Theory of Computation 3
Deterministic Finite Automata

Frank Stephan

Department of Computer Science

Department of Mathematics

National University of Singapore

fstephan@comp.nus.edu.sg

Repetition 1

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 2

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

CH0: No restriction, all recursively enumerable languages.

CH1 (context-sensitive): Every rule is of the form $\mathbf{uAw} \rightarrow \mathbf{uvw}$ with $\mathbf{A} \in \mathbf{N}$, $\mathbf{u}, \mathbf{v}, \mathbf{w} \in (\mathbf{N} \cup \mathbf{\Sigma})^*$ and $\mathbf{v} = \varepsilon$ is only possible if $\mathbf{A} = \mathbf{S}$ and $\mathbf{S}$ never on right side of a rule.

Easier formalisation: If $\mathbf{l} \rightarrow \mathbf{r}$ is a rule then $|\mathbf{l}| \leq |\mathbf{r}|$, that is, $\mathbf{r}$ is at least as long as $\mathbf{l}$. Special rule (as above) for the case that $\varepsilon \in \mathbf{L}$.

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

CH3 (regular): Every rule is of the form $\mathbf{A} \rightarrow \mathbf{wB}$ or $\mathbf{A} \rightarrow \mathbf{w}$ 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

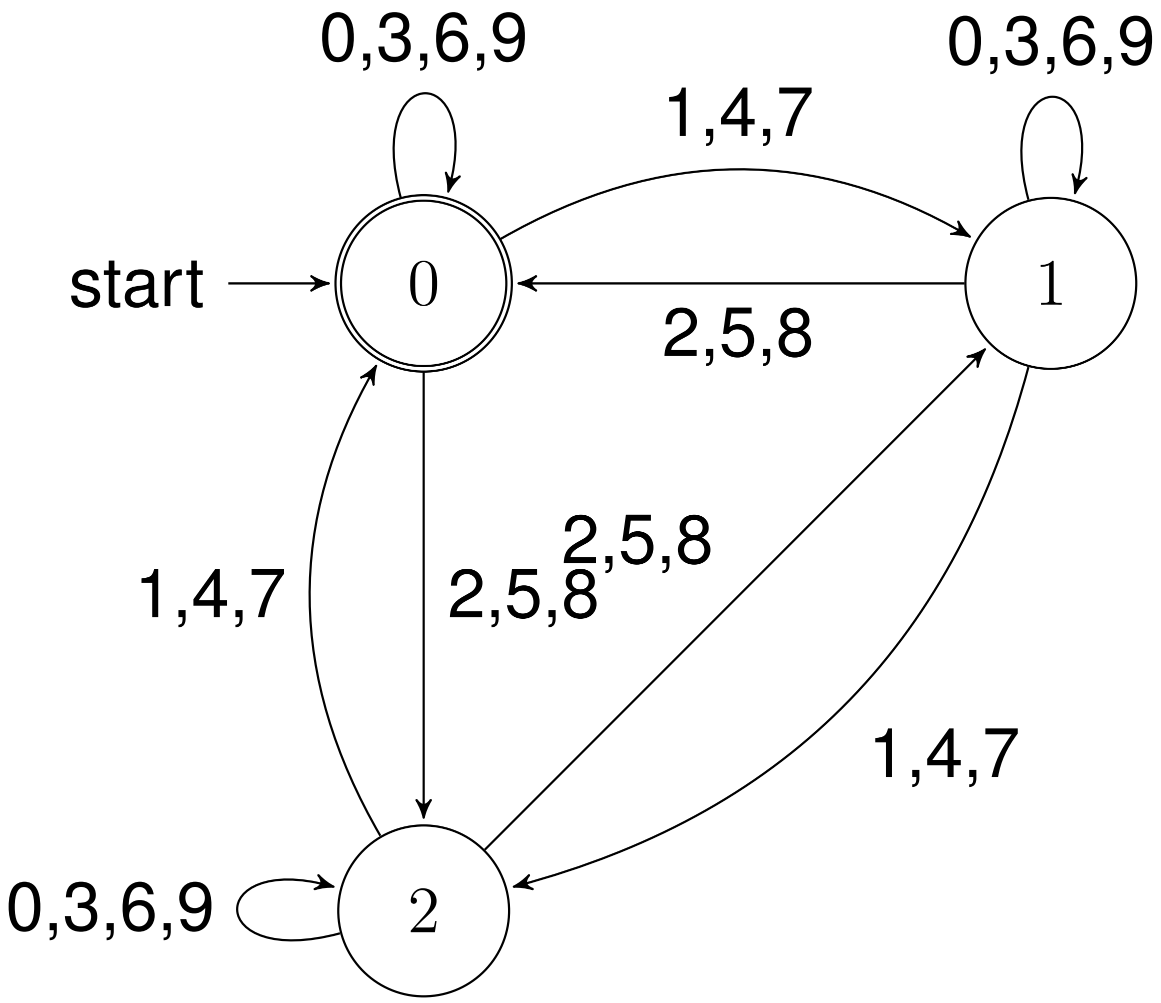
Check whether decimal number $a_1a_2\ldots a_n$ is a multiple of 3.

