Advanced Automata Theory 2
Finite Automata

Frank Stephan

Department of Computer Science

Department of Mathematics

National University of Singapore

fstephan@comp.nus.edu.sg

Repetition 1

Union:

$\mathbf{L \cup H = \{u : u \in L \vee u \in H\}}$;

$\mathbf{\{00, 01, 02\} \cup \{01, 11, 21\} = \{00, 01, 02, 11, 21\}}$;

$\mathbf{\{0, 00, 000\} \cup \{00, 000, 0000\} = \{0, 00, 000, 0000\}}$.

Intersection:

$\mathbf{L \cap H = \{u : u \in L \wedge u \in H\}}$;

$\mathbf{\{0, 00, 000\} \cap \{00, 000, 0000\} = \{00, 000\}}$;

$\mathbf{\{00, 01, 02\} \cap \{01, 11, 21\} = \{01\}}$.

Set Difference:

$\mathbf{L - H = \{u : u \in L \wedge u \notin H\}}$;

$\mathbf{\{00, 01, 02\} - \{01, 11, 21\} = \{00, 02\}}$.

Concatenation:

$\mathbf{000 \cdot 1122 = 0001122}$;

$\mathbf{L \cdot H = \{v \cdot w : v \in L \wedge w \in H\}}$;

$\mathbf{\{0, 00\} \cdot \{1, 2\} = \{01, 001, 02, 002\}}$.

Repetition 2

Definition

$\mathbf{L^* = \{\varepsilon\} \cup L \cup L \cdot L \cup L \cdot L \cdot L \cup \ldots}$

$\mathbf{= \{w_1 \cdot w_2 \cdot \ldots \cdot w_n : n \geq 0 \wedge w_1, w_2, \ldots, w_n \in L\}}$;

$\mathbf{L^+ = L \cup L \cdot L \cup L \cdot L \cdot L \cup \ldots}$

$\mathbf{= \{w_1 \cdot w_2 \cdot \ldots \cdot w_n : n > 0 \wedge w_1, w_2, \ldots, w_n \in L\}}$.

Examples

$\emptyset^* = \{\varepsilon\}$.

$\mathbf{\Sigma^*}$ is the set of all words over $\mathbf{\Sigma}$.

$\mathbf{\{0\}^* = \{\varepsilon, 0, 00, 000, 0000, \ldots\}}$.

$\mathbf{\{00, 01, 10, 11\}^*}$ are all binary words of even length.

$\varepsilon \in \mathbf{L^+}$ iff $\varepsilon \in \mathbf{L}$.

Notation

Often $\mathbf{w^*}$ in place of $\mathbf{\{w\}^*}$;

Often $\mathbf{w \cdot L}$ in place of $\mathbf{\{w\} \cdot L}$.

Repetition 3

Grammar $(\mathbf{N}, \mathbf{\Sigma}, \mathbf{P}, \mathbf{S})$ describes how to generate the words in a language; the language $\mathbf{L}$ of a grammar consists of all the words in $\mathbf{\Sigma}^*$ which can be generated.

$\mathbf{N}$: Non-terminal alphabet, disjoint to $\mathbf{\Sigma}$.

$\mathbf{S} \in \mathbf{N}$ is the start symbol.

$\mathbf{P}$ consists of rules $\mathbf{l} \to \mathbf{r}$ with each rule having at least one symbol of $\mathbf{N}$ in the word $\mathbf{l}$.

$\mathbf{v} \Rightarrow \mathbf{w}$ iff there are $\mathbf{x}, \mathbf{y}$ and rule $\mathbf{l} \to \mathbf{r}$ in $\mathbf{P}$ with $\mathbf{v} = \mathbf{xly}$ and $\mathbf{w} = \mathbf{xry}$. $\mathbf{v} \Rightarrow^* \mathbf{w}$: several such steps.

The grammar with $\mathbf{N} = \{\mathbf{S}\}$, $\mathbf{\Sigma} = \{\mathbf{0}, \mathbf{1}\}$ and $\mathbf{P} = \{\mathbf{S} \to \mathbf{SS}, \mathbf{S} \to \mathbf{0}, \mathbf{S} \to \mathbf{1}\}$ permits to generate all nonempty binary strings.

$\mathbf{S} \Rightarrow \mathbf{SS} \Rightarrow \mathbf{SSS} \Rightarrow \mathbf{0SS} \Rightarrow \mathbf{01S} \Rightarrow \mathbf{011}$.

Repetition 4

Grammar $(\mathbf{N}, \mathbf{\Sigma}, \mathbf{P}, \mathbf{S})$ generating $\mathbf{L}$.

CH0: No restriction. Generates all recursively enumerable languages.

CH1 (context-sensitive): Every rule is of the form $\mathbf{uAw} \to \mathbf{uvw}$ with $\mathbf{A} \in \mathbf{N}$, $\mathbf{u}, \mathbf{v}, \mathbf{w} \in (\mathbf{N} \cup \mathbf{\Sigma})^*$, $\mathbf{v} \neq \varepsilon$.
Easier formalisation: If $\mathbf{l} \to \mathbf{r}$ is a rule then $|\mathbf{l}| \leq |\mathbf{r}|$, that is, $\mathbf{r}$ is at least as long as $\mathbf{l}$.
Special rule for both in the case that $\varepsilon \in \mathbf{L}$.

CH2 (context-free): Every rule is of the form $\mathbf{A} \to \mathbf{w}$ with $\mathbf{A} \in \mathbf{N}$ and $\mathbf{w} \in (\mathbf{N} \cup \mathbf{\Sigma})^*$.

CH3 (regular): Every rule is of the form $\mathbf{A} \to \mathbf{wB}$ or $\mathbf{A} \to \mathbf{w}$ with $\mathbf{A}, \mathbf{B} \in \mathbf{N}$ and $\mathbf{w} \in \mathbf{\Sigma}^*$.

$\mathbf{L}$ is called context-sensitive / context-free / regular iff it can be generated by a grammar of respective type.

