Discrete Structures

Modeling Computation

Languages

- Σ: The set of alphabets
- Σ^*: The set of all words on Σ
- λ; The word with length zero.
- Language: Any $L \subset \Sigma^*$

Example: English Language (any sentence can be seen as a word), C++ (any code can be seen as a word, although it can be very large), $L = \{a^n b^n : n \geq 0\}$ over $\Sigma = \{a, b\}$, and $L = \{0,00,00000\}$ over $\Sigma = \{0\}$

The set of all languages is uncountable as it is the power set of Σ^*.

Languages

The main questions:

- How to represent a language L? For instance, C++ is a complicated language. It is impossible to show the set C++ by its elements (codes).
- How to detect a give word $w \in L$? Assume you are give a code in C++ and you are asked to check whether the code satisfies all syntaxes of C++ or not?

Problems as Languages

Problem: Given a graph G. Is it possible to color the vertices with 3 colors s.t. every two adjacent vertices have different colors.

Language: Let $L = \{$ all 3-colorable graphs $\}$. Now for the given graph G, we have to detect whether $G \in L$.

Problem: Given a positive integer number p. detect whether p is a prime number or not.

Language: Let $L = \{2,3,5,7,11,\ldots\}$. For a given p, detect whether $p \in L$.

Representing Languages

Several tools (machines) have been defined to help us to represent our desired languages like

- Automata
- Regular expressions
- Grammars
- Turning machines

The power of these machines are different

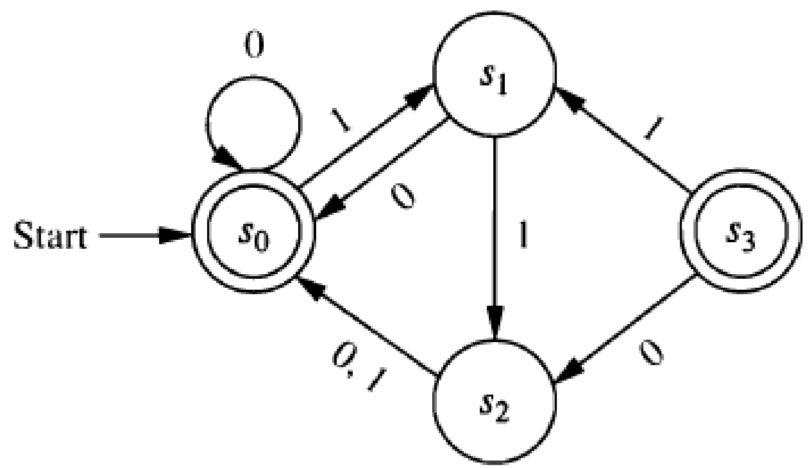
- A language may be possible to be represented by one machine while not possible to be defined with another machine.
- Detecting $w \in L$ is easier in one machine than another machine.

Automata

A finite-state automaton is a 5-tuple $M = (S\ \Sigma, f, s_0, F)$

- S: states
- Σ: alphabets
- s_0: the initial or start state
- F: final states or accepting states
- $f: S \times \Sigma \rightarrow S$: a transition function

State	f Input 0	f Input 1
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_0	s_0
s_3	s_2	s_1



Language of an Automaton

Definition:

- $f(s, \lambda) = s$
- $f(s, xa) = f(f(s, x), a), a \in \Sigma, x \in \Sigma^*$

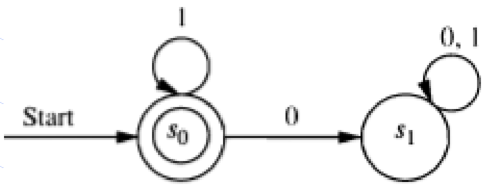
Definition:

- A string x is said to be recognized or accepted by M iff $f(s_0, x) \in F$.
- $L(M)$ is the set of strings recognized by M.
- Two finite-state automata are equivalent if they recognize the same language.

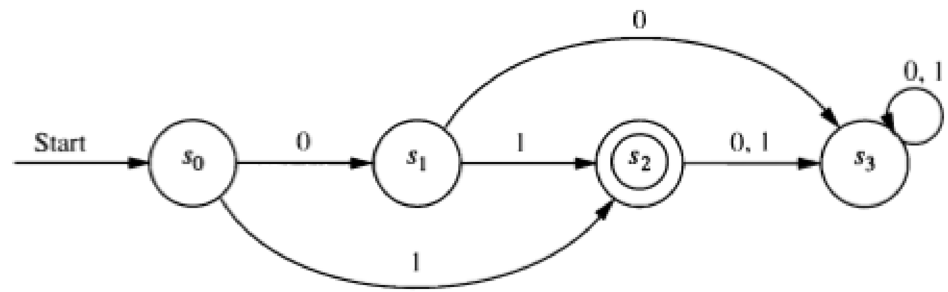
Language of an Automaton

Examples:

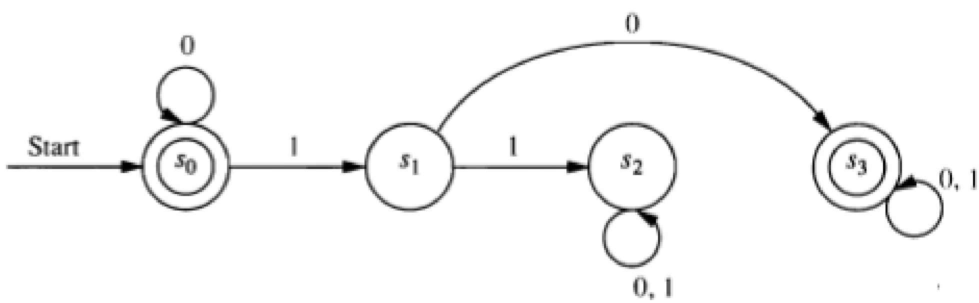
- $L(M_1) = \{1^n : n \geq 0\}$
- $L(M_2) = \{1, 01\}$
- $L(M_3) = \{0^n, 0^n10x : n \geq 0, x \in \Sigma^*\}$



M_1



M_2



M_3

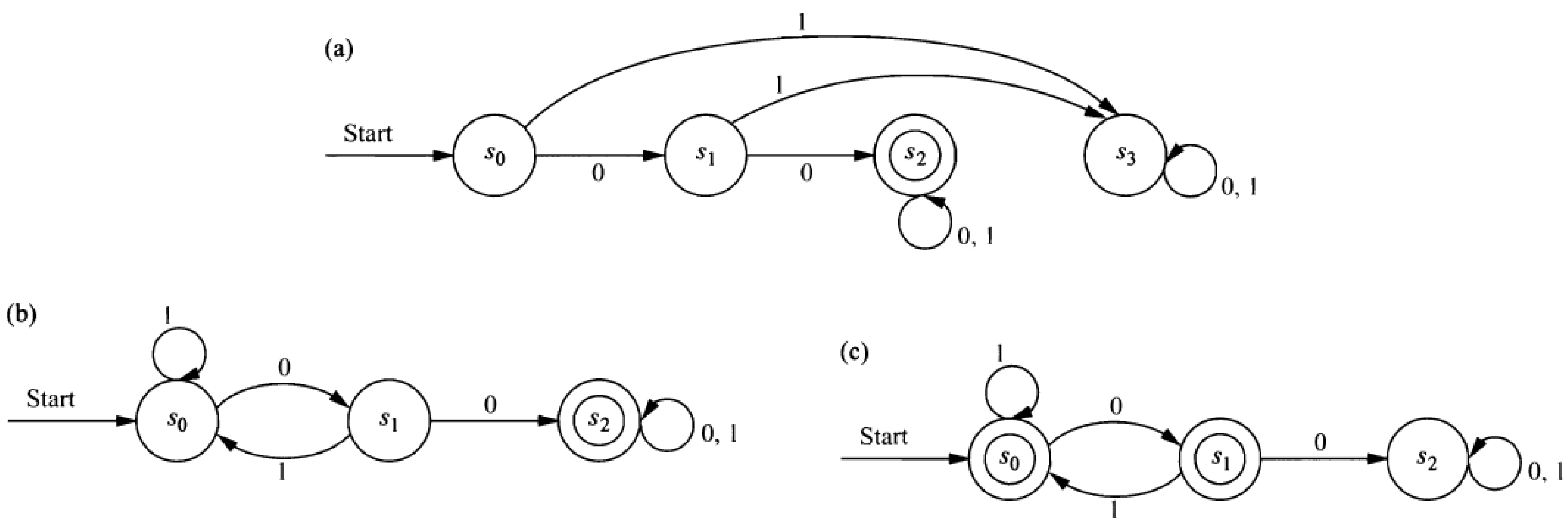
Constructing Automata

Problems:

(a) The set of bit strings that begin with two 0s

(b) The set of bit strings that contain two consecutive 0s

(c) The set of bit strings that do not contain two consecutive 0s



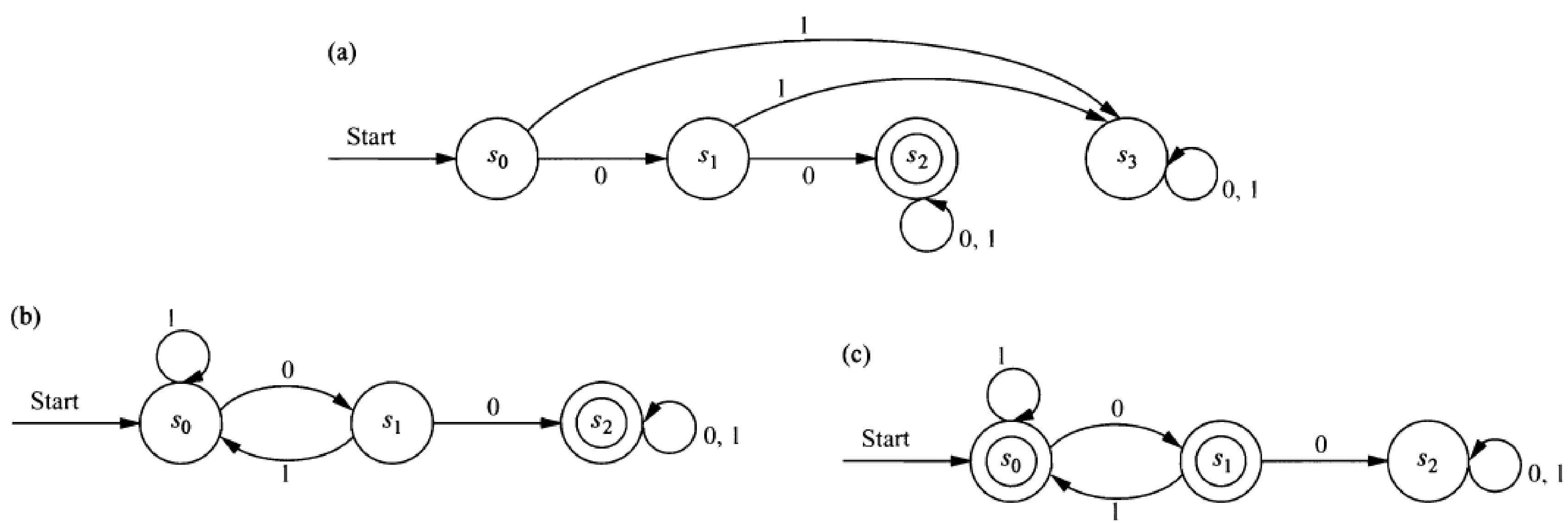
Constructing Automata

Problems:

(a) The set of bit strings that begin with two 0s

(b) The set of bit strings that contain two consecutive 0s

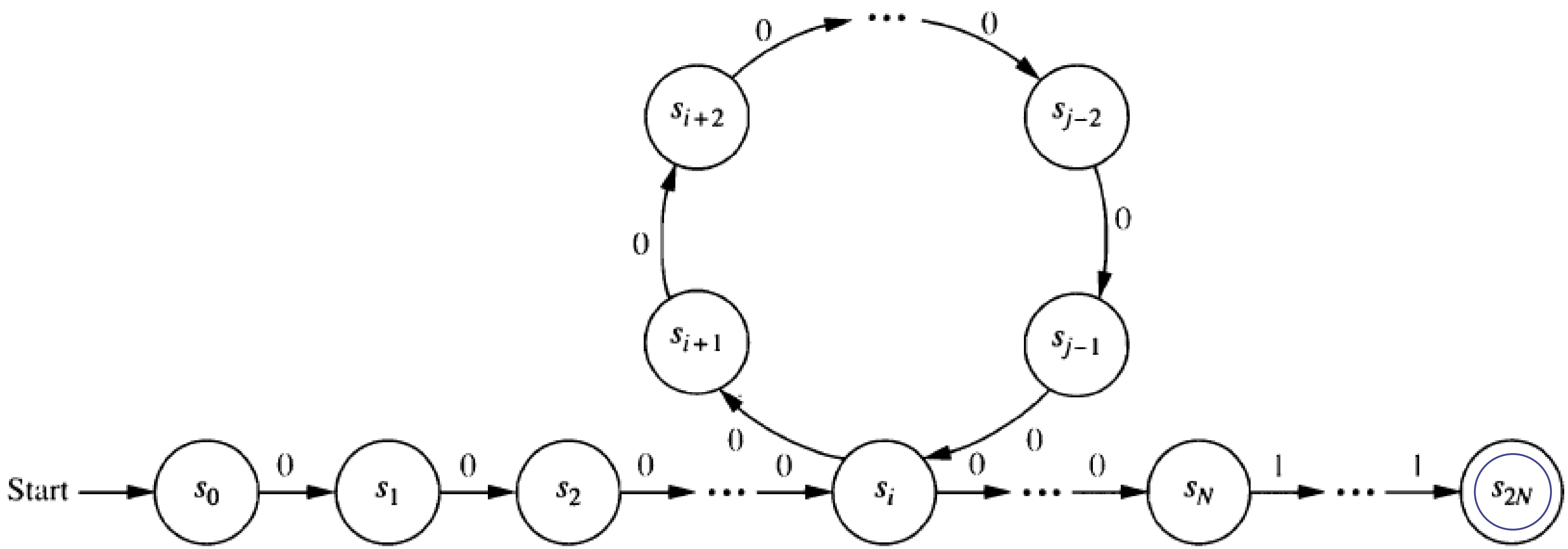
(c) The set of bit strings that do not contain two consecutive 0s