Easy Algorithm

```
Scan through the word from $a_1$ to $a_n$.
Maintain memory $s$.
Initialise $s = 0$.
For $m = 1, 2, \ldots, n$ Do
    Begin Let $s = s + a_m$ modulo $3$ End.
If $s = 0$
    Then $a_1a_2\ldots a_n$ is multiple of $3$
    Else $a_1a_2\ldots a_n$ is not a multiple of $3$.
```

Test the algorithm on 1, 20, 304, 2913, 49121, 391213, 2342342, 123454321, 1234554321.

Finite Automaton



Automata Working Mod 7

Automaton $(\{\mathbf{0},\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{4},\mathbf{5},\mathbf{6}\},\{\mathbf{0},\mathbf{1},\ldots,\mathbf{9}\},\delta,\mathbf{0},\{\mathbf{0}\})$ 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 $\mathbf{10}*\mathbf{q}+\mathbf{a}$ 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 $\mathbf{Q} \times \mathbf{\Sigma}$ to $\mathbf{Q}$, the starting state $\mathbf{s} \in \mathbf{Q}$ and a set $\mathbf{F} \subseteq \mathbf{Q}$ 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 $\mathbf{q_0} = \mathbf{s}$ and $\delta(\mathbf{q_m}, \mathbf{a_{m+1}}) = \mathbf{q_{m+1}}$ for all $\mathbf{m} < \mathbf{n}$. 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 $\mathbf{Q} \times \mathbf{\Sigma}^*$ to $\mathbf{Q}$ by letting $\delta(\mathbf{q}, \varepsilon) = \mathbf{q}$ and $\delta(\mathbf{q}, \mathbf{wa}) = \delta(\delta(\mathbf{q}, \mathbf{w}), \mathbf{a})$. So the dfa accepts $\mathbf{w}$ iff $\delta(\mathbf{s}, \mathbf{w}) \in \mathbf{F}$.

Exercise 3.6

Make a finite automaton for the program from the Slide 7.

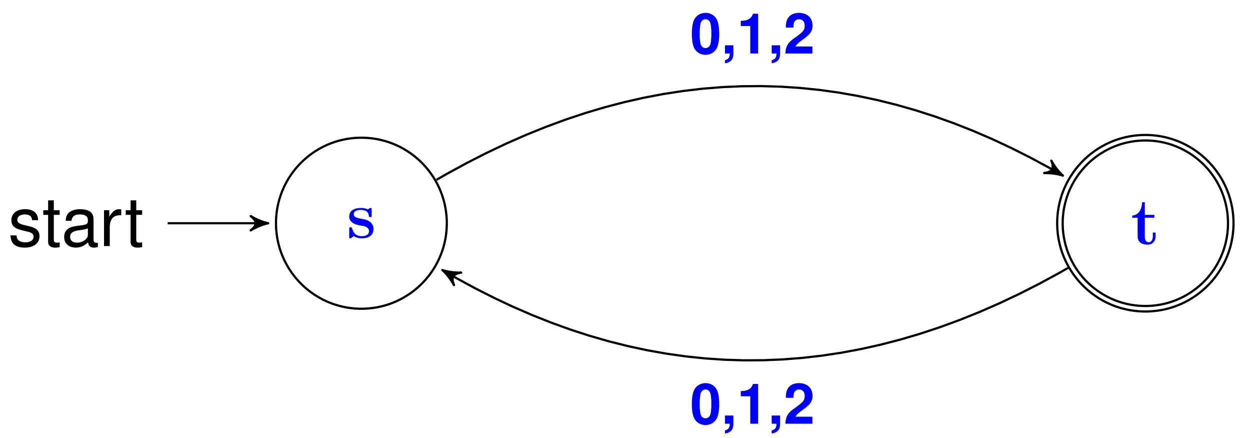
Use $\mathbf{Q} = \{\mathbf{s}, \mathbf{z}, \mathbf{r}, \mathbf{q_0}, \mathbf{q_1}, \ldots, \mathbf{q_{256}}\}$.