Multiples of 3

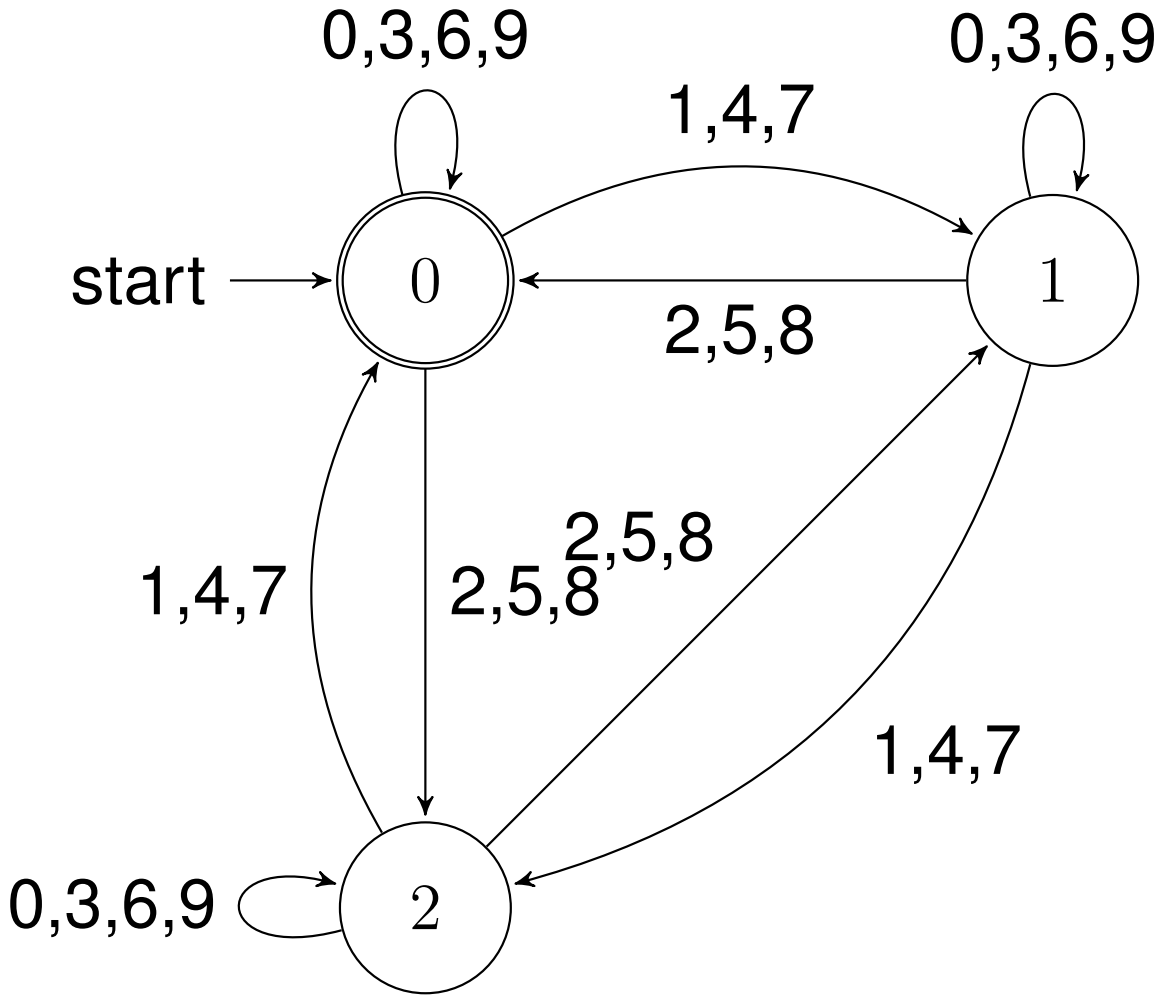
Check whether decimal number $\mathbf{a_1 a_2 \ldots a_n}$ is a multiple of $\mathbf{3}$.

Easy Algorithm

Scan through the word from $\mathbf{a_1}$ to $\mathbf{a_n}$.
Maintain memory $\mathbf{s}$.
Initialise $\mathbf{s = 0}$.
For $\mathbf{m = 1, 2, \ldots, n}$ Do
    Begin Let $\mathbf{s = s + a_m}$ modulo $\mathbf{3}$ End.
If $\mathbf{s = 0}$
    Then $\mathbf{a_1 a_2 \ldots a_n}$ is multiple of $\mathbf{3}$
    Else $\mathbf{a_1 a_2 \ldots a_n}$ is not a multiple of $\mathbf{3}$.

Quiz 2.2: Test the algorithm on $\mathbf{1}$, $\mathbf{20}$, $\mathbf{304}$, $\mathbf{2913}$, $\mathbf{49121}$, $\mathbf{391213}$, $\mathbf{2342342}$, $\mathbf{123454321}$, $\mathbf{1231231233}$, $\mathbf{12345654321}$.

Finite Automaton



Automata Working Mod 7

Automaton $(\{0,1,2,3,4,5,6\}, \{0,1,\ldots,9\}, \delta, 0, \{0\})$ with δ given as table.

q	type	$\delta(q,a)$ for $a=0$	1	2	3	4	5	6	7	8	9
0	acc	0	1	2	3	4	5	6	0	1	2
1	rej	3	4	5	6	0	1	2	3	4	5
2	rej	6	0	1	2	3	4	5	6	0	1
3	rej	2	3	4	5	6	0	1	2	3	4
4	rej	5	6	0	1	2	3	4	5	6	0
5	rej	1	2	3	4	5	6	0	1	2	3
6	rej	4	5	6	0	1	2	3	4	5	6

$\delta(\mathbf{q},\mathbf{a})$ is the remainder of $\mathbf{10 * q + a}$ by $\mathbf{7}$.
$\delta(\mathbf{0},\mathbf{568}) = \delta(\delta(\delta(\mathbf{0},\mathbf{5}),\mathbf{6}),\mathbf{8}) = \mathbf{1}$.

Automaton as Program

```
function div257 begin
  var a in {0,1,2,...,256};
  var b in {0,1,2,3,4,5,6,7,8,9};
  if exhausted(input) then reject;
  read(b,input); a = b;
  if b == 0 then
    begin if exhausted(input)
      then accept else reject end;
  while not exhausted(input) do
    begin read(b,input);
      a = (a*10+b) mod 257 end;
  if a == 0 then accept else reject end.
```

Automaton checks whether input is multiple of 257.
Automaton rejects leading 0s of decimal numbers.
Important: All variables can only store constantly many information during the run of the automaton.

Finite Automaton - Formal

A deterministic finite automaton (dfa) is given by a set $\mathbf{Q}$ of states, the alphabet $\mathbf{\Sigma}$ used, the state-transition function δ mapping $\mathbf{Q \times \Sigma}$ to $\mathbf{Q}$, the starting state $\mathbf{s \in Q}$ and a set $\mathbf{F \subseteq Q}$ of final states.

On input $\mathbf{a_1 a_2 \ldots a_n}$, one can associate to this input a sequence $\mathbf{q_0 q_1 q_2 \ldots q_n}$ of states of the finite automaton with $\mathbf{q_0 = s}$ and $\delta(\mathbf{q_m, a_{m+1}}) = \mathbf{q_{m+1}}$ for all $\mathbf{m < n}$. This sequence is called the run of the dfa on this input.