Pumping Lemma

Problem: Show that there is not any automaton whose language is $\{0^n 1^n\}$

- Assume there is such an automaton M.
- Let N be a number greater than #states
- When we feed 0^N into M, we get into a cycle
- This shows $0^i 0^{k(j-i)} 0^{N-j} 1^N$ will be accepted for any k while $i + k(j-i) + N - j \neq N$ for $k > 1$



Pumping Lemma

Theorem: Let L be the language of a finite-state automaton. There exists a number N such that any string w in L with length at least N can be written as $w = xyz$ satisfying the following condition:

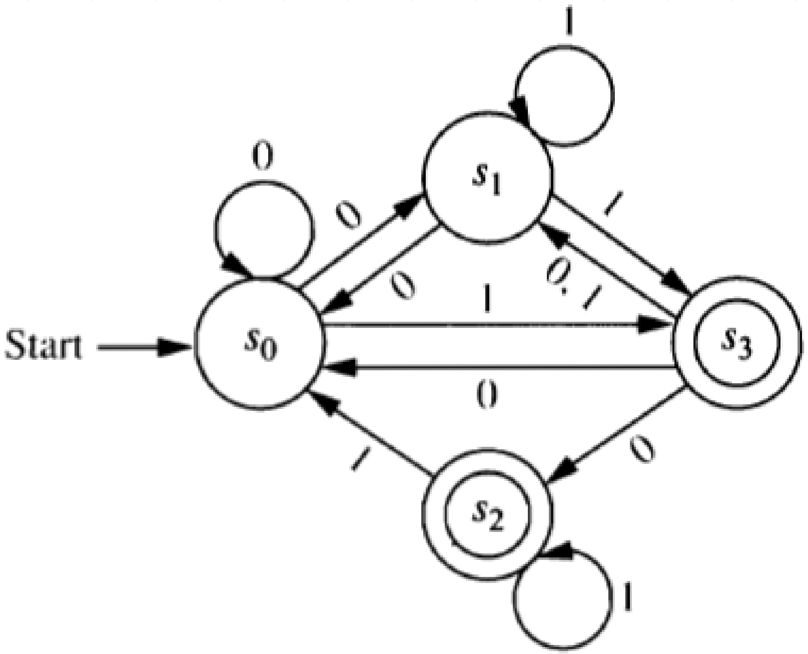
- $|y| \geq 1$
- $|xy| \leq N$
- $\forall n \geq 0: xy^n z \in L$

Nondeterministic Automata

The definition of a nondeterministic automaton is similar to that of a deterministic automaton except in the definition of its transition function which is

- $f : S \times \Sigma \rightarrow P(S)$

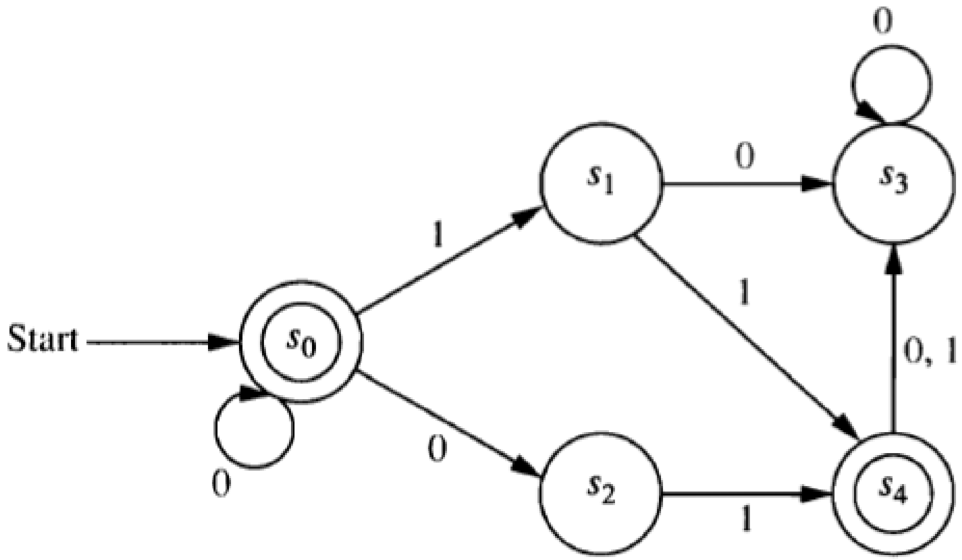
State	*f* Input 0	1
s_0	s_0, s_1	s_3
s_1	s_0	s_1, s_3
s_2		s_0, s_2
s_3	s_0, s_1, s_2	s_1