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 3.7

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



Regular Sets

Theorem 3.8
The following statements are equivalent for a language L.

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

(b) L is generated by a regular expression;

(c) L is generated by a regular grammar.

Equivalence of (b) and (c) was in Lecture 2. Now implication (a) to (c) is shown; the missing implication comes in Lecture 4.

Implication (a) to (c)

Assume $(\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{s}, \mathbf{F})$ is a dfa recognising $\mathbf{L}$.

Consider grammar $(\mathbf{Q}, \mathbf{\Sigma}, \mathbf{P}, \mathbf{s})$ with $\mathbf{P}$ having the following rules:

- $\mathbf{q} \rightarrow \mathbf{ar}$ whenever $\delta(\mathbf{q}, \mathbf{a}) = \mathbf{r}$;
- $\mathbf{q} \rightarrow \varepsilon$ whenever $\mathbf{q} \in \mathbf{F}$.

Let $\mathbf{w} = \mathbf{a_1 a_2 \ldots a_n}$ be a word.

The dfa recognises $\mathbf{w}$ iff there is an accepting run starting in $\mathbf{q_0} = \mathbf{s}$ and transiting from $\mathbf{q_{m-1}}$ to $\mathbf{q_m}$ on symbol $\mathbf{a_m}$ with $\mathbf{q_n} \in \mathbf{F}$ iff there is a derivation of $\mathbf{w}$ of the form $\mathbf{q_0 \Rightarrow a_1 q_1 \Rightarrow a_1 a_2 q_2 \Rightarrow \ldots \Rightarrow a_1 a_2 \ldots a_n q_n \Rightarrow a_1 a_2 \ldots a_n}$ with $\mathbf{q_0} = \mathbf{s}$ for the given grammar iff the grammar generates $\mathbf{w}$.

Example

Language: Multiples of 3 (with leading zeroes).

Grammar
Set of Terminals: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
Set of Non-Terminals: $\{q_0, q_1, q_2\}$.
Rules:

$q_0 \to 0q_0|1q_1|2q_2|3q_0|4q_1|5q_2|6q_0|7q_1|8q_2|9q_0|\varepsilon$;
$q_1 \to 0q_1|1q_2|2q_0|3q_1|4q_2|5q_0|6q_1|7q_2|8q_0|9q_1$;
$q_2 \to 0q_2|1q_0|2q_1|3q_2|4q_0|5q_1|6q_2|7q_0|8q_1|9q_2$.

Start Symbol: q_0.

Sample Derivations

$q_0 \Rightarrow 2q_2 \Rightarrow 22q_1 \Rightarrow 222q_0 \Rightarrow 222$;
$q_0 \Rightarrow 2q_2 \Rightarrow 24q_0 \Rightarrow 243q_0 \Rightarrow 243$;
$q_0 \Rightarrow 7q_1 \Rightarrow 72q_0 \Rightarrow 729q_0 \Rightarrow 729$;
$q_0 \Rightarrow 2q_2 \Rightarrow 25q_1 \Rightarrow 256q_1 \not\Rightarrow 256$.