A dfa accepts a word $\mathbf{w}$ iff its run on the input $\mathbf{w}$ ends in an accepting state, that is, in a member of $\mathbf{F}$. Otherwise the dfa rejects the word $\mathbf{w}$.

One can inductively extend δ to a function from $\mathbf{Q \times \Sigma^*}$ to $\mathbf{Q}$ by letting $\delta(\mathbf{q}, \varepsilon) = \mathbf{q}$ and $\delta(\mathbf{q, wa}) = \delta(\delta(\mathbf{q, w}), \mathbf{a})$. So the dfa accepts $\mathbf{w}$ iff $\delta(\mathbf{s, w}) \in \mathbf{F}$.

Exercise 2.6

Make a finite automaton for the program from the Slide 9.

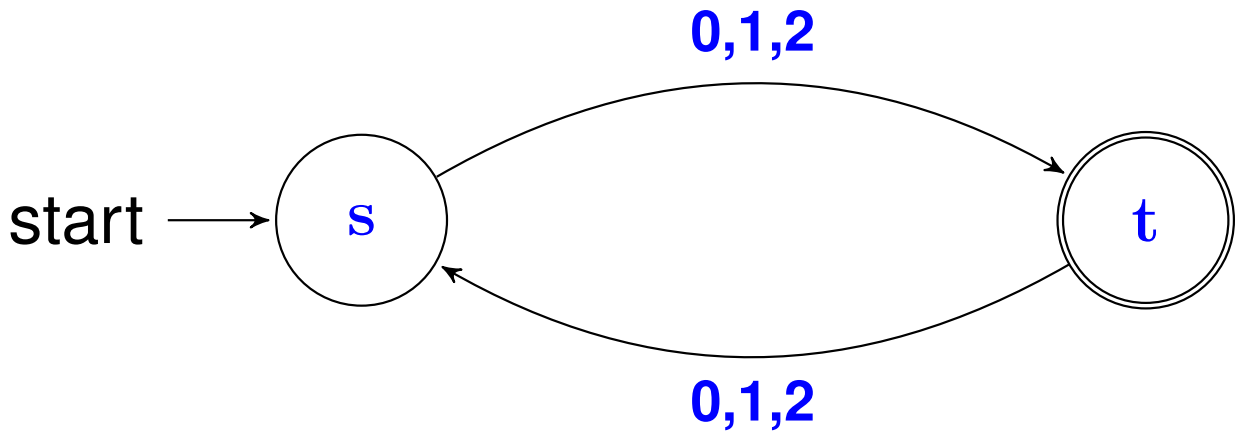
Use $\mathbf{Q} = \{\mathbf{s}, \mathbf{z}, \mathbf{r}, \mathbf{q_0}, \mathbf{q_1}, \ldots, \mathbf{q_{256}}\}$.

Here $\mathbf{s}$ is the starting state, $\mathbf{r}$ is an always rejecting state which is never left and $\mathbf{z}$ is the state which is reached after reading the first $\mathbf{0}$. Furthermore, when the word is starting with $\mathbf{1}, \mathbf{2}, \ldots, \mathbf{9}$, then the automaton should cycle between the states $\mathbf{q_0}, \mathbf{q_1}, \ldots, \mathbf{q_{256}}$.

Describe when the automaton is in state $\mathbf{q_a}$ and how the states are updated on $\mathbf{b}$. There is no need to write a table for δ, it is sufficient to say how δ works in each relevant case.

Quiz 2.7

Let $(\{\mathbf{s}, \mathbf{t}\}, \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}, \delta, \mathbf{s}, \{\mathbf{t}\})$ be a finite automaton with $\delta(\mathbf{s}, \mathbf{a}) = \mathbf{t}$ and $\delta(\mathbf{t}, \mathbf{a}) = \mathbf{s}$ for all $\mathbf{a} \in \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}$. Determine the language of strings recognised by this automaton.



Regular Sets

Theorem 2.8

The following statements are equivalent for a language $\mathbf{L}$.

(a) $\mathbf{L}$ is recognised by a deterministic finite automaton;

(b) $\mathbf{L}$ is generated by a regular expression;

(c) $\mathbf{L}$ is generated by a regular grammar.

Block Pumping Lemma (Thm 2.9)

If $\mathbf{L}$ is a regular set then there is a constant $\mathbf{k}$ such that for all strings $\mathbf{u_0}, \mathbf{u_1}, \ldots, \mathbf{u_k}$ with $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_{k-1}}$ not empty and $\mathbf{u_0 u_1 \ldots u_k} \in \mathbf{L}$ there are $\mathbf{i}, \mathbf{j}$ with $\mathbf{0 < i < j \leq k}$ and

$$\mathbf{(u_0 u_1 \ldots u_{i-1}) \cdot (u_i u_{i+1} \ldots u_{j-1})^* \cdot (u_j u_{j+1} \ldots u_k) \subseteq L.}$$

So if one splits a word in $\mathbf{L}$ into $\mathbf{k+1}$ parts then one can select some parts in the middle of the word and pump them.

Proof-Idea: Choose dfa and let $\mathbf{k}$ be two larger than number of states. Let $\mathbf{q_h}$ be the state after reading $\mathbf{u_0 u_1 \ldots u_{h-1}}$. There are $\mathbf{i}, \mathbf{j}$ with $\mathbf{q_i = q_j}$. Then after reading $\mathbf{u_0 \ldots u_{i-1}}$, one is in state $\mathbf{q_i}$ and can read as many copies of $\mathbf{u_i u_{i+1} \ldots u_{j-1}}$ as one wants without changing the state. Thus the state after reading the whole word is for all words in $\mathbf{H}$ the same. So the selected blocks can be pumped up or down.

Example 2.10

$\{1,2\}^* \cdot (0 \cdot \{1,2\}^* \cdot 0 \cdot \{1,2\}^*)^*$: Satisfies Block Pumping Lemma with $k = 3$.

If there are at least two inner blocks, either one of them contains an even number of 0 or two neighbouring blocks contain both an odd number of 0. Thus one can pump one or two neighbouring inner blocks which have together an even number of 0.

$\{u : u$ has a different number of 0s than 1s$\}$: Does not satisfy the Block Pumping Lemma with any k.

Consider $0^{k+1}1^{(k+1)!+k+1}$ and cut it in blocks such that each inner block consists of a single 0. Thus the pump, whatever it is, has h symbols with $1 \leq h \leq k$ and inserting $(k+1)!/h$ copies of the pump produces a word with as many 0 as 1.

