Theory of Computation 4
Non-Deterministic Finite Automata

Frank Stephan

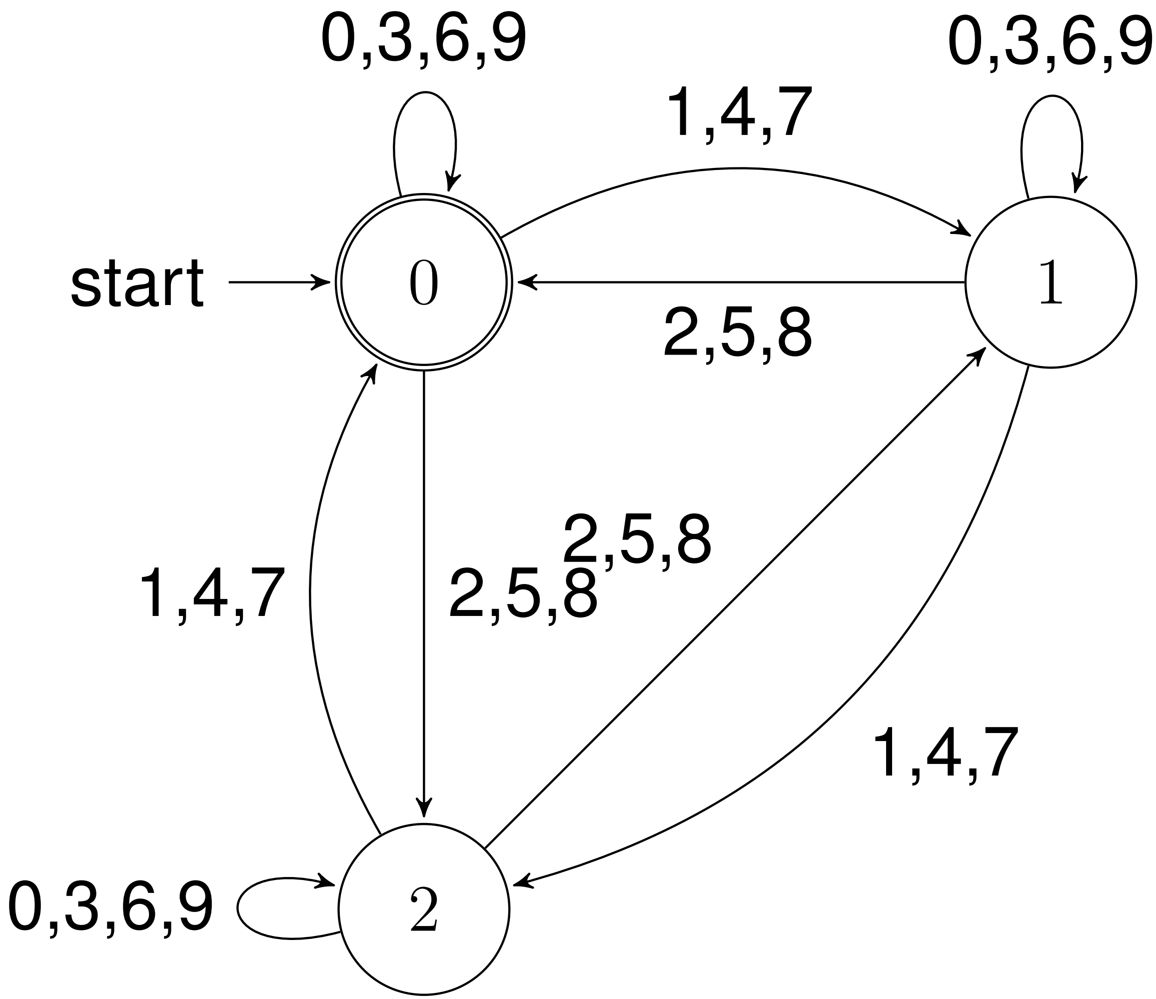
Department of Computer Science

Department of Mathematics

National University of Singapore

fstephan@comp.nus.edu.sg

Repetition 1 – DFA



Also representations as tables or computer programs.

Repetition 2

Theorem 3.9: Block Pumping Lemma
If $\mathbf{L}$ is a regular set then there is a constant $\mathbf{k}$ such that for all strings $\mathbf{u_0, u_1, \ldots, u_k}$ with $\mathbf{u_0 u_1 \ldots u_k} \in \mathbf{L}$ there are $\mathbf{i, j}$ with $\mathbf{0 < i < j \leq k}$ and

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

Theorem 3.11 [Ehrenfeucht, Parikh and Rozenberg 1981]
A language $\mathbf{L}$ is regular if and only if both $\mathbf{L}$ and the complement of $\mathbf{L}$ satisfy the Block Pumping Lemma.