Block Pumping Lemma

Theorem 3.9 [Ehrenfeucht, Parikh and Rozenberg 1981]
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 $\mathbf{u_0 u_1 \ldots u_k \in L}$ there are $\mathbf{i, j}$ with $\mathbf{0 < i < j \leq k}$ and

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

So if one splits a word in $\mathbf{L}$ into $\mathbf{k+1}$ parts then one can select some neighbouring parts in the middle of the word which can be pumped.

If one $\mathbf{u_i}$ with $\mathbf{0 < i < k}$ is empty then $\mathbf{u_i}$ can be pumped; one can also require that $\mathbf{u_1, u_2, \ldots, u_{k-1}}$ are nonempty.

Example 3.10

$L = \{1,2\}^* \cdot (0 \cdot \{1,2\}^* \cdot 0 \cdot \{1,2\}^*)^*$ satisfies the Block Pumping Lemma with $k = 3$:
Let u_0, u_1, u_2, u_3 be given with $u_0u_1u_2u_3 \in L$.
If u_1 contains an even number of 0 then $u_0(u_1)^*u_2u_3 \subseteq L$;
If u_2 contains an even number of 0 then $u_0u_1(u_2)^*u_3 \subseteq L$;
If u_1, u_2 both contain an odd number of 0 then $u_0(u_1u_2)^*u_3 \subseteq L$.

$H = \{u : u$ has a different number of 0s than 1s$\}$ does not satisfy the Block Pumping Lemma with any k:
If $u = 0^k1^{k+k!}$ then one takes $u_0, u_1, \ldots, u_{k-1} = 0$ and $u_k = 1^{k+k!}$ and whatever pumping interval one choses, the pump is of the form 0^h for $h < k$ and $0^k \cdot (0^h)^{k!/h}1^{k+k!}$ is not in H.

Block Pumping

Theorem 3.11 [Ehrenfeucht, Parikh and Rozenberg 1981] If a language and its complement both satisfy the Block Pumping Lemma then the language is regular.

Quiz 3.12 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 $0\}$?

Blockpumping Constants

The optimal constant for a language $\mathbf{L}$ is the least $\mathbf{n}$ such that for all words $\mathbf{u_0u_1u_2 \ldots u_n \in L}$ there are $\mathbf{i, j}$ with $\mathbf{0 < i < j \leq n}$ and $\mathbf{u_0 \ldots u_{i-1}(u_i \ldots u_{j-1})^* u_j \ldots u_n \subseteq L}$.

Exercise 3.13 Find the optimal block pumping constants for the following languages:
(a) $\{\mathbf{w \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^*}$: at least one nonzero digit $\mathbf{a}$ occurs in $\mathbf{w}$ at least three times$\}$;
(b) $\{\mathbf{w \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^* : |w| = 255}\}$;
(c) $\{\mathbf{w \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^*}$: the length $|\mathbf{w}|$ is not a multiple of $\mathbf{6}\}$.

Exercise 3.14 Find the optimal block pumping constants for the following languages:
(a) $\{\mathbf{w \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^* : w}$ is a multiple of $\mathbf{25}\}$;
(b) $\{\mathbf{w \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^* : w}$ is not a multiple of $\mathbf{3}\}$;
(c) $\{\mathbf{w \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^* : w}$ is a multiple of $\mathbf{400}\}$.

Exercises 3.15 and 3.16

Exercise 3.15

Find a regular language $\mathbf{L}$ so that the constant of the Block Pumping Lemma for $\mathbf{L}$ is $\mathbf{4}$ and for the complement of $\mathbf{L}$ is $\mathbf{4096}$ or more. Note that you can use an alphabet $\mathbf{\Sigma}$ of sufficiently large size.

Exercise 3.16

Give an example of a language $\mathbf{L}$ which satisfies the normal Pumping Lemma (where there is a $\mathbf{k}$ such that for all $\mathbf{w}$ with $|\mathbf{w}| \geq \mathbf{k}$ there are $\mathbf{x}, \mathbf{y}, \mathbf{z}$ with $\mathbf{xyz} = \mathbf{w}$, $|\mathbf{y}| > \mathbf{0}$, $|\mathbf{xy}| \leq \mathbf{k}$ and $\mathbf{xy^*z} \subseteq \mathbf{L}$) but not the Block Pumping Lemma.

