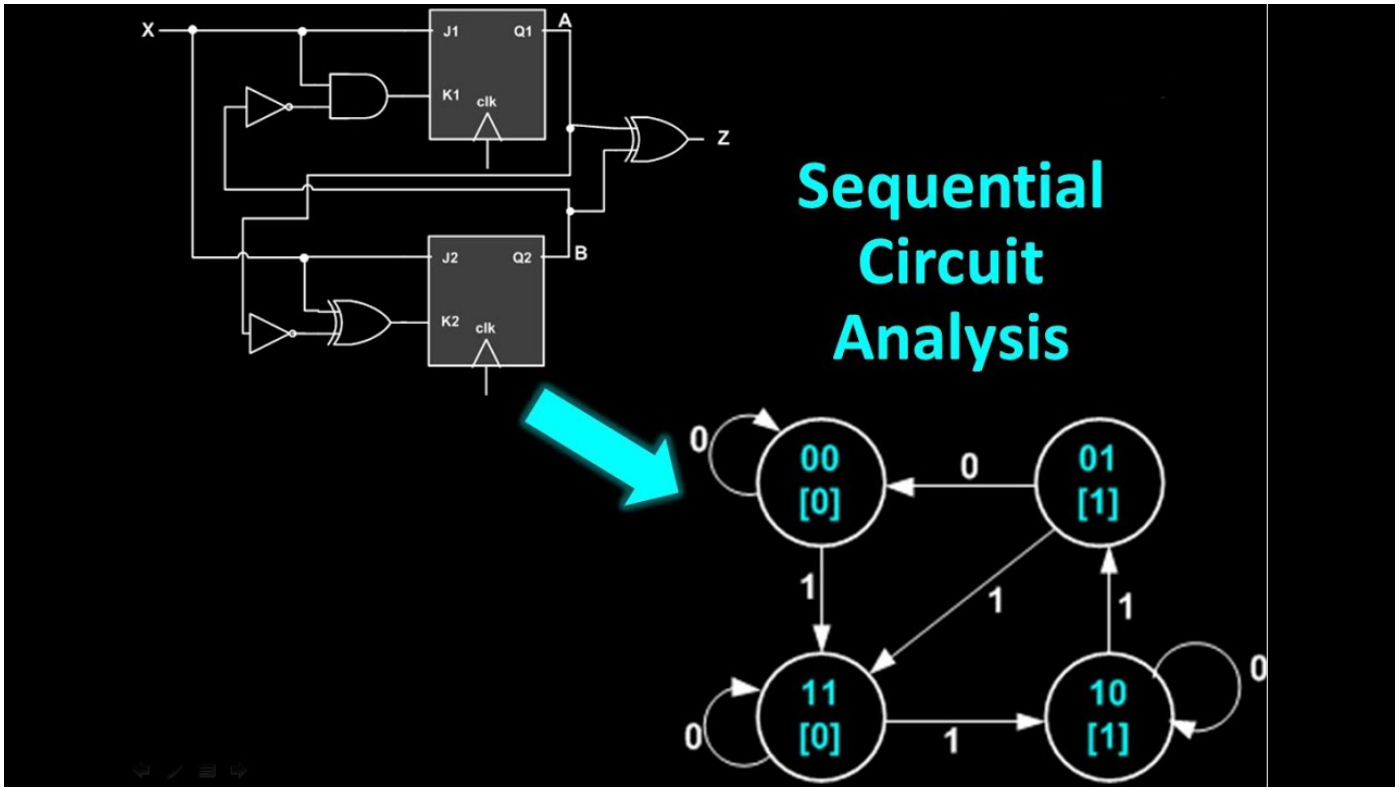
- Finite State Automata
- Push Down Automata
- Turing Machines

Finite State Automata

Computations which can be carried out with only a bounded amount of memory.

Theorem

Every Sequential Circuit can be modelled as an equivalent Finite State Automaton and vice versa.



Finite State Automata

Model of computational systems with bounded amount of memory.