Lemma 3.21 [Jaffe 1978]
A language $\mathbf{L} \subseteq \mathbf{\Sigma}^*$ is regular iff there is a constant $\mathbf{k}$ such that for all $\mathbf{x} \in \mathbf{\Sigma}^*$ and $\mathbf{y} \in \mathbf{\Sigma^k}$ there are $\mathbf{u, v, w}$ with $\mathbf{y = uvw}$ and $\mathbf{v} \neq \varepsilon$ such that, for all $\mathbf{h}$, $\mathbf{L_{xuv^hw} = L_{xy}}$;
that is, $\forall \mathbf{h} \in \mathbb{N}\, \forall \mathbf{z} \in \mathbf{\Sigma}^*\, [\mathbf{L(xuv^hwz) = L(xyz)}]$.

Repetition 3 – Derivatives

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

Theorem 3.17 [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 recognising $\mathbf{L}$.
The states are strings $\mathbf{x_1, x_2, \ldots, x_k}$ representing the derivatives $\mathbf{L_{x_1}, L_{x_2}, \ldots, L_{x_k}}$.
The transition rule $\delta(\mathbf{x_i, a})$ is the unique $\mathbf{x_j}$ with $\mathbf{L_{x_j} = L_{x_i a}}$.
The starting state is the unique state $\mathbf{x_i}$ with $\mathbf{L_{x_i} = L}$.
A state $\mathbf{x_i}$ is accepting iff $\varepsilon \in \mathbf{L_{x_i}}$ iff $\mathbf{x_i \in L}$.

Repetition 4 – Minimal DFA

Minimise dfa $(\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{s}, \mathbf{F})$
Construct Set R of Reacheable States: $\mathbf{R} = \{\mathbf{s}\}$;

While there are $\mathbf{q} \in \mathbf{R}$ and $\mathbf{a} \in \mathbf{\Sigma}$ with $\delta(\mathbf{q}, \mathbf{a}) \notin \mathbf{R}$ Do Begin
$\mathbf{R} = \mathbf{R} \cup \{\delta(\mathbf{q}, \mathbf{a})\}$ End.

Identify Distinguishable States γ:
Initialise $\gamma = \{(\mathbf{q}, \mathbf{p})$: exactly one of $\mathbf{p}, \mathbf{q}$ is accepting$\}$;
While $\exists(\mathbf{p}, \mathbf{q}) \in \mathbf{R} \times \mathbf{R} - \gamma, \mathbf{a} \in \mathbf{\Sigma}\, [(\delta(\mathbf{p}, \mathbf{a}), \delta(\mathbf{q}, \mathbf{a})) \in \gamma]$ Do
Begin $\gamma = \gamma \cup \{(\mathbf{p}, \mathbf{q}), (\mathbf{q}, \mathbf{p})\}$ End.

$\mathbf{Q}' = \{\mathbf{r} \in \mathbf{R} : \forall \mathbf{p} < \mathbf{r}\, [\gamma(\mathbf{p}, \mathbf{r}) \text{ or } \mathbf{r} \notin \mathbf{R}]\}$;
$\delta'(\mathbf{q}, \mathbf{a})$ is the unique $\mathbf{p} \in \mathbf{Q}'$ with $(\mathbf{p}, \delta(\mathbf{q}, \mathbf{a})) \notin \gamma$;
$\mathbf{s}'$ is the unique $\mathbf{s}' \in \mathbf{Q}'$ with $(\mathbf{s}, \mathbf{s}') \notin \gamma$;
$\mathbf{F}' = \mathbf{F} \cap \mathbf{Q}'$.

Motivation

Example 4.1
Let $\mathbf{n} = |\boldsymbol{\Sigma}|$ and $\mathbf{L} = \{\mathbf{w} : \exists \mathbf{a} \in \boldsymbol{\Sigma}\, [\mathbf{a}$ occurs in $\mathbf{w}$ at least twice$]\}$.

By the Theorem of Myhill and Nerode, a dfa for $\mathbf{L}$ needs $\mathbf{2^n + 1}$ states, as the language has $\mathbf{2^n + 1}$ derivatives:
If $\mathbf{x} \in \mathbf{L}$ then $\mathbf{L_x} = \boldsymbol{\Sigma}^*$;
if $\mathbf{x} \notin \mathbf{L}$ then $\varepsilon \notin \mathbf{L_x}$ and $\mathbf{L_x} \cap \boldsymbol{\Sigma} = \{\mathbf{a} : \mathbf{a}$ occurs in $\mathbf{x}\}$.

Dfa with states $\mathbf{A} \subseteq \boldsymbol{\Sigma}$ plus final state $\#$; Starting state $\emptyset$;
If $\mathbf{a} \in \mathbf{A}$ then $\delta(\mathbf{A}, \mathbf{a}) = \#$ else $\delta(\mathbf{A}, \mathbf{a}) = \mathbf{A} \cup \{\mathbf{a}\}$;
$\delta(\#, \mathbf{a}) = \#$ for all $\mathbf{a} \in \boldsymbol{\Sigma}$.

Can one do better with some other mechanism?

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 $(\mathbf{q}, \mathbf{a}, \mathbf{p}) \in \delta$.