Derivatives

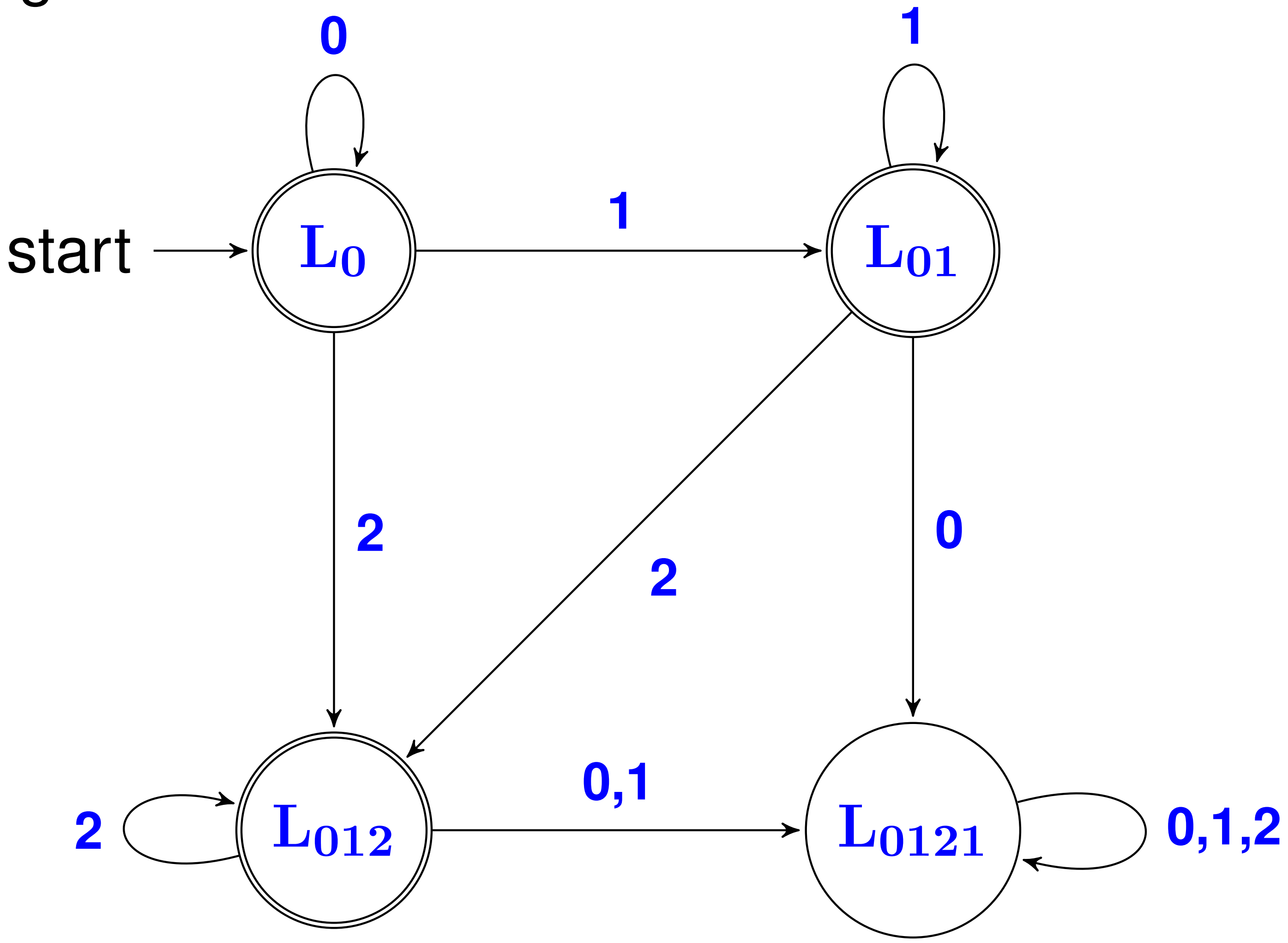
Given a language $\mathbf{L}$, let $\mathbf{L_x = \{y : x \cdot y \in L\}}$ be the derivative of $\mathbf{L}$ at $\mathbf{x}$.