Sequence of Morse and Thue

Definition

An infinite sequence $\mathbf{a_0 a_1 a_2 \ldots}$ is square-free if and only if it does not have a subword of the form $\mathbf{ww}$ and $\mathbf{a_0 a_1 a_2 \ldots}$ is cube-free if and only if it does not have a subword of the form $\mathbf{www}$.

Theorem 2.12 [Morse and Thue]

The sequence given by $\mathbf{a_0 = 0}$, $\mathbf{a_{2n} = a_n}$, $\mathbf{a_{2n+1} = 1 - a_n}$ is an infinite binary cube-free sequence.
There is an infinite ternary square-free sequence.

There is no infinite binary square-free sequence: Such a sequence cannot have the subwords $\mathbf{00}$ and $\mathbf{11}$, hence it must alternate at every bit and start with $\mathbf{0101}$ or $\mathbf{1010}$.

Corollary

There are cube-free binary strings of every length.

Block Pumping and Regularity

Theorem 2.11 [Ehrenfeucht, Parikh and Rozenberg]
If a language and its complement both satisfy the Block Pumping Lemma then the language is regular.

Theorem 2.13
The following languages satisfy the Block Pumping Lemma but are not regular:

- $\mathbf{L} = \{\mathbf{w} \in \{\mathbf{0}, \mathbf{1}\}^* : \mathbf{w}$ contains a cube or the length of $\mathbf{w}$ is not a power of $\mathbf{10}\}$;
- $\mathbf{H} = \{\mathbf{w} \in \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}^* : \mathbf{w}$ contains a square or the length of $\mathbf{w}$ is not a power of $\mathbf{10}\}$.

The idea is based on the fact that when pumping, all the long repetition of the pump contain a square and a cube. For small pumping, omitting the pump or repeating it once, one uses the length-constraint to satisfy the Block Pumping Lemma.

Quiz 2.14

Which of the following languages over $\Sigma = \{0, 1, 2, 3\}$ satisfies the pumping-condition of the Block Pumping Lemma:

(a) $\{00, 111, 22222\}^* \cap \{11, 222, 00000\}^* \cap \{22, 000, 11111\}^*$,

(b) $\{0^m 1^n 2^o : m + n + o = 5555\}$,

(c) $\{0^m 1^n 2^o : m + n = o + 5555\}$,

(d) $\{w : w$ contains more 1 than $0\}$?

Exercises 2.15 and 2.16

Find the optimal block pumping constants for the following languages.

Exercise 2.15

(a) $\{w \in \{0,1,2,3,4,5,6,7,8,9\}^* :$ at least one nonzero digit a occurs in w at least three times$\}$;

(b) $\{w \in \{0,1,2,3,4,5,6,7,8,9\}^* : |w| = 255\}$;

(c) $\{w \in \{0,1,2,3,4,5,6,7,8,9\}^* :$ the length $|w|$ is not a multiple of $6\}$.

Exercise 2.16

(a) $\{w \in \{0,1,2,3,4,5,6,7,8,9\}^* : w$ is a multiple of $25\}$;

(b) $\{w \in \{0,1,2,3,4,5,6,7,8,9\}^* : w$ is not a multiple of $3\}$;

(c) $\{w \in \{0,1,2,3,4,5,6,7,8,9\}^* : w$ is a multiple of $400\}$.

Derivatives

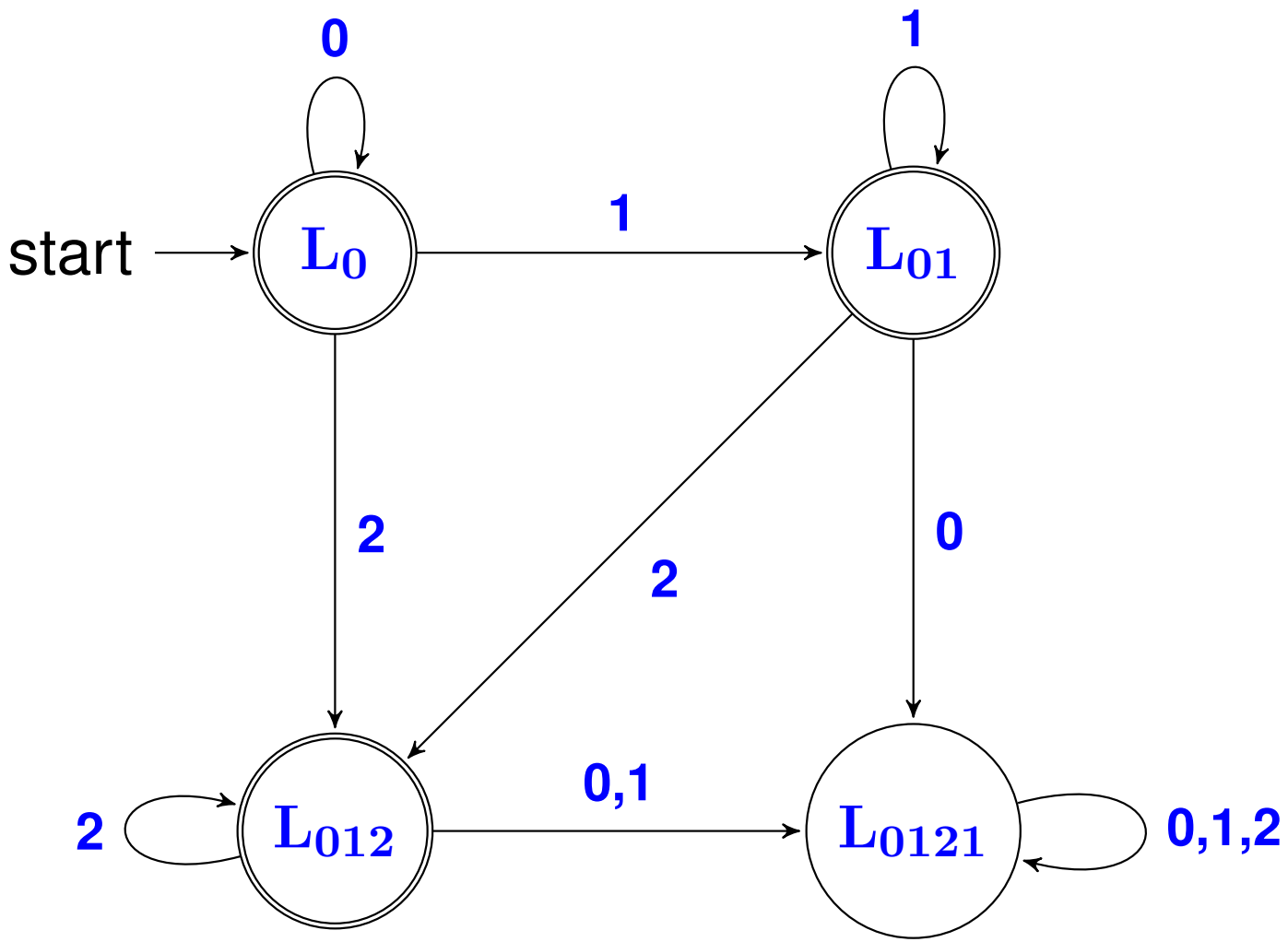
Given a language $\mathbf{L}$, let $\mathbf{L_x} = \{\mathbf{y} : \mathbf{x} \cdot \mathbf{y} \in \mathbf{L}\}$ be the derivative of $\mathbf{L}$ at $\mathbf{x}$.