Nondeterministic Automata

Definition: The nondeterministic automaton M accept or recognize a string x if there is a final state in the set of all states that can be obtained from s_0 using x.

Example: The language of the following NDA is $\{0^n, 0^n01, 0^n11 : n \geq 0\}$



State	*f* Input 0	1
s_0	s_0, s_2	s_1
s_1	s_3	s_4
s_2		s_4
s_3	s_3	
s_4	s_3	s_3

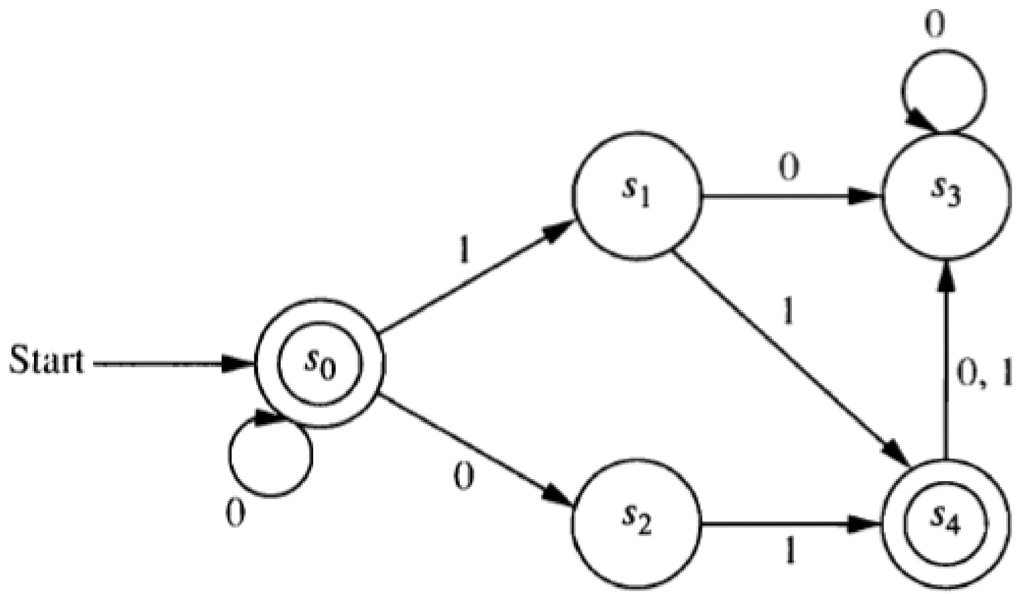
NDA to DA

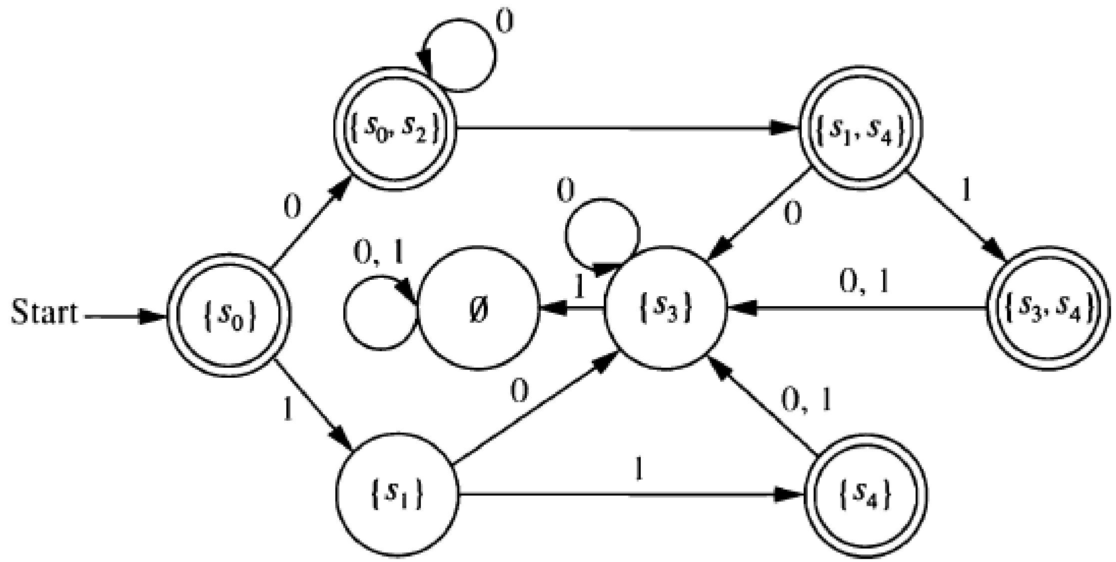
Theorem: If a language L is recognized by a NDA M_0, then L is recognized by a DA M_1.