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 $\mathbf{x_i \in L}$.

Example 3.19

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



Other Direction

Assume that a dfa recognises a language $\mathbf{L}$ and that δ is the transition function of the dfa. Now if $\delta(\mathbf{s},\mathbf{v}) = \delta(\mathbf{s},\mathbf{w})$ then $\mathbf{L_v} = \mathbf{L_w}$: Given a word $\mathbf{u}$, then $\mathbf{u} \in \mathbf{L_v}$ iff $\mathbf{vu} \in \mathbf{L}$ iff $\delta(\delta(\mathbf{s},\mathbf{v}),\mathbf{u})$ is accepting iff $\delta(\delta(\mathbf{s},\mathbf{w}),\mathbf{u})$ is accepting iff $\mathbf{wu} \in \mathbf{L}$ iff $\mathbf{u} \in \mathbf{L_w}$.

So one can pick for every reachable state $\mathbf{q}$ a word $\mathbf{x_q}$ with $\delta(\mathbf{s},\mathbf{x_q}) = \mathbf{q}$ and it follows that for every word $\mathbf{y}$ there is a reachable state $\mathbf{q}$ with $\delta(\mathbf{s},\mathbf{y}) = \mathbf{q}$ and thus $\mathbf{L_y} = \mathbf{L_{x_q}}$.

In summary, every derivative $\mathbf{L_y}$ is equal to one of the derivatives $\mathbf{L_{x_q}}$ with $\mathbf{q}$ being a reachable state and therefore there are only finitely many derivatives in a regular language.

Example 3.20

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

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

Jaffe's Pumping Lemma

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

Proof
If a dfa recognises with $\mathbf{k}$ states recognises $\mathbf{L}$ then there are for every $\mathbf{x}, \mathbf{y}$ with $|\mathbf{y}| = \mathbf{k}$ two distinct prefixes $\mathbf{u}, \mathbf{uv}$ of $\mathbf{y}$ such that the dfa is in the same state after reading $\mathbf{xu}$ and $\mathbf{xuv}$. Thus when splitting $\mathbf{y}$ into $\mathbf{u} \cdot \mathbf{v} \cdot \mathbf{w}$ for $\mathbf{u}, \mathbf{v}$ from above then, for all $\mathbf{z}$, the automaton is for all $\mathbf{h}$ on the words $\mathbf{xuv^hwz}$ in the same state; hence $\mathbf{L}_{\mathbf{xuv^hw}} = \mathbf{L}_{\mathbf{xy}}$ for all $\mathbf{h}$.

Conversely, for every $\mathbf{z}$ of length at least $\mathbf{k}$ there is a $\mathbf{z}'$ shorter than $\mathbf{z}$ with $\mathbf{L}_{\mathbf{z}'} = \mathbf{L}_{\mathbf{z}}$; thus there are at most as many derivatives as there are words up to length $\mathbf{k} - \mathbf{1}$ and thus $\mathbf{L}$ is regular by the Theorem of Myhill and Nerode.

Exercises

Exercise 3.22

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 3.23

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.

Algorithm 3.29

Minimise dfa $(\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{s}, \mathbf{F})$