Widely applicable

- Digital Sequential Circuits.
- Embedded System Controllers.
  Languages: Esterel, Lustre, Verilog, SMV.
- Regular Expression Pattern Matching
- Protocols: Network Protocols, Bus, Cache Coherence ... protocols
- Compiler Design : Lexical analysis.
- Natural Language Processing: Sequence to sequence transformers.

Good Analysis Procedures

- Automata Composition
- Decision Procedures

Topics

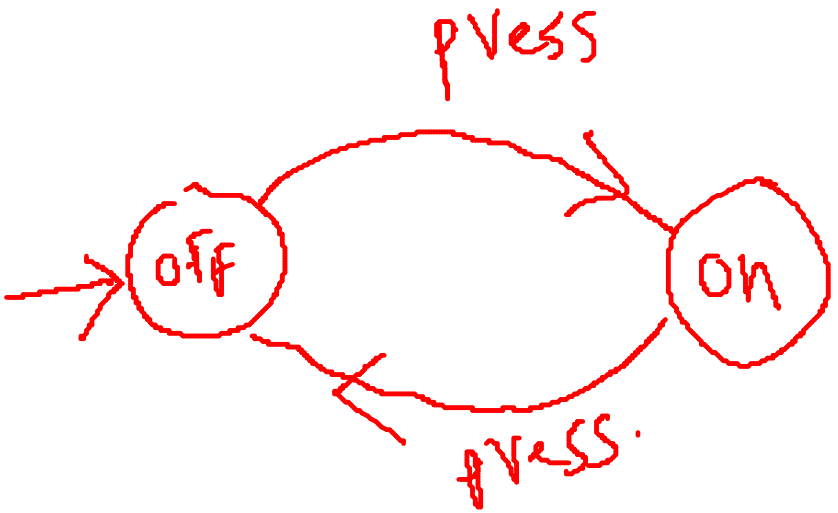
- Deterministic Finite Automata (DFA) and its uses.
- DFA, NFA and Equivalence
- Closure Properties and Decision Problems
- Regular Expressions and Equivalence to DFA
- DFA minization
- Pumping Lemma
- Myhill Nerode Theorem

Today's Lecture

- DFA its Language: Formal Definition
- Examples: Language $\rightarrow$ DFA
  DFA $\rightarrow$ ~~DFA~~ Language
- Structural Induction over words
- Big Step Semantics
- Product Construction

Deterministic Finite State Automaton (DFA)

Example: Light with toggle Switch



Set of states

$Q = \{off, on\}$

$\Sigma = \{press\}$ alphabet

Initial state.