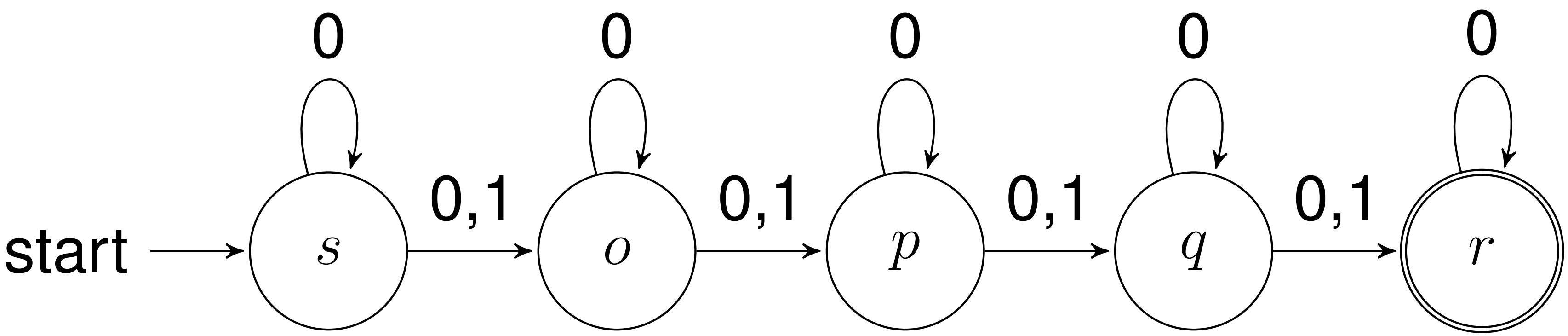
A run of an nfa on a word $\mathbf{a_1 a_2 \ldots a_n}$ is a sequence $\mathbf{q_0 q_1 q_2 \ldots q_n} \in \mathbf{Q}^*$ such that $\mathbf{q_0} = \mathbf{s}$ and $(\mathbf{q_m}, \mathbf{a_{m+1}}, \mathbf{q_{m+1}}) \in \delta$ for all $\mathbf{m} < \mathbf{n}$.

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.

Example 4.3

Language of all words with at least four letters and at most four ones.



Input $\mathbf{00111}$: Accepting runs $\mathbf{s}\,(\mathbf{0})\,\mathbf{s}\,(\mathbf{0})\,\mathbf{o}\,(\mathbf{1})\,\mathbf{p}\,(\mathbf{1})\,\mathbf{q}\,(\mathbf{1})\,\mathbf{r}$ and $\mathbf{s}\,(\mathbf{0})\,\mathbf{o}\,(\mathbf{0})\,\mathbf{o}\,(\mathbf{1})\,\mathbf{p}\,(\mathbf{1})\,\mathbf{q}\,(\mathbf{1})\,\mathbf{r}$; the rejecting run $\mathbf{s}\,(\mathbf{0})\,\mathbf{s}\,(\mathbf{0})\,\mathbf{s}\,(\mathbf{1})\,\mathbf{o}\,(\mathbf{1})\,\mathbf{p}\,(\mathbf{1})\,\mathbf{q}$ is not relevant.

