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 \( L \) is a regular set then there is a constant \( k \) such that for all strings \( u_0, u_1, \ldots, u_k \) with \( u_0u_1 \ldots u_k \in L \) there are \( i, j \) with \( 0 < i < j \leq k \) and

\[
(u_0u_1 \ldots u_{i-1}) \cdot (u_iu_{i+1} \ldots u_{j-1})^* \cdot (u_ju_{j+1} \ldots u_k) \subseteq L.
\]

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

Lemma 3.21 [Jaffe 1978]
A language \( L \subseteq \Sigma^* \) is regular iff there is a constant \( k \) such that for all \( x \in \Sigma^* \) and \( y \in \Sigma^k \) there are \( u, v, w \) with \( y = uvw \) and \( v \neq \varepsilon \) such that, for all \( h \), \( L_{xuv^hw} = L_{xy} \); that is, \( \forall h \in \mathbb{N} \forall z \in \Sigma^* \ [L(xuv^hwz) = L(xyz)] \).
Repetition 3 – Derivatives

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

**Theorem 3.17** [Myhill and Nerode].
A language \( L \) is regular iff \( L \) has only finitely many derivatives.

If \( L \) has \( k \) derivatives, one can make a dfa recognising \( L \).
The states are strings \( x_1, x_2, \ldots, x_k \) representing the derivatives \( L_{x_1}, L_{x_2}, \ldots, L_{x_k} \).
The transition rule \( \delta(x_i, a) \) is the unique \( x_j \) with \( L_{x_j} = L_{x_ia} \).
The starting state is the unique state \( x_i \) with \( L_{x_i} = L \).
A state \( x_i \) is accepting iff \( \varepsilon \in L_{x_i} \) iff \( x_i \in L \).
Repetition 4 – Minimal DFA

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

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

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

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

Example 4.1
Let \( n = |\Sigma| \) and \( L = \{ w : \exists a \in \Sigma \ [a \text{ occurs in } w \text{ at least twice}] \} \).

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

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

Can on do better with some other mechanism?
If \((Q, \Sigma, \delta, s, F)\) is a non-deterministic finite automaton (nfa) then \(\delta\) is a relation and not a function, that is, for \(q \in Q\) and \(a \in \Sigma\) there can be several \(p \in Q\) with \((q, a, p) \in \delta\).

A run of an nfa on a word \(a_1a_2\ldots a_n\) is a sequence \(q_0q_1q_2\ldots q_n \in Q^*\) such that \(q_0 = s\) and \((q_m, a_{m+1}, q_{m+1}) \in \delta\) for all \(m < n\).

If \(q_n \in F\) then the run is “accepting” else the run is “rejecting”.

The nfa accepts a word \(w\) iff it has an accepting run on \(w\); 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 00111: Accepting runs $s(0)s(0)o(1)p(1)q(1)r$ and $s(0)o(0)o(1)p(1)q(1)r$; the rejecting run $s(0)s(0)s(1)o(1)p(1)q$ is not relevant.

Input 11111: No accepting run; only possible run $s(1)o(1)p(1)q(1)r(1)$... gets stuck.

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

Quiz: How many runs for 1001001 are accepting?
Exponential Improvement

The language from Example 4.1 has an NFA with $n + 2$ states while a DFA needs $2^n + 1$ states; here for $n = 4$. 

![NFA Diagram]

- Start state: $\emptyset$
- States: $\{0\}, \{1\}, \{2\}, \{3\}$
- Transitions:
  - From $\emptyset$: 0, 1, 2, 3
  - From $\{0\}$: 0, 1, 2, 3
  - From $\{1\}$: 0, 2, 3
  - From $\{2\}$: 0, 1, 3
  - From $\{3\}$: 0, 1, 2
Büchi’s Powerset Construction

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

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

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 $\delta'$ is indeed a deterministic transition function.

Let $w = a_1a_2 \ldots a_m$ be a word. Now let $Q_0 = \{s\}$ and, for $k = 0, 1, \ldots, m - 1$, $Q_{k+1} = \delta'(Q_k, a_{k+1})$ be the run (sequence of states) of the dfa while processing $w$.

If the dfa accepts $w$ then there is $q_m \in Q_m \cap F$ and one can select, for $k = m - 1, n - 2, \ldots, 1, 0$, states $q_k \in Q_k$ with $q_{k+1} \in \delta(q_k, a_k)$. It follows that $q_0q_1 \ldots q_m$ is an accepting run for the nfa.