- Each state in M_1 will be made up of a set of states in M_0.
- The input set of M_1 is the same as the input set of M_0
- The start state of M_1 is $\{s_0\}$
- Given a state $\{s_{i_1}, \ldots, s_{i_k}\}$ of M_1, the input symbol x takes this state to the union of the sets $f(s_{i_1}, \mathrm{x}), \ldots, \mathrm{f}(s_{i_k}, \mathrm{x})$.
- The final states of M_1 are those sets that contain a final state of M_0.
- There are as many as 2^n states in the DA, where n is #states in NDA.

NDA to DA

Example:





Regular Expressions

Definition: Regular expressions over alphabet Σ are defined recursively by:

- The symbol $\emptyset$ is a regular expression
- The symbol λ is a regular expression
- The symbol x is a regular expression whenever $x \in \Sigma$
- The symbols $(AB), (A \cup B)$ and A^* are regular expressions whenever A and B are regular expressions

Regular Expressions

Each regular expression represent a set specified by these rules:

- $\emptyset$ represents the empty set.
- λ represents the set $\{\lambda\}$
- x represents the set $\{x\}$
- (AB) represents the concatenation of the sets represented by A and by B
- $(A \cup B)$ represents the union of the sets represented by A and by B
- A^* represents the closure of the set represented by A, that is, $A^* = \{xa \colon x \in A^*, a \in A\}$

Definition: Sets represented by regular expressions are called regular sets.

Regular Expressions

Problem: What are the strings in the regular sets specified by the regular expressions

$10^*, (10)^*, 0 \cup 01, 0(0 \cup 1)^*$ and $(0^*1)^*$

Expression	*Strings*
$\mathbf{10^*}$	a 1 followed by any number of 0s (including no zeros)
$\mathbf{(10)^*}$	any number of copies of 10 (including the null string)
$\mathbf{0 \cup 01}$	the string 0 or the string 01
$\mathbf{0(0 \cup 1)^*}$	any string beginning with 0
$\mathbf{(0^*1)^*}$	any string not ending with 0

Regular Expressions

Problem: Find a regular expression that specifies each of these sets:

(a) The set of bit strings with even length

(b) The set of bit strings ending with a 0 and not containing 11

(c) The set of bit strings containing and odd number of 0s

Solution:

(a) $(00 \cup 01 \cup 10 \cup 11)^*$

(b) $(0 \cup 10)^*(0 \cup 10)$

(c) $1^*01^*(01^*01^*)^*$

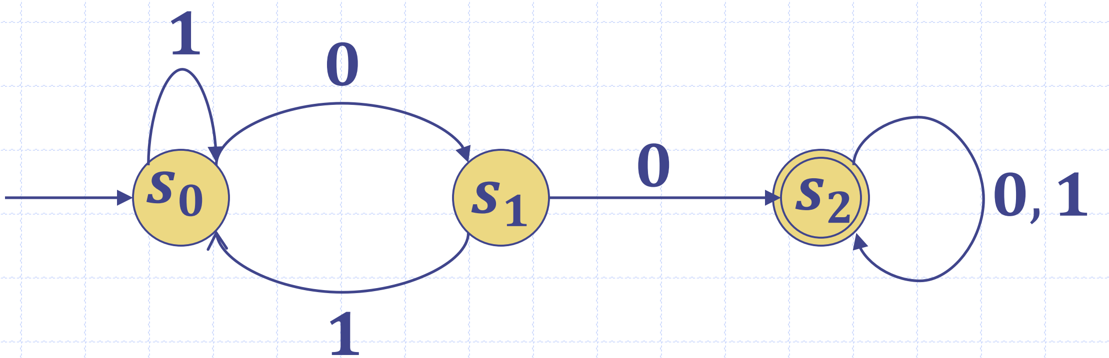
Kleene Theorem

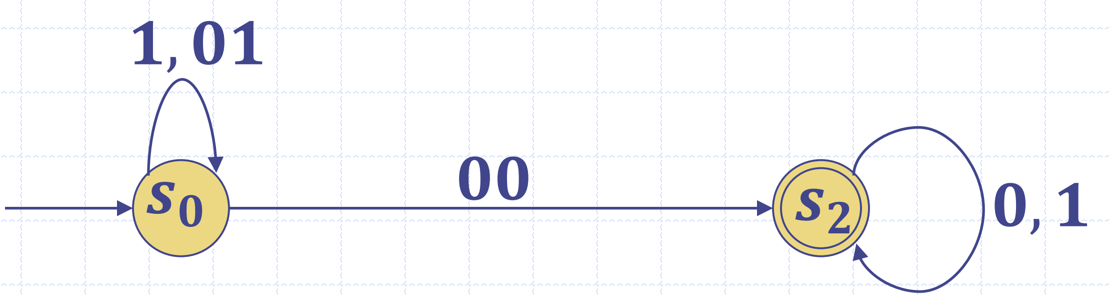
Theorem: *A set is regular iff it is recognized by a finite-state automaton*

Problem: Find the regular expression corresponding to the following automaton.

Solution:

- Remove s_1
- $(1 \cup 01)^* 00 (0 \cup 1)^*$





Grammars

Definition: A grammar $G = (V, T, S, P)$ consists of

- A vocabulary V
- A subset T of V (called terminal elements or alphabets, and usually denoted by lower case letters)
- A start symbol S
- A finite set of productions P of form $x \to y$ where x and y are strings on V

$V - T$ is called non-terminal symbols (usually denoted by capital letters). Each production must contain a non-terminal on its left side.

Example:

- $V = \{a, b, A, B, S\}$
- $T = \{a, b\}$ and Start symbol: S
- $P = \{S \to ABa, A \to BB, B \to ab, AB \to b\}$

Grammars

Definition: Consider a grammar $G = (V, T, S, P)$.

- Let $w_0 = lz_0r$ and $w_1 = lz_1r$ be string over V.
- If $z_0 \to z_1$ is a production in P, we say w_1 is directly derivable from w_0, and we write $w_0 \Rightarrow w_1$
- If $w_0, w_1, \ldots, w_n$ are strings over V such that $w_0 \Rightarrow w_1, w_1 \Rightarrow w_2, \ldots, w_{n-1} \Rightarrow w_n$, we say that w_n is derivable from w_0 and we write $w_0 \overset{*}{\Rightarrow} \mathrm{w_n}$.
- The sequence of steps used to obtain w_n from w_0 is called a derivation.

Example:

- $ABa \Rightarrow Aaba$
- $ABa \overset{*}{\Rightarrow} \mathrm{abababa}$ ($ABa \Rightarrow Aaba \Rightarrow \mathrm{BBaba} \Rightarrow Bababa \Rightarrow abababa$)

Grammars

Definition: The language of a grammar $G = (V, T, S, P)$ is $L(G) = \left\{w \in T^*: S \overset{*}{\Rightarrow} \mathrm{w}\right\}$.

Example: For $G = \{\{S, 0,1\}, \{0,1\}, S, \{S \rightarrow 11S, S \rightarrow 0\}$, we have $L(G) = \{0{,}110, 11110{,}1111110, \dots\}$

Example: For $G = \{\{S, A, a, b\}, \{a, b\}, S, \{S \rightarrow aA, S \rightarrow b, A \rightarrow aa\}$, we have $L(G) = \{b, aaa\}$

Example: For $G = \{\{S, 0,1\}, \{0,1\}, S, \{S \rightarrow 1S0, S \rightarrow \lambda\}$, we have $L(G) = \{0^n 1^n: n \geq 0\}$

Grammars

Problem: Find a grammar G whose language is $\{0^n 1^m : n, m \geq 0\}$

- $S \to 0S$
- $S \to 1A$
- $S \to 1$
- $A \to 1A$
- $A \to 1$
- $S \to \lambda$

Grammars

Problem: Find a grammar G whose language is $\{0^n 1^n 2^n : n \geq 0\}$

- $S \rightarrow C$
- $C \rightarrow 0CAB$
- $S \rightarrow \lambda$
- $BA \rightarrow AB$
- $0A \rightarrow 01$
- $1A \rightarrow 11$
- $1B \rightarrow 12$
- $2B \rightarrow 22$

Grammars

Problem: Find a grammar G whose language is fully parenthesized math expressions only including $+$ and $\times$