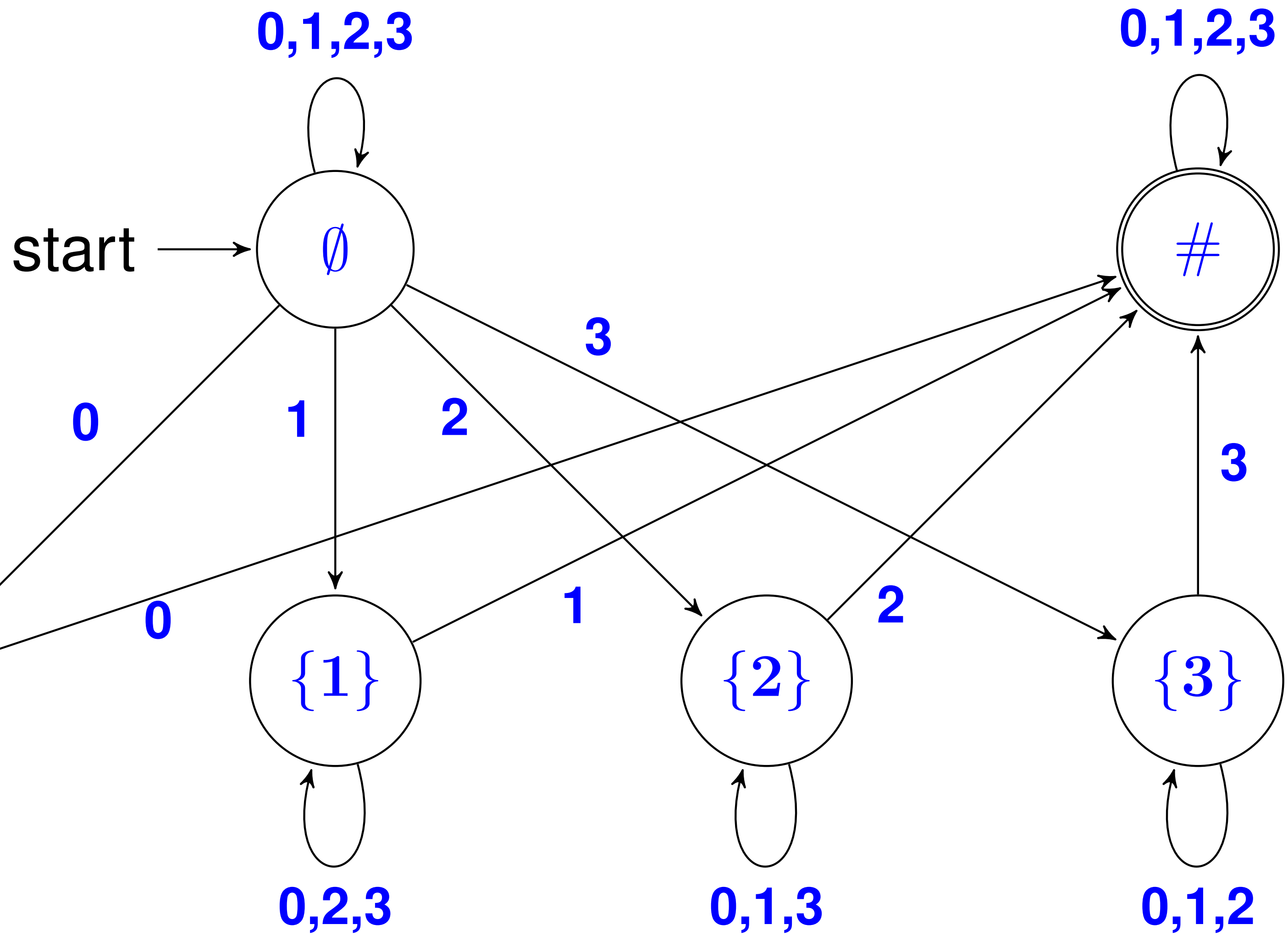
Input $\mathbf{11111}$: No accepting run; only possible run $\mathbf{s}\,(\mathbf{1})\,\mathbf{o}\,(\mathbf{1})\,\mathbf{p}\,(\mathbf{1})\,\mathbf{q}\,(\mathbf{1})\,\mathbf{r}\,(\mathbf{1})\ldots$ gets stuck.

Input $\mathbf{000}$: No run reaches accepting state $\mathbf{r}$ in time, $\mathbf{s}\,(\mathbf{0})\,\mathbf{o}\,(\mathbf{0})\,\mathbf{p}\,(\mathbf{0})\,\mathbf{q}$ is fastest run and falls short of final state.

Quiz: How many runs for $\mathbf{1001001}$ are accepting?

Exponential Improvement

The language from Example 4.1 has an nfa with $\mathbf{n+2}$ states while a dfa needs $\mathbf{2^n+1}$ states; here for $\mathbf{n=4}$.



Büchi's Powerset Construction

Given an nfa, one let for given state $\mathbf{q}$ and symbol $\mathbf{a}$ the set $\delta(\mathbf{q},\mathbf{a})$ denote all states $\mathbf{q}'$ to which the nfa can transit from $\mathbf{q}$ on symbol $\mathbf{a}$.

Theorem 4.5 [Büchi; Rabin and Scott]
For each nfa $(\mathbf{Q},\mathbf{\Sigma},\delta,\mathbf{s},\mathbf{F})$ with $\mathbf{n} = |\mathbf{Q}|$ states, there is an equivalent dfa $(\{\mathbf{Q}' : \mathbf{Q}' \subseteq \mathbf{Q}\},\mathbf{\Sigma},\delta',\{\mathbf{s}\},\mathbf{F}')$ with $\mathbf{2^n}$ states such that $\mathbf{F}' = \{\mathbf{Q}' \subseteq \mathbf{Q} : \mathbf{Q}' \cap \mathbf{F} \neq \emptyset\}$ and

$$\forall \mathbf{Q}' \subseteq \mathbf{Q}\, \forall \mathbf{a} \in \mathbf{\Sigma}\, [\delta'(\mathbf{Q}',\mathbf{a}) \;=\; \textstyle\bigcup_{\mathbf{q}' \in \mathbf{Q}} \delta(\mathbf{q}',\mathbf{a}) \;=\; \{\mathbf{q}'' \in \mathbf{Q} : \exists \mathbf{q}' \in \mathbf{Q}'\, [\mathbf{q}'' \in \delta(\mathbf{q}',\mathbf{a})]\}].$$

As the number of states is often overshooting, it is good to minimise the resulting automaton with the algorithm of Myhill and Nerode.

Verification

It is easy to see that δ' is indeed a deterministic transition function.

Let $\mathbf{w = a_1 a_2 \ldots a_m}$ be a word. Now let $\mathbf{Q_0 = \{s\}}$ and, for $\mathbf{k = 0, 1, \ldots, m-1}$, $\mathbf{Q_{k+1}} = \delta'(\mathbf{Q_k, a_{k+1}})$ be the run (sequence of states) of the dfa while processing $\mathbf{w}$.