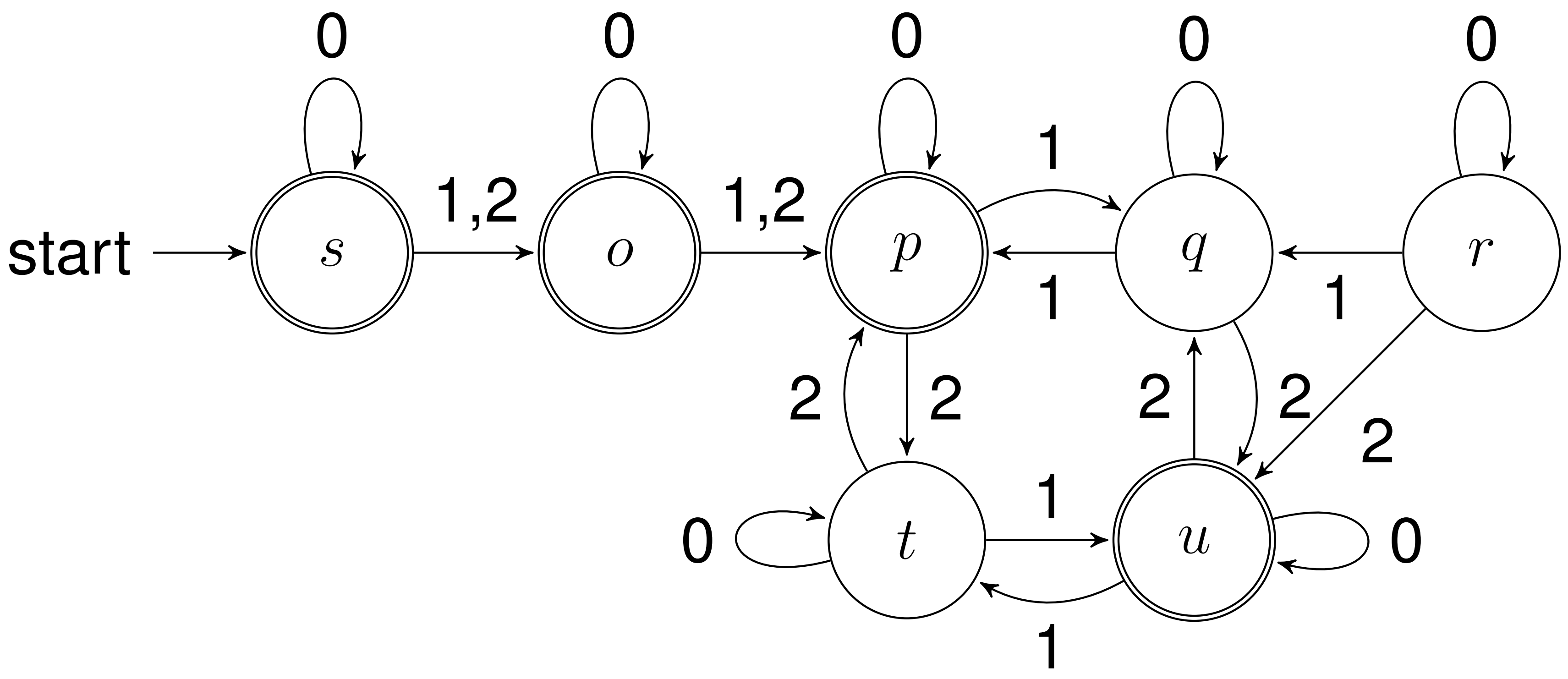
Construct Set R of Reacheable States: $\mathbf{R} = \{\mathbf{s}\}$;
While $\exists \mathbf{q} \in \mathbf{R}\, \exists \mathbf{a} \in \mathbf{\Sigma}\, [\delta(\mathbf{q}, \mathbf{a}) \notin \mathbf{R}]$
Do Begin $\mathbf{R} = \mathbf{R} \cup \{\delta(\mathbf{q}, \mathbf{a})\}$ End.

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

Minimal Automaton $(\mathbf{Q}', \mathbf{\Sigma}, \delta', \mathbf{s}', \mathbf{F}')$:
$\mathbf{Q}' = \{\mathbf{q} \in \mathbf{R} : \forall \mathbf{p} < \mathbf{q}\, [(\mathbf{p}, \mathbf{q}) \in \gamma \text{ or } \mathbf{p} \notin \mathbf{R}]\}$;
$\delta'(\mathbf{q}, \mathbf{a})$ 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 $(\mathbf{s}, \mathbf{s}') \notin \gamma$;
$\mathbf{F}' = \mathbf{F} \cap \mathbf{Q}'$.

Exercise 3.30

Make an equivalent minimal complete dfa for this one:



Follow the steps of the algorithm of Myhill and Nerode.