Consider G with the following properties

- $V = \{E, N, D, +, \times, (,), 0, 1, \ldots, 9\}$
- $T = \{(,), +, \times, 0, 1, \ldots, 9\}$
- Start symbol: E
- Productions:

$$E \to (E) \mid (E + E) \mid (E \times E) \mid N$$
$$N \to DN \mid D$$
$$D \to 0 \mid 1 \mid \ldots \mid 9$$

Types of Grammars

- Type 0: Has no restriction on its productions
- Type 1 (context-sensitive grammar): Has productions of forms $w_1 \rightarrow w_2$ where $w_1 = lAr$ and $w_2 = lwr$, where A is a non-terminal l and r are strings over V and w is a non-empty string over V.
- Type 2 (context-free grammar): Has productions of forms $w_1 \rightarrow w_2$ where w_1 is a single non-terminal
- Type 3 (regular grammar): Has productions of forms $w_1 \rightarrow w_2$ with $w_1 = A$ and either $w_2 = \mathrm{a}$ or $w_2 = aB$ where A and B are non-terminals.

Parsing

Problem: Determine whether the word $cbab$ belongs to the language of the following grammar.

- $S \to AB$
- $A \to Ca$
- $B \to Ba|Cb|b$
- $C \to cb|c$

Top-down parsing: beging with the start symbol and proceed by successively applying the productions

$$S \Rightarrow AB \Rightarrow \mathrm{CaB} \Rightarrow cbaB \Rightarrow \mathrm{cbab}$$

There is a bottom-up parsing trying to work backward.

Programing Languages

- Programing languages usually are modeled by context-free grammars.
- The compiler first parses your code to see whether your codes belongs to the programming language. Otherwise, it gives you “the syntax error”.

Pumping Lemma

Theorem: Let G be a context free grammar and let $L(G)$ be its language. There is a number N such that if z is a word in $L(G)$ with $|z| \geq N$, then z can be written as $uvwxy$ where $|vwx| \leq N, |\mathrm{vx}| \geq 1$ and $uv^iwx^iy \in L(G)$ for all non-negative integer i,

Problem: Prove that there is no context-free grammar G with $L(G) = \{0^n1^n2^n : n \geq 0\}$

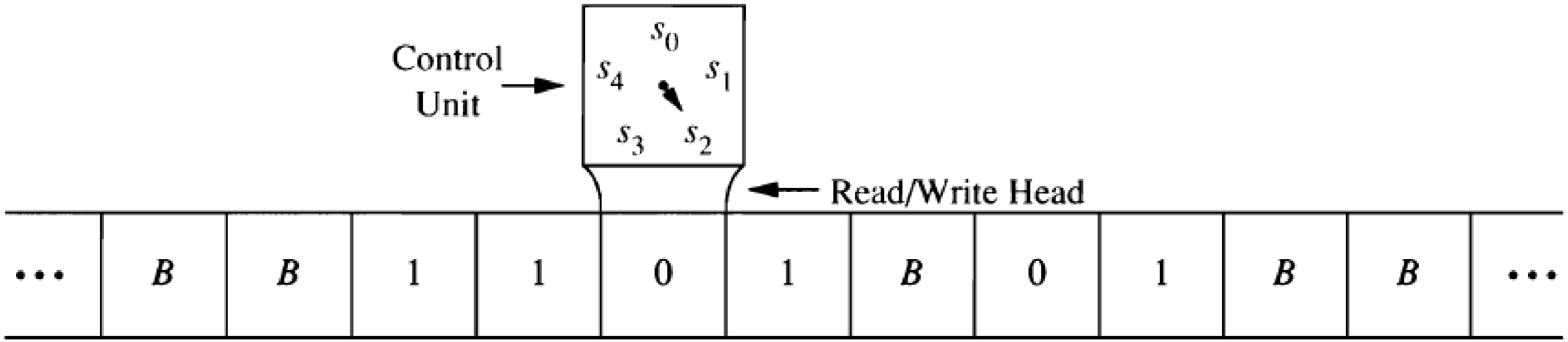
Turing Machine

Definition: A turing machine is a 7-tuple $M = (S, \Gamma, b, \Sigma, \delta, s_0, F)$

- S: states
- Γ: is a finite set of tape alphabet symbols
- $b \in \Gamma$: the blank symbol
- $\Sigma \subset \Gamma$: The input symbol
- $s_0 \in S$: the initial state
- $F \subset S$: the final states
- $\delta: S \times \Gamma \rightarrow S \times \Gamma \times \{L, R\}$: A partial function where L is left shift and R is right shift. If δ is not defined on the current state and the current tape symbol, then the machine halt.

Turing Machine

Definition: M accepts a string x over Σ if and only if M, starting in the initial position when x is written on the tape, halts in a final state.



Tape is infinite in both directions.
Only finitely many nonblank cells at any time.