If the dfa accepts $\mathbf{w}$ then there is $\mathbf{q_m \in Q_m \cap F}$ and one can select, for $\mathbf{k = m-1, n-2, \ldots, 1, 0}$, states $\mathbf{q_k \in Q_k}$ with $\mathbf{q_{k+1}} \in \delta(\mathbf{q_k, a_k})$. It follows that $\mathbf{q_0\, q_1\, \ldots\, q_m}$ is an accepting run for the nfa.

If the nfa accepts $\mathbf{w}$ with an accepting run $\mathbf{q_0\, q_1\, \ldots\, q_m}$ then $\mathbf{q_0 = s}$, $\mathbf{q_0 \in Q_0}$ and, for $\mathbf{k = 0, 1, \ldots, m-1}$, it follows from $\mathbf{q_k \in Q_k}$ that $\mathbf{q_{k+1}} \in \delta(\mathbf{q_k, a_{k+1}})$ and thus $\mathbf{q_{k+1} \in Q_{k+1}}$. Thus $\mathbf{q_m \in Q_m \cap F}$ and the run of the dfa is accepting as well.

Example 4.6

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 δ' of the dfa is given as

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

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

Exercises

Exercise 4.7

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;

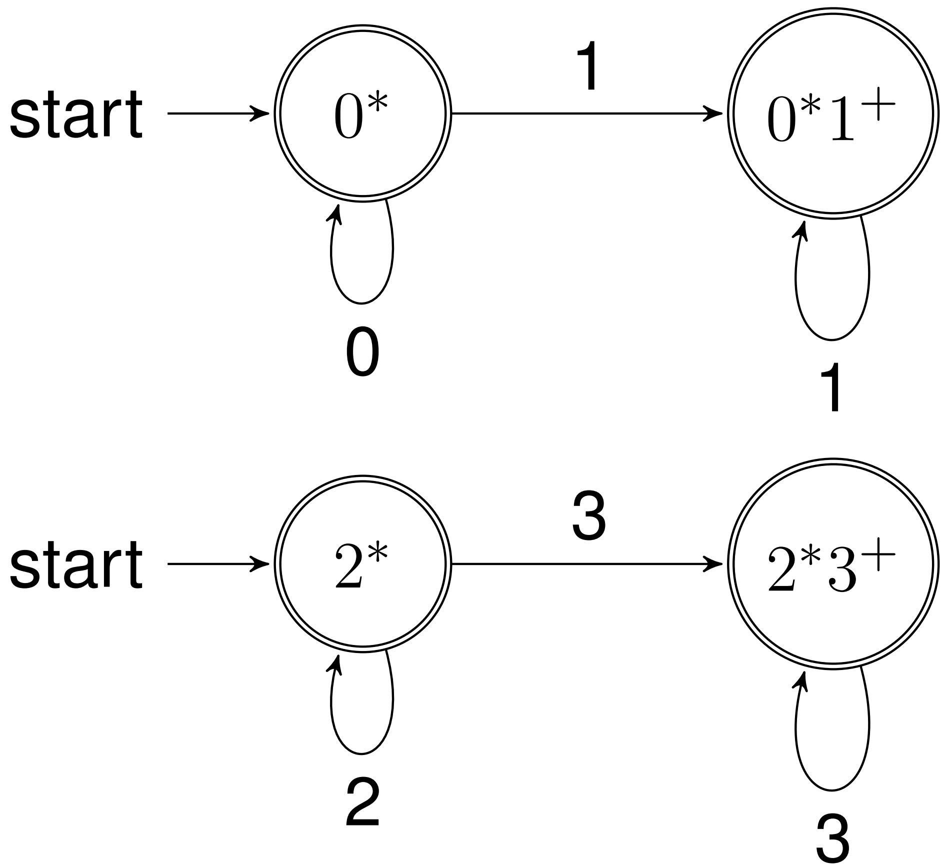
(c) Made a dfa recognising it with exactly 2^n states.

Exercise 4.8

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.