Exercise 3.31

Assume that $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$ and $Q = \{(a,b,c) : a,b,c \in \Sigma\}$ is the set of states. Furthermore assume that $\delta((a,b,c),d) = (b,c,d)$ for all $a,b,c,d \in \Sigma$, $(0,0,0)$ is the start state and that $F = \{(1,1,0),(3,1,0),(5,1,0),(7,1,0),(9,1,0)\}$ is the set of accepting states.

This dfa has 1000 states. Find a smaller dfa for this set and try to get the dfa as small as possible.

Exercise 3.32

Assume that $\mathbf{\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}}$ and $\mathbf{Q = \{(a, b, c) : a, b, c \in \Sigma\}}$ is the set of states. Furthermore assume that $\delta((\mathbf{a, b, c}), \mathbf{d}) = (\mathbf{b, c, d})$ for all $\mathbf{a, b, c, d \in \Sigma}$, $(\mathbf{0, 0, 0})$ is the start state and that $\mathbf{F = \{(1, 2, 5), (3, 7, 5), (6, 2, 5), (8, 7, 5)\}}$ is the set of accepting states.

This dfa has $\mathbf{1000}$ states. Find a smaller dfa for this set and try to get the dfa as small as possible.

Exercises 3.33 to 3.36

These two exercises ask to provide a minimal dfa for a language $\mathbf{L}$; though $\mathbf{L}$ is given by a context-free grammar, it is in both cases regular. The dfas need not be complete.

Exercise 3.33 – The grammar is given as

$(\{\mathbf{S},\mathbf{T},\mathbf{U}\},\{\mathbf{0},\mathbf{1},\mathbf{2},\mathbf{3}\},\mathbf{P},\mathbf{S})$ with $\mathbf{P} =$ $\{\mathbf{S} \rightarrow \mathbf{TTT}|\mathbf{TTU}|\mathbf{TUU}|\mathbf{UUU}, \mathbf{T} \rightarrow \mathbf{0T}|\mathbf{T1}|\mathbf{01},$ $\mathbf{U} \rightarrow \mathbf{2U}|\mathbf{U3}|\mathbf{23}\}.$

Exercise 3.34 – The grammar is given as

$(\{\mathbf{S},\mathbf{T},\mathbf{U}\},\{\mathbf{0},\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{4},\mathbf{5}\},\mathbf{P},\mathbf{S})$ with $\mathbf{P} =$ $\{\mathbf{S} \rightarrow \mathbf{TS}|\mathbf{SU}|\mathbf{T23U}, \mathbf{T} \rightarrow \mathbf{0T}|\mathbf{T1}|\mathbf{01},$ $\mathbf{U} \rightarrow \mathbf{4U}|\mathbf{U5}|\mathbf{45}\}.$

Exercises 3.35 and 3.36 – Provide regular expressions for the first and the second of the above grammars, respectively.

Additional Exercises

Provide finite automata for the below sets of numbers; the dfas can be made in any of the styles of slides 5 to 7.

Exercise 3.37. All decimal numbers where between between two occurences of a digit $\mathbf{d}$ are at least three other digits.

Exercise 3.38. All decimal numbers which are not multiples of a one-digit prime number.

Exercise 3.39. All decimal numbers with at least five decimal digits which are divisible by $\mathbf{8}$.

Exercise 3.40. All decimal numbers which have in their decimal representation twenty consecutive odd digits.

Exercise 3.41. All octal numbers (digits $\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{6}, \mathbf{7}$) without leading zeroes which are not multiples of $\mathbf{7}$.

Automata to Regular Expressions

Consider the automaton $(\{0,1,2,3\},\{0,1,2,3\},\delta,0,\{1,3\})$ with δ given in this table.

q	type	$\delta(q,a)$ for $a=0$	1	2	3
0	start, rej	0	1	2	3
1	acc	1	1	2	3
2	rej	2	2	2	3
3	acc	3	3	3	3

Exercise 3.42. Make a regular expression for the language L recognised by the dfa.

Exercise 3.43. Let L as in Exercise 3.42 and make a regular expression for the language of words of odd lengths in L.

Exercise 3.44. Let L as in Exercise 3.42 and make a regular expression for the language of words of length at least 5 in L.