If the nfa accepts $w$ with an accepting run $q_0q_1 \ldots q_m$ then $q_0 = s$, $q_0 \in Q_0$ and, for $k = 0, 1, \ldots, m - 1$, it follows from $q_k \in Q_k$ that $q_{k+1} \in \delta(q_k, a_{k+1})$ and thus $q_{k+1} \in Q_{k+1}$. Thus $q_m \in Q_m \cap F$ and the run of the dfa is accepting as well.
Example 4.6

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

Then the corresponding DFA has the four states \(\emptyset, \{s\}, \{q\}, \{s, q\}\) where \(\{q\}, \{s, q\}\) are the final states and \(\{s\}\) is the initial state. The transition function \(\delta'\) of the DFA is given as

\[
\begin{align*}
\delta'(\emptyset, a) &= \emptyset \text{ for } a \in \{0, 1\}, \\
\delta'(\{s\}, 0) &= \{s, q\}, \quad \delta'(\{s\}, 1) = \{s\}, \\
\delta'(\{q\}, a) &= \emptyset \text{ for } a \in \{0, 1\}, \\
\delta'(\{s, q\}, 0) &= \{s, q\}, \quad \delta'(\{s, q\}, 1) = \{s\}.
\end{align*}
\]

This automaton can be further optimised: The states \(\emptyset\) and \(\{q\}\) 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 \((Q, \Sigma, \delta, I, F)\) has a set \(I\) of possible initial states and an accepting run is any run starting in one member of \(I\) and finishing in one member of \(F\). Here an example for \(0^*1^* \cup 2^*3^*\).

![Diagram showing initial states and transitions for 0*1+ and 2*3+](image)
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.
Exercise 4.10
Consider \( L = \{ w \in \Sigma^* : \text{some } a \in \Sigma \text{ does not occur in } w \} \).
Show that there is an nfa with an initial set of states which recognises \( L \) using \(|\Sigma|\) states.
Show that every complete dfa recognising \( L \) needs \( 2^{\mid\Sigma\mid} \) states; here complete means that the dfa never gets stuck.

Exercise 4.11
Given an nfa \((\{q_0, q_1, \ldots, q_{n-1}\}, \{0, 1\}, \delta, q_0, \{q_0\})\) with \( \delta \) allowing on 1 to go from \( q_m \) to \( q_{(m+1) \mod n} \) and on 0 to go from \( q_m \) with \( m > 0 \) to either \( q_0 \) or \( q_m \). One cannot go to any state from \( q_0 \) on 0. Determine the number of states of an equivalent complete and minimal dfa. Explain how this number of states is found.
Theorem 4.12
Every language generated by a regular grammar is also recognised by an nfa.

Let $(N, \Sigma, P, S)$ be a grammar generating $L$.

Normalisations:

- Replace in $N$ each rule $A \rightarrow w$ with $w \in \Sigma^+$ by $A \rightarrow wB$, $B \rightarrow \varepsilon$ for new non-terminal $B$;
- Replace in $N$ each rule $A \rightarrow a_1a_2 \ldots a_nB$ by new rules $A \rightarrow a_1C_1$, $C_1 \rightarrow a_2C_2$, $\ldots$, $C_{n-1} \rightarrow a_nB$ for new non-terminals $C_1, C_2, \ldots, C_{n-1}$.

Now make nfa $(N, \Sigma, \delta, S, F)$ with $\delta(A, a) = \{B : A \Rightarrow^* aB\}$ and $F = \{C \in N : C \Rightarrow^* \varepsilon\}$.
Example 4.13

$L = 0123^*$. 

Grammar \((\{S, T\}, \{0, 1, 2\}, P, S)\) with rules 

\[ P = \{ S \rightarrow 012|012T, \, T \rightarrow 3T|3 \} \]. 

Updated to grammar with non-terminals 

\[ N = \{ S, S', S'', S''', T, T' \} \] and rules 

\[ S \rightarrow 0S', \, S' \rightarrow 1S'', \, S'' \rightarrow 2S'''|2T, \, S''' \rightarrow \varepsilon, \, T \rightarrow 3T|3T', \, T' \rightarrow \varepsilon \]. 

NFA \((N, \{0, 1, 2, 3\}, \delta, S, \{S''', T'\})\) with 

\[ \delta(S, 0) = \{S'\}, \delta(S', 1) = \{S''\}, \delta(S'', 2) = \{S''', T\}, \delta(T, 3) = \{T, T'\} \] and 

\[ \delta(A, a) = \emptyset \] in all other cases.

Accepting run for \(0\ 1\ 2\) is \(S (0) \ S' (1) \ S'' (2) \ S'''\) and for \(0\ 1\ 2\ 3\ 3\ 3\) is \(S (0) \ S' (1) \ S'' (2) \ T (3) \ T (3) \ T (3) \ T'.\)
Exercises for Grammar to NFA

Exercise 4.14
Let the regular grammar \( \{ \{ S, T \}, \{0, 1, 2\}, P, S \} \) with the rules \( P \) being \( S \rightarrow 01T | 20S \), \( T \rightarrow 01 | 20S | 12T \). Construct a non-deterministic finite automaton recognising the language generated by this grammar.