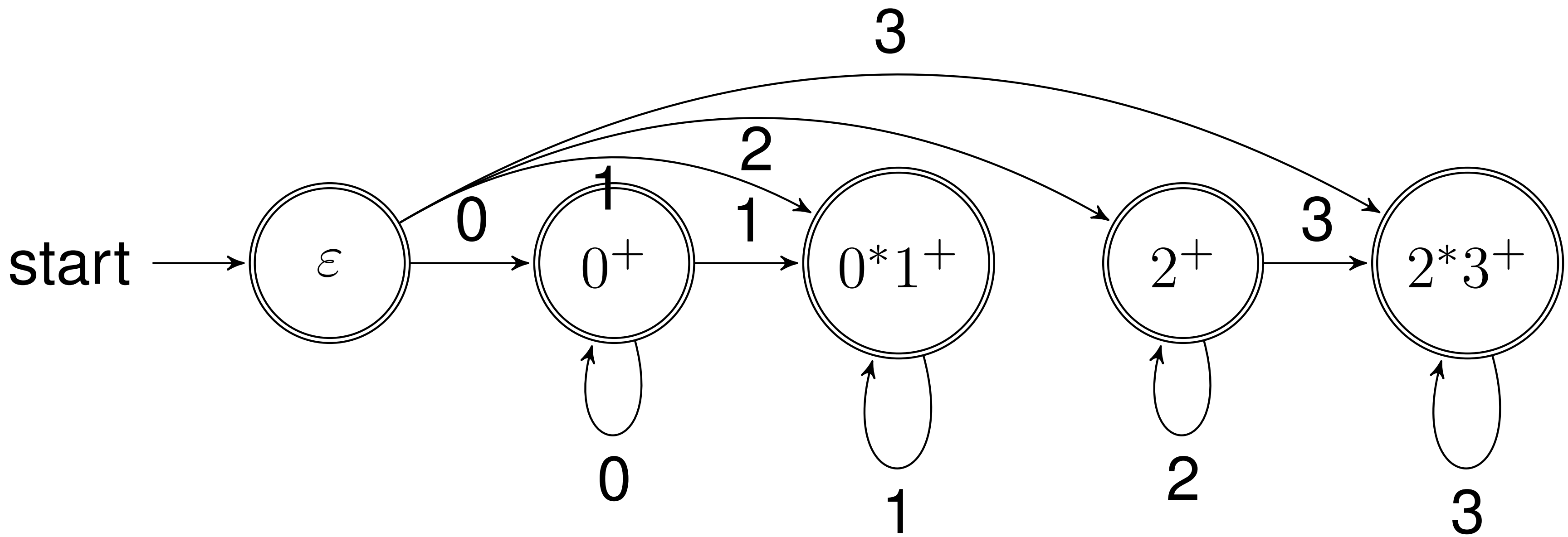
Set of Initial States

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 $\mathbf{F}$. Here an example for $\mathbf{0^*1^* \cup 2^*3^*}$.



Traditional NFA

One needs only to add one state to get a traditional nfa.



One new starting state is added and the transitions from old starting states to successor states are now done from the new starting state directly.

Matching Exponential Bounds

Exercise 4.10. Consider $\mathbf{L} = \{\mathbf{w} \in \mathbf{\Sigma}^* : \text{some } \mathbf{a} \in \mathbf{\Sigma} \text{ does not occur in } \mathbf{w}\}$.

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.

Exercise 4.11. Let $(\{\mathbf{q_0}, \mathbf{q_1}, \ldots, \mathbf{q_{n-1}}\}, \{\mathbf{0}, \mathbf{1}\}, \delta, \mathbf{q_0}, \{\mathbf{q_0}\})$ be an nfa with δ allowing on $\mathbf{1}$ to go from $\mathbf{q_m}$ to $\mathbf{q_{(m+1) \bmod n}}$ and on $\mathbf{0}$ to go from $\mathbf{q_m}$ with $\mathbf{m} > \mathbf{0}$ to either $\mathbf{q_0}$ or $\mathbf{q_m}$. One cannot go to any state from $\mathbf{q_0}$ on $\mathbf{0}$. Determine the number of states of an equivalent complete and minimal dfa. Explain how this number of states is found.

Exercise 4.12. Show that a dfa equivalent to an nfa with two states over alphabet $\{\mathbf{0}\}$ needs at most three states.

Regular Grammar to NFA

Theorem 4.13
Every language generated by a regular grammar is also recognised by an nfa.

Let $(\mathbf{N}, \mathbf{\Sigma}, \mathbf{P}, \mathbf{S})$ be a grammar generating $\mathbf{L}$.
Normalisations:

- Replace in $\mathbf{N}$ each rule $\mathbf{A} \to \mathbf{w}$ with $\mathbf{w} \in \mathbf{\Sigma}^+$ by $\mathbf{A} \to \mathbf{wB}$, $\mathbf{B} \to \varepsilon$ for new non-terminal $\mathbf{B}$;
- Replace in $\mathbf{N}$ each rule $\mathbf{A} \to \mathbf{a_1 a_2 \ldots a_n B}$ by new rules $\mathbf{A} \to \mathbf{a_1 C_1}$, $\mathbf{C_1} \to \mathbf{a_2 C_2}$, ..., $\mathbf{C_{n-1}} \to \mathbf{a_n B}$ for new non-terminals $\mathbf{C_1, C_2, \ldots, C_{n-1}}$.

Now make nfa $(\mathbf{N}, \mathbf{\Sigma}, \delta, \mathbf{S}, \mathbf{F})$ with $\delta(\mathbf{A}, \mathbf{a}) = \{\mathbf{B} : \mathbf{A} \Rightarrow^* \mathbf{aB}\}$ and $\mathbf{F} = \{\mathbf{C} \in \mathbf{N} : \mathbf{C} \Rightarrow^* \varepsilon\}$.

Example for Grammar to NFA

Example 4.14
$\mathbf{L = 0123^*}$.

Grammar $(\{\mathbf{S}, \mathbf{T}\}, \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}, \mathbf{P}, \mathbf{S})$ with rules
$\mathbf{P} = \{\mathbf{S} \to \mathbf{012}|\mathbf{012T}, \mathbf{T} \to \mathbf{3T}|\mathbf{3}\}$.