Theorem 2.19 [Myhill and Nerode].
A language $\mathbf{L}$ is regular iff $\mathbf{L}$ has only finitely many derivatives.

If $\mathbf{L}$ has $\mathbf{k}$ derivatives, one can make a dfa by selecting strings $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}$ representing the derivatives $\mathbf{L_{x_1}}, \mathbf{L_{x_2}}, \ldots, \mathbf{L_{x_k}}$ and letting $\delta(\mathbf{x_i}, \mathbf{a})$ be the unique $\mathbf{x_j}$ with $\mathbf{L_{x_j}} = \mathbf{L_{x_i a}}$. A state $\mathbf{x_i}$ is accepting iff $\varepsilon \in \mathbf{L_{x_i}}$ iff $\mathbf{x_i} \in \mathbf{L}$.

Example 2.21

Let $\mathbf{L} = 0^*1^*2^*$. Now $\mathbf{L_0} = 0^*1^*2^*$, $\mathbf{L_{01}} = 1^*2^*$, $\mathbf{L_{012}} = 2^*$ and $\mathbf{L_{0121}} = \emptyset$. The corresponding automaton is the following.



Example 2.22

Let $\mathbf{L} = \{\mathbf{0^n 1^n} : \mathbf{n} \in \mathbb{N}\}$.

Then $\mathbf{L_{0^n}} = \{\mathbf{0^m 1^{n+m}} : \mathbf{m} \in \mathbb{N}\}$.

The shortest string in $\mathbf{L_{0^n}}$ is $\mathbf{1^n}$.

If $\mathbf{n} \neq \mathbf{n'}$ then $\mathbf{L_{0^n}} \neq \mathbf{L_{0^{n'}}}$. Hence there are infinitely many different derivatives.

The language $\mathbf{L}$ cannot be regular.

Jaffe's Pumping Lemma 2.23

Jaffe was the first to provide a form of the pumping lemma which characterises the regular languages.

Lemma 2.23

A language $\mathbf{L} \subseteq \mathbf{\Sigma}^*$ is regular iff $\exists \mathbf{k}\ \forall \mathbf{x} \in \mathbf{\Sigma}^* \forall \mathbf{y} \in \mathbf{\Sigma}^{\mathbf{k}}\ \exists \mathbf{u}, \mathbf{v}, \mathbf{w}$ $[\mathbf{v} \neq \varepsilon$ and $\mathbf{uvw} = \mathbf{y}$ and all $\mathbf{h} \in \mathbb{N}$ satisfy $\mathbf{L_{xuv^h w}} = \mathbf{L_{xy}}]$.

Note that $\mathbf{L_{xuw}} = \mathbf{L_{xy}}$ implies that every derivative of a string of length $\mathbf{k}$ or more is equal to a shorter derivative. Thus there are at most $(|\mathbf{\Sigma}|^{\mathbf{k}} - \mathbf{1})/(|\mathbf{\Sigma}| - \mathbf{1})$ in the case $|\mathbf{\Sigma}| > \mathbf{1}$ and $\mathbf{k}$ in the case that $|\mathbf{\Sigma}| = \mathbf{1}$ many derivatives and therefore the language is regular by the Theorem of Myhill and Nerode. The other direction is proven by looking at the dfas.

Exercises 2.24 and 2.25

Exercise 2.24: Assume that the alphabet Σ has 5000 elements. Define a language $L \subseteq \Sigma^*$ such that Jaffe's Matching Pumping Lemma is satisfied with constant $k = 3$ while every deterministic finite automaton recognising L has more than 5000 states. Prove your answer.

Exercise 2.25: Find a language which needs for Jaffe's Matching Pumping Lemma at least constant $k = 100$ and can be recognised by a deterministic finite automaton with 100 states. Prove your answer.

Corollary 2.26

Jaffe's Pumping Lemma for members of L

If $\mathbf{L}$ is regular then there is a constant $\mathbf{k}$ such that for all $\mathbf{x} \in \mathbf{\Sigma}^*$ and $\mathbf{y} \in \mathbf{\Sigma^k}$ with $\mathbf{xy} \in \mathbf{L}$ there are $\mathbf{u}, \mathbf{v}, \mathbf{w}$ with $\mathbf{y} = \mathbf{uvw}$ and $\mathbf{v} \neq \varepsilon$ such that, for all $\mathbf{h} \in \mathbb{N}$, $\mathbf{L_{xuv^hw}} = \mathbf{L_{xy}}$.

Exercise 2.27: Show that $\mathbf{L}$ consisting of ε and all words $\mathbf{0^n 1^m 2^k 3}$ with $\mathbf{n = m}$ or $\mathbf{k = 0}$ is context-free, not regular and satisfies Corollary 2.26.

Exercise 2.28: If $\mathbf{L}$ satisfies Corollary 2.26 and $\mathbf{H}$ is regular, does $\mathbf{L \cdot H}$ satisfy Corollary 2.26? Prove the answer.

Exercise 2.29: Call $\mathbf{L}$ prefix-free iff $\mathbf{vw} \in \mathbf{L}$ and $\mathbf{w} \neq \varepsilon$ always implies $\mathbf{v} \notin \mathbf{L}$. If $\mathbf{L}$ is prefix-free and $\mathbf{L^{mi}}$ satisfies Theorem 1.19 (a), does $\mathbf{L}$ then satisfy Corollary 2.26?

Example 2.31

Assume that Σ has n elements, $n > 0$. Let L consist of all strings which contain at least one symbol twice.

If $\varepsilon \in L_x$ then $L_x = \Sigma^*$.

If $\varepsilon \notin L_x$ then $L_x \cap \Sigma = \{a : a \text{ occurs in } x\}$.