Exercise 4.15
Let \( L \) be generated by the regular grammar \( \{ \{ S \}, \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}, P, S \} \) where the rules in \( P \) are all the rules of the form \( S \rightarrow aaaaaaS \) for some digit \( a \) and the rule \( S \rightarrow \varepsilon \). What is the minimum number of states of a non-deterministic finite automaton recognising \( L \)? What is the trade-off of the nfa compared to the minimal dfa for \( L \)? Prove your answers.
Corollary 4.16: 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 Jaffee’s Matching Pumping Lemma; 
(g) $L$ has only finitely many derivatives.
Size of Expressions

Example 4.17
The language

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

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

Note that $\{0, 1\}^2$ is written as $\{0, 1\} \cdot \{0, 1\}$ and $\{0, 1\}^3$ is written as $\{0, 1\} \cdot \{0, 1\} \cdot \{0, 1\}$ 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.
Theorem 4.18
Let \( p_1, p_2, p_3, \ldots \) be the prime numbers in ascending order. The language \( L_n = \{0^{p_1}\}^+ \cap \{0^{p_2}\}^+ \cap \ldots \cap \{0^{p_n}\}^+ \) has a regular expression which can be written down with approximately \( O(n^2 \log(n)) \) symbols if one can use intersection. However, every \( \text{nfa} \) recognising \( L_n \) has at least \( 2^n \) states and every regular expression for \( L_n \) only using union, concatenation and Kleene star needs at least \( 2^n \) symbols.

The expression - when written \( \text{000} \) in place of \( 0^3 \) and so on – has length \( O(n^2 \log(n)) \) and shortest word has length \( p_1 \cdot p_2 \cdot \ldots \cdot p_n \geq 2^n \). Shortest word recognised by \( \text{nfa} \) cannot be longer as the number of states, as in the accepting run, no state is repeated. Thus \( \text{nfa} \) has at least \( 2^n \) states.
Proposition
If a regular expression \( \sigma \) uses only lists of members, union, concatenation and Kleene star, then the shortest word \( sw(\sigma) \) satisfies \( |sw(\sigma)| \leq |\sigma| \).

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

Thus if one writes the Expression from Theorem 4.18 without intersections then its length is at least \( 2^n \).
Length of Short Words

Exercise 4.19
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 $\sigma$ is at most as long as $\sigma$. Either prove it by structural induction or by an assumption of contradiction as in the proof before; both methods are nearly equivalent.

Exercise 4.20
Is Exercise 4.19 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 4.21 [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:

$$(\{(a, b) \cdot (b, c) : a, b, c \in \{1, 2, \ldots, n\}\}^* \cdot (\{\varepsilon\} \cup \{(a, b) : a, b \in \{1, 2, \ldots, n\}\})) \cap (\{(a, b) : a, b \in \{1, 2, \ldots, n\}\} \cdot \{(a, b) \cdot (b, c) : a, b, c \in \{1, 2, \ldots, n\}\}^* \cdot (\{\varepsilon\} \cup \{(a, b) : a, b \in \{1, 2, \ldots, n\}\}))$$
Exercise 4.22
Assume that an nfa of \( k \) states recognises a language \( L \). Show that the language does then satisfy the Block Pumping Lemma with constant \( k + 1 \), that is, given any words \( u_0, u_1, \ldots, u_k, u_{k+1} \) such that their concatenation \( u_0u_1 \ldots u_ku_{k+1} \) is in \( L \) then there are \( i, j \) with \( 0 < i < j \leq k + 1 \) and

\[
  u_0u_1 \ldots u_{i-1}(u_iu_{i+1} \ldots u_{j-1})^*u_ju_{j+1} \ldots u_{k+1} \subseteq L.
\]

Exercise 4.23
Given numbers \( n, m \) with \( n > m > 2 \), provide an example of a regular language where the Block pumping constant is exactly \( m \) and where every nfa needs at least \( n \) states.
Exercises 4.24 - 4.28

Let \( n \) be the size of the alphabet \( \Sigma \) and assume \( n \geq 2 \). Determine the size of the smallest NFA and DFA for the following languages in dependence of \( n \). Explain the results and construct the automata for \( \Sigma = \{0, 1\} \) (4.28: \( \{0, 1, 2\} \)).

Exercise 4.24
\[ H = \{vawa : v, w \in \Sigma^*, a \in \Sigma\}. \]

Exercise 4.25
\[ I = \{ua : u \in (\Sigma - \{a\})^*, a \in \Sigma\}. \]

Exercise 4.26
\[ J = \{abuc : a, b \in \Sigma, u \in \Sigma^*, c \in \{a, b\}\}. \]

Exercise 4.27
\[ K = \{avbwc : a, b \in \Sigma, v, w \in \Sigma^*, c \in \Sigma - \{a, b\}\}. \]

Exercise 4.28
\[ L = \{w : \exists a, b \in \Sigma [w \in \{a, b\}^*]\}. \]
Exercises 4.29, 4.30 and 4.31

Exercise 4.29
Show that an nfa for the language \( \{0000000\}^* \cup \{00000000\}^* \) needs only 16 states while the constant for Jaffe’s pumping lemma is 56.

Exercise 4.30
Generalise the idea of Exercise 4.29 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 4.31
Determine the constant of Jaffe’s pumping lemma and the sizes of minimal nfa and dfa for \( (\{00\} \cdot \{00000\}) \cup (\{00\}^* \cap \{000\}^*).\)