Updated to grammar with non-terminals
$\mathbf{N} = \{\mathbf{S}, \mathbf{S}', \mathbf{S}'', \mathbf{S}''', \mathbf{T}, \mathbf{T}'\}$ and rules $\mathbf{S} \to \mathbf{0S}'$, $\mathbf{S}' \to \mathbf{1S}''$, $\mathbf{S}'' \to \mathbf{2S}'''|\mathbf{2T}$, $\mathbf{S}''' \to \varepsilon$, $\mathbf{T} \to \mathbf{3T}|\mathbf{3T}'$, $\mathbf{T}' \to \varepsilon$.

NFA $(\mathbf{N}, \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}, \delta, \mathbf{S}, \{\mathbf{S}''', \mathbf{T}'\})$ with $\delta(\mathbf{S}, \mathbf{0}) = \{\mathbf{S}'\}$, $\delta(\mathbf{S}', \mathbf{1}) = \{\mathbf{S}''\}$, $\delta(\mathbf{S}'', \mathbf{2}) = \{\mathbf{S}''', \mathbf{T}\}$, $\delta(\mathbf{T}, \mathbf{3}) = \{\mathbf{T}, \mathbf{T}'\}$ and $\delta(\mathbf{A}, \mathbf{a}) = \emptyset$ in all other cases.

Accepting run for $\mathbf{0\,1\,2}$ is $\mathbf{S}\,(\mathbf{0})\,\mathbf{S}'\,(\mathbf{1})\,\mathbf{S}''\,(\mathbf{2})\,\mathbf{S}'''$ and for $\mathbf{0\,1\,2\,3\,3\,3}$ is $\mathbf{S}\,(\mathbf{0})\,\mathbf{S}'\,(\mathbf{1})\,\mathbf{S}''\,(\mathbf{2})\,\mathbf{T}\,(\mathbf{3})\,\mathbf{T}\,(\mathbf{3})\,\mathbf{T}\,(\mathbf{3})\,\mathbf{T}'$.

Exercises for Grammar to NFA

Exercise 4.15

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 4.16

Let $\mathbf{L}$ be generated by 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 the rules of the form $\mathbf{S} \to \mathbf{aaaaaS}$ for some digit $\mathbf{a}$ and the rule $\mathbf{S} \to \varepsilon$. What is the minimum number of states of a non-deterministic finite automaton recognising $\mathbf{L}$? What is the trade-off of the nfa compared to the minimal dfa for $\mathbf{L}$? Prove your answers.

Corollary 4.17: Regular

The following statements are all equivalent to "L is regular":

(a) L is generated by a regular expression;

(b) L is generated by a regular grammar;

(c) L is recognised by a deterministic finite automaton;

(d) L is recognised by a non-deterministic finite automaton;

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

(f) L satisfies Jaffe's Matching Pumping Lemma;

(g) L has only finitely many derivatives.

Size of Expressions

Example 4.18

The language

$$\mathbf{L} = \bigcup_{\mathbf{m}<\mathbf{n}} (\{\mathbf{0},\mathbf{1}\}^{\mathbf{m}} \cdot \{\mathbf{1}\} \cdot \{\mathbf{0},\mathbf{1}\}^* \cdot \{\mathbf{10}^{\mathbf{m}}\})$$

can be written down in $\mathbf{O}(\mathbf{n}^2)$ symbols as a regular expression but the corresponding dfa has at least $\mathbf{2}^{\mathbf{n}}$ states: if $\mathbf{x}$ has $\mathbf{n}$ digits then $\mathbf{10}^{\mathbf{m}} \in \mathbf{L}_{\mathbf{x}}$ iff the $\mathbf{m}$-th digit of $\mathbf{x}$ is $\mathbf{1}$.

Note that $\{\mathbf{0},\mathbf{1}\}^{\mathbf{2}}$ is written as $\{\mathbf{0},\mathbf{1}\} \cdot \{\mathbf{0},\mathbf{1}\}$ and $\{\mathbf{0},\mathbf{1}\}^{\mathbf{3}}$ is written as $\{\mathbf{0},\mathbf{1}\} \cdot \{\mathbf{0},\mathbf{1}\} \cdot \{\mathbf{0},\mathbf{1}\}$ in the regular expression and so on; this permits to keep the quadratic bound. The expression uses finite sets of strings, union, concatenation and star only.

Unary Alphabet

Theorem 4.19
Let $\mathbf{p_1, p_2, p_3, \ldots}$ be the prime numbers in ascending order. The language $\mathbf{L_n} = \{\mathbf{0^{p_1}}\}^+ \cap \{\mathbf{0^{p_2}}\}^+ \cap \ldots \cap \{\mathbf{0^{p_n}}\}^+$ 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.

The expression - when written $\mathbf{000}$ in place of $\mathbf{0^3}$ and so on – has length $\mathbf{O(n^2 \log(n))}$ and shortest word has length $\mathbf{p_1 \cdot p_2 \cdot \ldots \cdot p_n \geq 2^n}$. Shortest word recognised by nfa cannot be longer as the number of states, as in the accepting run, no state is repeated. Thus nfa has at least $\mathbf{2^n}$ states.