There are $2^n + 1$ many derivatives of this type; for each subset of Σ one derivative with $L_x \cap \Sigma$ being that set plus Σ^*.

These are also all the derivatives which exist. A dfa recognising L needs at least $2^n + 1$ states.

Non-Deterministic Finite Automaton

If $(\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{s}, \mathbf{F})$ is a non-deterministic finite automaton (nfa) then δ is a relation and not a function, that is, for $\mathbf{q} \in \mathbf{Q}$ and $\mathbf{a} \in \mathbf{\Sigma}$ there can be several $\mathbf{p} \in \mathbf{Q}$ with $(\mathbf{q}, \mathbf{a}, \mathbf{p}) \in \delta$.

A run of an nfa on a word $\mathbf{a_1 a_2 \ldots a_n}$ is a sequence $\mathbf{q_0 q_1 q_2 \ldots q_n} \in \mathbf{Q}^*$ such that $\mathbf{q_0} = \mathbf{s}$ and $(\mathbf{q_m}, \mathbf{a_{m+1}}, \mathbf{a_{m+1}}) \in \delta$ for all $\mathbf{m} < \mathbf{n}$.

If $\mathbf{q_n} \in \mathbf{F}$ then the run is "accepting" else the run is "rejecting".

The nfa accepts a word $\mathbf{w}$ iff it has an accepting run on $\mathbf{w}$; this is also the case if there exist other rejecting runs.

Büchi's Powerset Construction

Theorem 2.34

If $\mathbf{L}$ can be recognised by an nfa with $\mathbf{m}$ states then $\mathbf{L}$ can be recognised by a dfa with $\mathbf{2^m}$ states.

Construction

Given $(\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{s}, \mathbf{F})$, let $\mathbf{Pow(Q)}$ be the set of all subsets of $\mathbf{Q}$. For $\tilde{\mathbf{p}} \subseteq \mathbf{Q}$, let

$$\mathbf{\Delta}(\tilde{\mathbf{p}}, \mathbf{a}) = \{\mathbf{q} \in \mathbf{Q} : \exists \mathbf{p} \in \tilde{\mathbf{p}}\,[(\mathbf{p}, \mathbf{a}, \mathbf{q}) \in \delta]\}.$$

The dfa $(\mathbf{Pow(Q)}, \mathbf{\Sigma}, \mathbf{\Delta}, \{\mathbf{s}\}, \{\tilde{\mathbf{p}} : \tilde{\mathbf{p}} \cap \mathbf{F} \neq \emptyset\})$ recognises the same language $\mathbf{L}$.

Idea of Verification

Show that both automata have same acceptance behaviour on words $\mathbf{a_1 a_2 \ldots a_n}$ by induction over word length.

Example 2.35

Consider nfa $(\{\mathbf{s}, \mathbf{q}\}, \{\mathbf{0}, \mathbf{1}\}, \delta, \mathbf{s}, \{\mathbf{q}\})$ with $\delta(\mathbf{s}, \mathbf{0}) = \{\mathbf{s}, \mathbf{q}\}$, $\delta(\mathbf{s}, \mathbf{1}) = \{\mathbf{s}\}$ and $\delta(\mathbf{q}, \mathbf{a}) = \emptyset$ for all $\mathbf{a} \in \{\mathbf{0}, \mathbf{1}\}$.

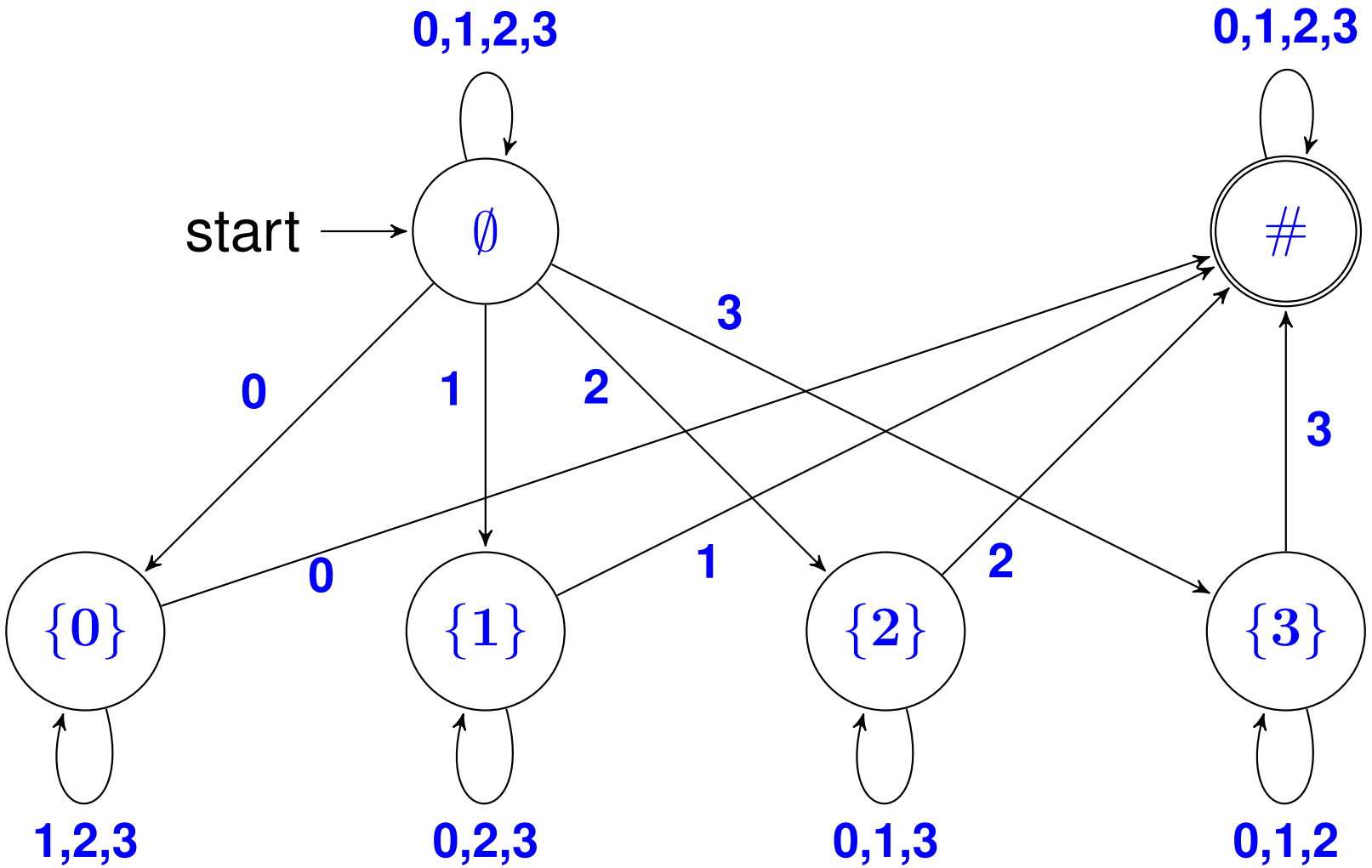
Then the corresponding dfa has the four states $\emptyset, \{\mathbf{s}\}, \{\mathbf{q}\}, \{\mathbf{s}, \mathbf{q}\}$ where $\{\mathbf{q}\}, \{\mathbf{s}, \mathbf{q}\}$ are the final states and $\{\mathbf{s}\}$ is the initial state. The transition function $\mathbf{\Delta}$ of the dfa is given as