$q_0 = off$

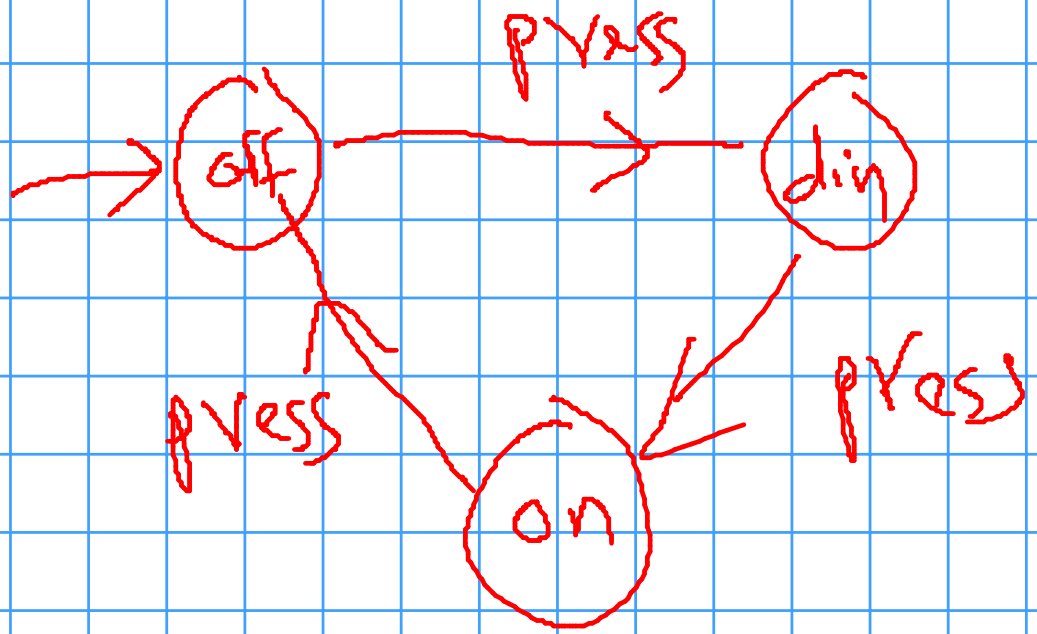
Transition Funch

$\delta : Q \times \Sigma \to Q$

$\delta(off, press) = on$

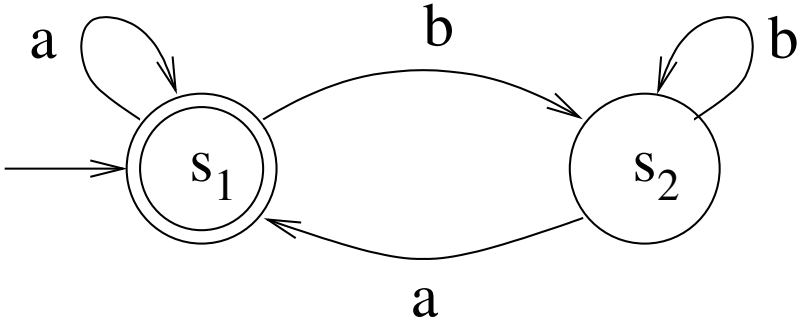
$\delta(on, press) = off$

off $\xrightarrow{press}$ on $\xrightarrow{press}$ off $\xrightarrow{press}$ on


pvess
off
din
pvess
pvesj
on

Deterministic Finite State Automaton (DFA)

Example: DFA A_1 over $\Sigma = \{a, b\}$.



$Q = \{s_1, s_2\}$

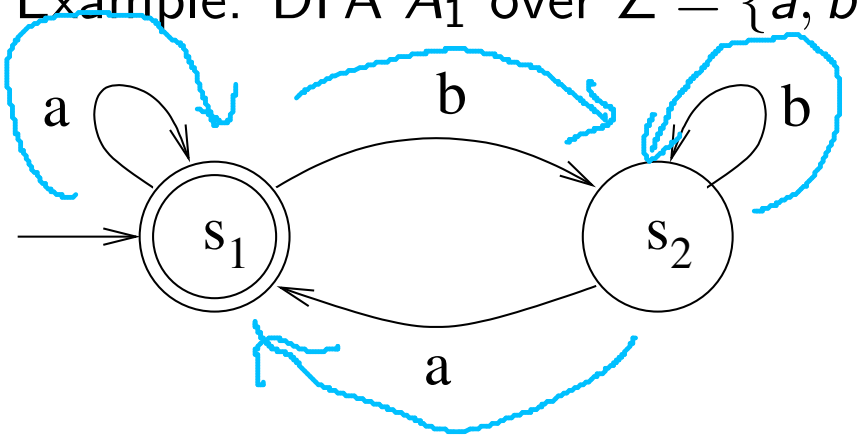
$\Sigma = \{a, b\}$

$q_0 = s_1$

$\delta(s_1, a) = s_1$

Deterministic Finite State Automaton (DFA)

Example: DFA A_1 over $\Sigma = \{a, b\}$.



Run

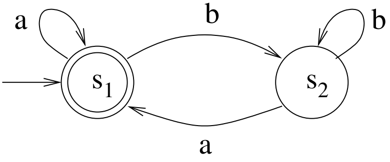
The execution of A_1 over the word $abba$ is :

$$s_1 \xrightarrow{a} s_1 \xrightarrow{b} s_2 \xrightarrow{b} s_2 \xrightarrow{a} s_1.$$

For a given word the run is unique.

$L(A) = \{\epsilon \text{ or } wa . w \in \Sigma^*\}$

Example: DFA A_1 over $\Sigma = \{a, b\}$.