Length of Shortest Word

Proposition
If a regular expression σ uses only lists of members, union, concatenation and Kleene star, then the shortest word $\mathbf{sw}(\sigma)$ satisfies $|\mathbf{sw}(\sigma)| \leq |\sigma|$.

Proof by structural induction.
If σ is a list of a finite set then every word in the list is shorter than $|\sigma|$.
If σ, τ satisfy $|\mathbf{sw}(\sigma)| \leq |\sigma|$ and $|\mathbf{sw}(\tau)| \leq |\tau|$ then also $|\mathbf{sw}(\sigma \cup \tau)| \leq |\sigma \cup \tau|$ and $|\mathbf{sw}(\sigma \cdot \tau)| \leq |\sigma \cdot \tau|$ and $|\mathbf{sw}(\sigma^*)| = \mathbf{0}$ (as the empty word ε is always in the Kleene star of an expression).

Thus if one writes the Expression from Theorem 4.19 without intersections then its length is at least $\mathbf{2^n}$.

Example of Inductive Definition

Recall the length-lexicographic ordering, for $\Sigma = \{0, 1\}$; it is $\varepsilon <_{ll} 0 <_{ll} 1 <_{ll} 00 <_{ll} 01 <_{ll} 10 <_{ll} 11 <_{ll} 000 <_{ll} \ldots$; one uses $<_{ll}$ to define $\mathbf{sw}(\mathbf{reg\,exp})$:

$$
\begin{aligned}
\mathbf{sw}(\emptyset) &= \infty; \\
\mathbf{sw}(\{\mathbf{w_1}, \ldots, \mathbf{w_n}\}) &= \mathbf{min_{ll}}\{\mathbf{w_1}, \ldots, \mathbf{w_n}\}; \\
\mathbf{sw}(\sigma \cup \tau) &= \begin{cases} \mathbf{sw}(\sigma) & \text{if } \mathbf{sw}(\tau) = \infty; \\ \mathbf{sw}(\tau) & \text{if } \mathbf{sw}(\sigma) = \infty; \\ \mathbf{min_{ll}}\{\mathbf{sw}(\sigma), \mathbf{sw}(\tau)\} & \text{otherwise}; \end{cases} \\
\mathbf{sw}(\sigma \cdot \tau) &= \begin{cases} \infty & \text{if } \mathbf{sw}(\sigma) = \infty \\ & \text{or } \mathbf{sw}(\tau) = \infty; \\ \mathbf{sw}(\sigma) \cdot \mathbf{sw}(\tau) & \text{otherwise}; \end{cases} \\
\mathbf{sw}(\sigma^*) &= \varepsilon.
\end{aligned}
$$

One can see by structural induction: $|\mathbf{sw}(\sigma)| \leq |\sigma|$ where ∞ denotes that there is no word in the expression and $\infty, \{, \}, (,), \cup, \cdot, ^*, \emptyset$ are symbols of length $\mathbf{1}$ and $|\varepsilon| = \mathbf{0}$.

Length of Short Words

Exercise 4.21
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 4.22
Is Exercise 4.21 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 $2|\sigma|$? Again prove that bound or provide a further counter example.

Exponential Gap

Theorem 4.23 [Ehrenfeucht and Zeiger 1976]
Let $\Sigma = \{(a, b) : a, b \in \{1, 2, \ldots, n\}\}$ and $L = \{(1, a_1)\,(a_1, a_2)\ \ldots\ (a_{m-1}, a_m) : a_1, \ldots, a_m \in \{1, \ldots, n\}, m \geq 1\}$. Now L can be recognised by a dfa with $n + 1$ states but there is no regular expression for L using lists of finite sets, union, concatenation and Kleene star which is shorter than 2^{n-1}.

Remark
One can make a short expression using intersection as well:

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

Pumping Constants and NFA

Exercise 4.24

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 4.25

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.

Exercises 4.26 - 4.30

Let $\mathbf{n}$ be the size of the alphabet $\mathbf{\Sigma}$ and assume $\mathbf{n \geq 2}$. Determine the size of the smallest nfa and dfa for the following languages in dependence of $\mathbf{n}$. Explain the results and construct the automata for $\mathbf{\Sigma = \{0, 1\}}$ (4.30: $\mathbf{\{0, 1, 2\}}$).

Exercise 4.26

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

Exercise 4.27

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

Exercise 4.28

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

Exercise 4.29

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

Exercise 4.30

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

Exercises 4.31, 4.32 and 4.33

Exercise 4.31

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

Exercise 4.32

Generalise the idea of Exercise 4.31 to show that there is a family $\mathbf{L_n}$ of languages such that an nfa for $\mathbf{L_n}$ can be constructed with $\mathbf{O(n^3)}$ states while Jaffe's pumping lemma needs a constant of at least $\mathbf{2^n}$. Provide the family of the $\mathbf{L_n}$ and explain why it satisfies the corresponding bounds.

Exercise 4.33

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