$\mathbf{\Delta}(\emptyset, \mathbf{a}) = \emptyset$ for $\mathbf{a} \in \{\mathbf{0}, \mathbf{1}\}$,
$\mathbf{\Delta}(\{\mathbf{s}\}, \mathbf{0}) = \{\mathbf{s}, \mathbf{q}\}$, $\mathbf{\Delta}(\{\mathbf{s}\}, \mathbf{1}) = \{\mathbf{s}\}$,
$\mathbf{\Delta}(\{\mathbf{q}\}, \mathbf{a}) = \emptyset$ for $\mathbf{a} \in \{\mathbf{0}, \mathbf{1}\}$,
$\mathbf{\Delta}(\{\mathbf{s}, \mathbf{q}\}, \mathbf{0}) = \{\mathbf{s}, \mathbf{q}\}$, $\mathbf{\Delta}(\{\mathbf{s}, \mathbf{q}\}, \mathbf{1}) = \{\mathbf{s}\}$.

This automaton can be further optimised: The states $\emptyset$ and $\{\mathbf{q}\}$ are never reached, hence they can be omitted from the dfa.

Exponential Bound

The language from Example 2.31 has an nfa with $n + 2$ states while a dfa needs $2^n + 1$ states; here for $n = 4$.



Exercises 2.36 and 2.37

Exercise 2.36

Consider the language $\{0,1\}^* \cdot 0 \cdot \{0,1\}^{n-1}$:

(a) Show that a dfa recognising it needs at least 2^n states;

(b) Make an nfa recognising it with at most $n+1$ states;

(c) Made a dfa recognising it with exactly 2^n states.

Exercise 2.37

Find a characterisation when a regular language L is recognised by an nfa only having accepting states. Examples of such languages are $\{0,1\}^*$, $0^*1^*2^*$ and $\{1,01,001\}^* \cdot 0^*$. The language $\{00,11\}^*$ is not a language of this type.

Set of Initial States

Assume that $(\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{I}, \mathbf{F})$ has a set $\mathbf{I}$ of possible initial states and an accepting run is any run starting in one member of $\mathbf{I}$ and finishing in one member of $\mathbf{F}$.

Exercise 2.39
Consider $\mathbf{L} = \{\mathbf{w} :$ some $\mathbf{a} \in \mathbf{\Sigma}$ does not occur in $\mathbf{w}\}$.

Show that there is an nfa with an initial set of states which recognises $\mathbf{L}$ using $|\mathbf{\Sigma}|$ states.

Show that every complete dfa recognising $\mathbf{L}$ needs $\mathbf{2}^{|\mathbf{\Sigma}|}$ states; here complete means that the dfa never gets stuck.

Regular Grammar to NFA

Given a regular grammar over alphabet $\mathbf{\Sigma}$.
While there is $\mathbf{A \to w}$ with $\mathbf{w \in \Sigma^+}$, replace rule by
$\mathbf{A \to wC}, \mathbf{C} \to \varepsilon$ for new non-terminal $\mathbf{C}$.
While there is $\mathbf{A \to vwB}$ with $\mathbf{v, w \in \Sigma^+}$, replace rule by
$\mathbf{A \to vC}, \mathbf{C \to wB}$ for new non-terminal $\mathbf{C}$.
Fix now the so normalised grammar as $(\mathbf{N}, \mathbf{\Sigma}, \mathbf{P}, \mathbf{S})$.

NFA is given as $(\mathbf{N}, \mathbf{\Sigma}, \delta, \mathbf{S}, \mathbf{F})$ with

$$\begin{aligned} \delta(\mathbf{A}, \mathbf{a}) &= \{\mathbf{B} \in \mathbf{N} : \mathbf{A} \Rightarrow^* \mathbf{aB}\}; \\ \mathbf{F} &= \{\mathbf{B} \in \mathbf{N} : \mathbf{B} \Rightarrow^* \varepsilon\}. \end{aligned}$$

The NFA recognises the same language which the given grammar generates.

Exercises 2.43 and 2.44

Exercise 2.43. Let the regular grammar $(\{\mathbf{S}, \mathbf{T}\}, \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}, \mathbf{P}, \mathbf{S})$ with the rules $\mathbf{P}$ being $\mathbf{S} \to \mathbf{01T}|\mathbf{20S}$, $\mathbf{T} \to \mathbf{01}|\mathbf{20S}|\mathbf{12T}$. Construct a non-deterministic finite automaton recognising the language generated by this grammar.

Exercise 2.44. Consider the regular grammar $(\{\mathbf{S}\}, \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{6}, \mathbf{7}, \mathbf{8}, \mathbf{9}\}, \mathbf{P}, \mathbf{S})$ where the rules in $\mathbf{P}$ are all rules of the form $\mathbf{S} \to \mathbf{aaaaaS}$ for some digit $\mathbf{a}$ and the rule $\mathbf{S} \to \varepsilon$ and let $\mathbf{L}$ be the language generated by this grammar. What is the minimum number of states of a non-deterministic finite automaton recognising this language $\mathbf{L}$? What is the trade-off of the nfa compared to the minimal dfa for the same language $\mathbf{L}$? Prove the answers.

Characterisation of Regular Sets

Corollary 2.45

The following conditions are equivalent for a language $\mathbf{L}$:

(a) $\mathbf{L}$ is generated by a regular expression;

(b) $\mathbf{L}$ is generated by a regular grammar;

(c) $\mathbf{L}$ is recognised by a dfa;

(d) $\mathbf{L}$ is recognised by a nfa;

(e) $\mathbf{L}$ and $\Sigma^* - \mathbf{L}$ satisfy the Block Pumping Lemma;

(f) $\mathbf{L}$ satisfies Jaffe's Matching Pumping Lemma;

(g) $\mathbf{L}$ has only finitely many derivatives (Theorem of Myhill and Nerode).