Run

The execution of A_1 over the word *abba* is :

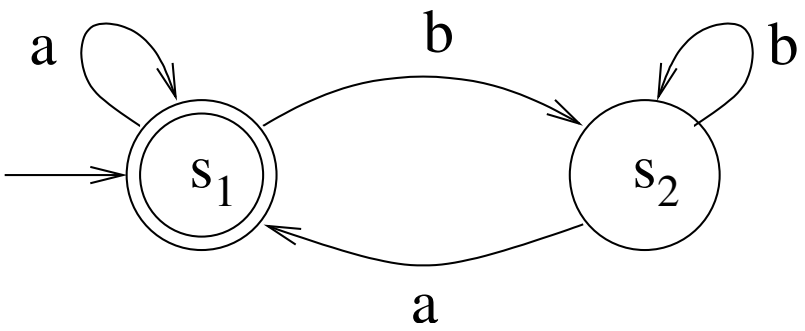
$$s_1 \xrightarrow{a} s_1 \xrightarrow{b} s_2 \xrightarrow{b} s_2 \xrightarrow{a} s_1.$$

For a given word the run is unique.

Final states $F \subseteq Q$. Run is accepting if endstate is in F.

Deterministic Finite State Automaton (DFA)

Example: DFA A_1 over $\Sigma = \{a, b\}$.



$Q = \{s_1, s_2\}$
$\Sigma = \{a, b\}$
$q_0 = s_1$
δ ....
$F = \{s_1\}$

Run

The execution of A_1 over the word *abba* is :

$$s_1 \xrightarrow{a} s_1 \xrightarrow{b} s_2 \xrightarrow{b} s_2 \xrightarrow{a} s_1.$$

For a given word the run is unique.

Recognises words ending with *a* or the empty word.

Mathematical Syntax of DFA

A DFA is given by $(Q, \Sigma, \delta, q_0, F)$ where

- Q Finite set of states.
- Σ the finite alphabet.
- $q_0 \in Q$ the initial state.
- $F \subseteq Q$ set of final states.
- $\delta : Q \times \Sigma \rightarrow Q$, the transition function.
  δ is a total function.

Three Formats

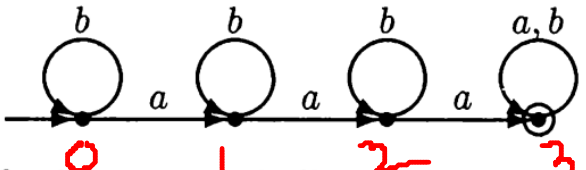
Here is an example of a simple four-state finite automaton. We'll take the set of states to be $\{0, 1, 2, 3\}$; the input alphabet to be $\{a, b\}$; the start state to be 0; the set of accept states to be $\{3\}$; and the transition function to be

$$\delta(0, a) = 1,$$
$$\delta(1, a) = 2,$$
$$\delta(2, a) = \delta(3, a) = 3,$$
$$\delta(q, b) = q, \quad q \in \{0, 1, 2, 3\}.$$

All parts of the automaton are completely specified. We can also specify the automaton by means of a table

		a	b
$\rightarrow$	0	1	0
	1	2	1
	2	3	2
	$3F$	3	3

or transition diagram



The final states are indicated by an F in the table and by a circle in the transition diagram. In both, the start state is indicated by $\rightarrow$. The states in

$$L(A) = \{ x \in \{a,b\}^* \mid \#_a(x) \geq 3 \}$$

DFA Small Step Semantics

DFA is $(Q, \Sigma, \delta, q_0, F)$

Q Finite set of states.

Σ is the finite alphabet.

$q_0 \in Q$ the initial states.

$F \subseteq Q$ set of final states.

$\delta : Q \times \Sigma \to Q$, transition function (total).

Notation: We use $q \xrightarrow{a} q'$ to denote $\delta(q, a) = q'$.