Sizes of Expressions versus NFAs

Example 2.46. $\mathbf{L = \bigcup_{m<n}(\{0,1\}^m \cdot \{1\} \cdot \{0,1\}^* \cdot \{10^m\})}$ can be written down in $\mathbf{O(n^2)}$ symbols as a regular expression but the corresponding dfa has at least $\mathbf{2^n}$ states: if $\mathbf{x = a_0 a_1 \ldots a_{n-1}}$ then $\mathbf{10^m \in L_x}$ iff $\mathbf{x10^m \in L}$ iff $\mathbf{a_0 a_1 \ldots a_{n-1} 10^m \in L}$ iff $\mathbf{a_m = 1}$. Thus for $\mathbf{x = a_0 a_1 \ldots a_{n-1}}$ and $\mathbf{y = b_0 b_1 \ldots b_{n-1}}$, it holds that $\mathbf{L_x = L_y}$ iff $\mathbf{\forall m < n\,[10^m \in L_x \Leftrightarrow 10^m \in L_y]}$ iff $\mathbf{\forall m < n\,[a_m = b_m]}$ iff $\mathbf{x = y}$. Thus the language $\mathbf{L}$ has at least $\mathbf{2^n}$ derivatives and therefore a dfa for $\mathbf{L}$ needs at least $\mathbf{2^n}$ states.

Theorem 2.47. $\mathbf{L_n = \{0^{p_1}\}^+ \cap \{0^{p_2}\}^+ \cap \ldots \cap \{0^{p_n}\}^+}$ has a regular expression which can be written down with approximately $\mathbf{O(n^2 \log(n))}$ symbols if one can use intersection. However, every nfa recognising $\mathbf{L_n}$ has at least $\mathbf{2^n}$ states and every regular expression for $\mathbf{L_n}$ only using union, concatenation and Kleene star needs at least $\mathbf{2^n}$ symbols.

Exercises 2.48 and 2.49

Exercise 2.48

Assume that a regular expression uses lists of finite sets, Kleene star, union and concatenation and assume that this expression generates at least two words. Prove that the second-shortest word of the language generated by σ is at most as long as σ. Either prove it by structural induction or by an assumption of contradiction as in the proof before; both methods are nearly equivalent.

Exercise 2.49

Is Exercise 2.48 also true if one permits Kleene plus in addition to Kleene star in the regular expressions? Either provide a counter example or adjust the proof. In the case that it is not true for the bound $|\sigma|$, is it true for the bound $\mathbf{2}|\sigma|$? Again prove that bound or provide a further counter example.

Example 2.50

Ehrenfeucht and Zeiger's Exponential Gap

Let $\Sigma = \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$. A complete dfa with $n + 1$ states recognises the set of all sequences of the form $(1, a_1), (a_1, a_2), (a_2, a_3), \ldots, (a_{m-1}, a_m)$ for any numbers $a_1, a_2, \ldots, a_m$. Ehrenfeucht and Zeiger showed that any regular expression for this language needs at least 2^{n-1} symbols.

If one would permit intersection, this gap would not be there for this example, as one could write

$$(\{(a, b) \cdot (b, c) : a, b, c \in \{1, 2, \ldots, n\}\}^* \cdot (\varepsilon \cup \{(a, b) : a, b \in \{1, 2, \ldots, n\}\})) \cap (\{(a, b) : a, b \in \{1, 2, \ldots, n\}\} \cdot \{(a, b) \cdot (b, c) : a, b, c \in \{1, 2, \ldots, n\}\}^* \cdot (\varepsilon \cup \{(a, b) : a, b \in \{1, 2, \ldots, n\}\}))$$

to obtain the desired expression whose size is polynomial in n.

Exercises 2.51-2.52

Exercise 2.51.

Assume that an nfa of $\mathbf{k}$ states recognises a language $\mathbf{L}$. Show that the language does then satisfy the Block Pumping Lemma with constant $\mathbf{k+1}$, that is, given any words $\mathbf{u_0, u_1, \ldots, u_k, u_{k+1}}$ such that their concatenation $\mathbf{u_0 u_1 \ldots u_k u_{k+1}}$ is in $\mathbf{L}$ then there are $\mathbf{i, j}$ with $\mathbf{0 < i < j \leq k+1}$ and

$$\mathbf{u_0 u_1 \ldots u_{i-1} (u_i u_{i+1} \ldots u_{j-1})^* u_j u_{j+1} \ldots u_{k+1} \subseteq L.}$$

Exercise 2.52.

Given numbers $\mathbf{n, m}$ with $\mathbf{n > m > 2}$, provide an example of a regular language where the Block pumping constant is exactly $\mathbf{m}$ and where every nfa needs at least $\mathbf{n}$ states.

Find Small NFAs

Let $\mathbf{n}$ be the size of the alphabet $\mathbf{\Sigma}$ and assume $\mathbf{n \geq 2}$ (2.57: $\mathbf{n \geq 3}$). Determine size of the smallest dfa in dependence of $\mathbf{n}$ and construct good small dfa and nfa for the smallest allowed $\mathbf{n}$.

Exercise 2.53. $\mathbf{H = \{vawa : v, w \in \Sigma^*, a \in \Sigma\}}$.

Exercise 2.54. $\mathbf{I = \{ua : u \in (\Sigma - \{a\})^*, a \in \Sigma\}}$.

Exercise 2.55. $\mathbf{J = \{abuc : a, b \in \Sigma, u \in \Sigma^*, c \in \{a, b\}\}}$.

Exercise 2.56. $\mathbf{K = \{avbwc : a, b, c \in \Sigma, v, w \in \Sigma^*, c \notin \{a, b\}\}}$.

Exercise 2.57. $\mathbf{L = \{w : \exists\, a, b \in \Sigma\, [w \in \{a, b\}^*]\}}$.

Jaffe's Pumping Lemma

For these exercises, $\Sigma = \{0\}$.

Exercise 2.58

Show that an nfa for $\{0000000\}^* \cup \{00000000\}^*$ needs only 16 states while Jaffe's pumping lemma has constant 56.

Exercise 2.59

Generalise the idea of Exercise 2.58 to show that there is a family L_n of languages such that an nfa for L_n can be constructed with $O(n^3)$ states while Jaffe's pumping lemma needs a constant of at least 2^n. Provide the family of the L_n and explain why it satisfies the corresponding bounds.

Exercise 2.60

Determine the constant of Jaffe's pumping lemma and determine the sizes of the minimal nfa and dfa for $(\{00\} \cdot \{00000\}) \cup (\{00\}^* \cap \{000\}^*)$.