- A run of DFA A on $x = a_0, a_1, \ldots, a_{n-1}$ is a sequence of states $q_0, q_1, \ldots, q_n$ s.t. $q_i \xrightarrow{a_i} q_{i+1}$ for $0 \leq i < n$.
- For a given word x, the DFA has a unique run.
- A run is accepting if last state $q_n \in F$.

Word x is accepted by the automaton if its run is accepting.

Regular Langauges

Language accepted by A:

$$L(A) = \{u \in \Sigma^* \mid \text{The run of } A \text{ on } u \text{ is accepting } \}$$

regular language

A language is called regular if it is accepted by some DFA.

Example DFAs

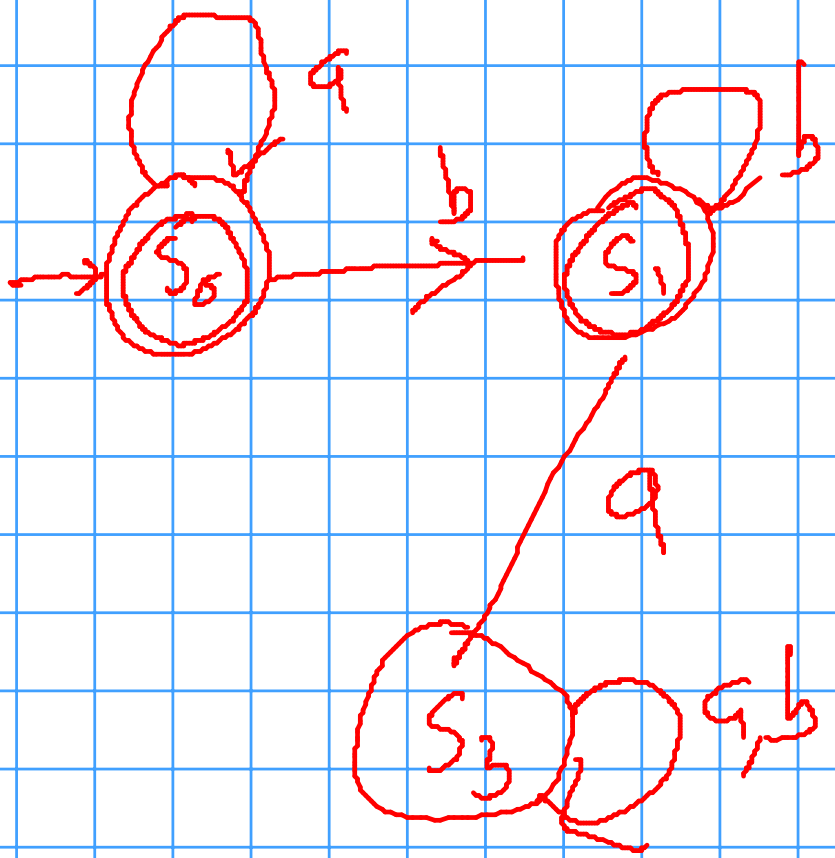
$x \leq_p y \triangleq \exists z.\ xz = y$

$x \leq_s y \triangleq \exists u, v.\ uxv = y$

Alphabet $\{a, b\}$.

- Words of the form $a^i b^j$ with $i, j \geq 0$.
- Three or more occurrences of 'a'.
- Contains subword 'aaa'
- Count of 'a' is even and count of 'b' is odd.

L $a^i b^j$, $i, j \geq 0$

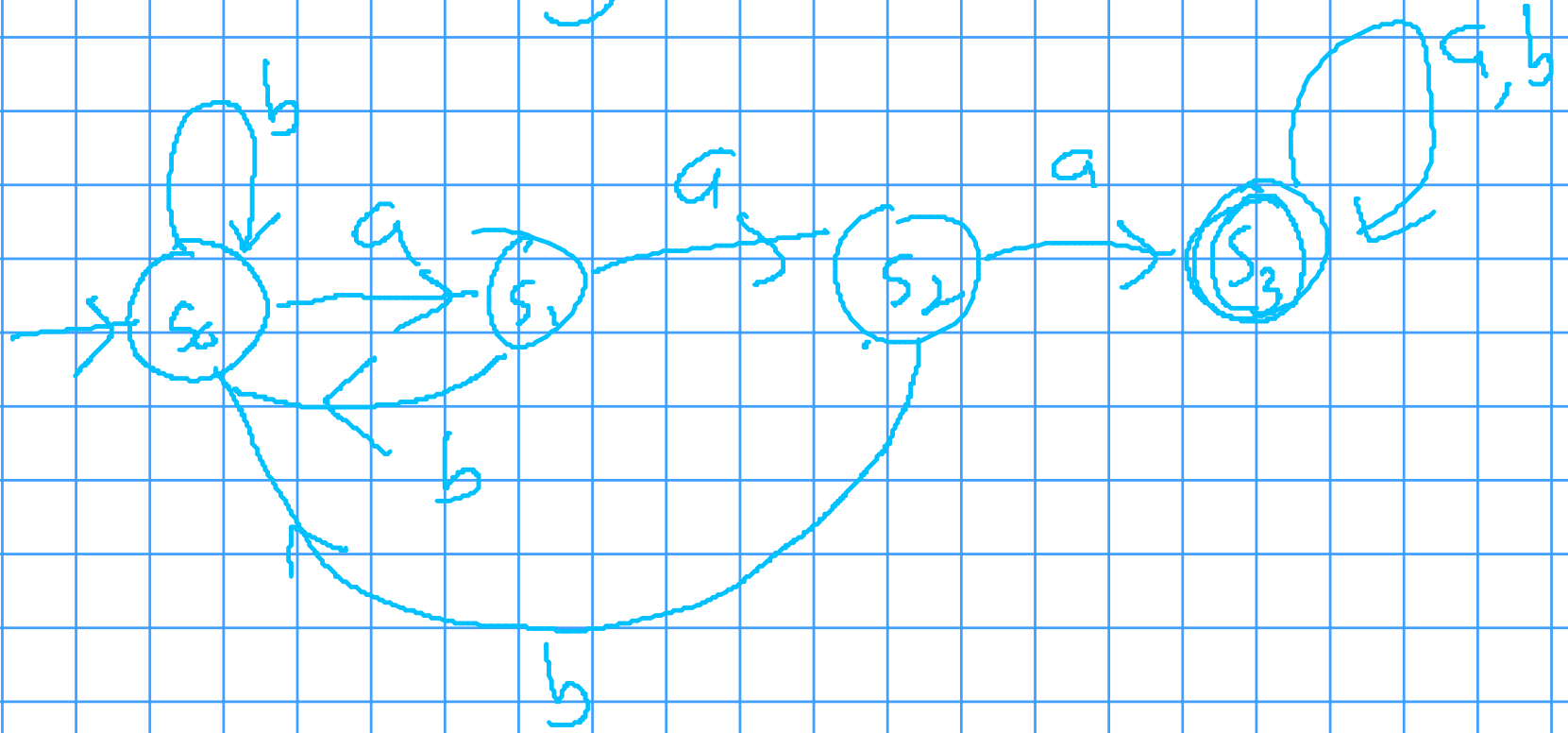


S_0 - #b = 0

S_1 #b > 0
a's before b's

S_2 an 'b' previously
an 'a'

$\{x \mid aaa \leq_s x\}$



Big Step Semantics

Given DFA with transition function $\delta : Q \times \Sigma \to Q$, define extended transition function

(unique homomorphic extension implies $\hat{\delta}$ is unique)

$$\hat{\delta} \ : \ Q \times \Sigma^* \to Q$$

Here, $\hat{\delta}(q, x)$ gives the last state of run of A on word x starting from state q. It can be defined inductively as follows:

$$\begin{aligned} \hat{\delta}(q, \epsilon) &= q, \\ \hat{\delta}(q, ua) &= \delta(\ \hat{\delta}(q, u),\ a) \end{aligned}$$

Alternate definition

Word x is accepted by the automaton iff $\hat{\delta}(q_0, x) \in F$.

$$L(A) \ = \ \{x \in \Sigma^* \ \mid \ \hat{\delta}(q_0, x) \in F\}$$

A proof using Structural Induction on words

Theorem

$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$

Proof: By structural induction on y

Basis: $y = \varepsilon$. $\hat{\delta}(q, x\varepsilon) = \hat{\delta}(q, x) = \hat{\delta}(\hat{\delta}(q, x), \varepsilon)$

Now, inducting on y, assume $\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$

Let $z = ya$

Hence,

$\hat{\delta}(q, xz) = \hat{\delta}(q, xya) = \hat{\delta}(\hat{\delta}(q, xy), a) = \hat{\delta}(\hat{\delta}(\hat{\delta}(q, x), y), a)$

$= \hat{\delta}(\hat{\delta}(q, x), ya)$

$= \hat{\delta}(\hat{\delta}(q, x), z)$ ∎

Constructions on Automata

Given component automata, we construct a new automaton from them.

Complement Automaton $\neg M$

Given $M_1 = (Q, \Sigma, \delta, q_0, F_1)$ construct $M_2 = (Q, \Sigma, \delta, q_0, F_2)$. where $F_2 = Q - F$. Note that $\hat{\delta}_{M_1} = \hat{\delta}_{M_2} = \hat{\delta}$.

Theorem

$L(M_2) = \sim L(M_1)$

Proof: $x \in L(N)$ iff $\hat{\delta}(q_0, x) \in F$ (by defn)

iff $\hat{\delta}(q_0, x) \notin \tilde{F} (= F^c)$

iff $x \notin L(\tilde{N})$

Hence, $\overline{L(N)} = L(\tilde{N})$

Product Construction

Given $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$
Define $M_3 = (Q_3, \Sigma, \delta_3, s_3, F_3)$ as follows:

$$Q_3 = Q_1 \times Q_2, \qquad s_3 = (s_1, s_2)$$
$$F_3 = F_1 \times F_2$$

$\delta_3 : Q_3 \times \Sigma \to Q_3$ given by
$\delta_3((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$

Product Construction

Given $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$
Define $M_3 = (Q_3, \Sigma, \delta_3, s_3, F_3)$ as follows:

$$Q_3 = Q_1 \times Q_2, \qquad s_3 = (s_1, s_2)$$
$$F_3 = F_1 \times F_2$$

$$\delta_3 : Q_3 \times \Sigma \to Q_3 \quad \text{given by}$$
$$\delta_3((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$$

(for union of langs,
$F_3 = F_1 \times F_2^c \cup$
$F_1^c \times F_2 \cup$
$F_1 \times F_2$

This givens us extended transition function $\hat{\delta}_3$

$$\hat{\delta}_3((p, q), \epsilon) = (p, q)$$
$$\hat{\delta}_3((p, q), xa) = \delta(\hat{\delta}_3((p, q), x), a)$$

For all $x \in \Sigma^*$,

$$\widehat{\delta}_3((p,q),x) = (\widehat{\delta}_1(p,x), \widehat{\delta}_2(q,x)).$$

Proof. By induction on $|x|$.

Basis

For $x = \epsilon$,

$$\widehat{\delta}_3((p,q),\epsilon) = (p,q) = (\widehat{\delta}_1(p,\epsilon), \widehat{\delta}_2(q,\epsilon)).$$

Induction step

Assuming the lemma holds for $x \in \Sigma^*$, we show that it holds for xa, where $a \in \Sigma$.

$$\begin{aligned}
&\widehat{\delta}_3((p,q),xa) \\
&= \delta_3(\widehat{\delta}_3((p,q),x),a) && \text{definition of } \widehat{\delta}_3 \\
&= \delta_3((\widehat{\delta}_1(p,x), \widehat{\delta}_2(q,x)),a) && \text{induction hypothesis} \\
&= (\delta_1(\widehat{\delta}_1(p,x),a), \delta_2(\widehat{\delta}_2(q,x),a)) && \text{definition of } \delta_3 \\
&= (\widehat{\delta}_1(p,xa), \widehat{\delta}_2(q,xa)) && \text{definition of } \widehat{\delta}_1 \text{ and } \widehat{\delta}_2.
\end{aligned}$$

□