Theoretical Computer Science

Lab Session 9

April 1, 2021

Agenda

- Non-deterministic FSA
- NDFSA to (D)FSA

Non-deterministic FSA.

Non-deterministic Finite State Automata (NDFSA)

Definition: NDFSA

A NDFSA is a tuple $\langle Q, I, \delta, q_0, F\rangle$, where Q, I, q_0, F are defined as in (D)FSA and the transition function is defined as

$$\delta : Q \times I \rightarrow \mathbb{P}(Q)$$

$\mathbb{P}$ is the powerset function (i.e. set of all possible subsets)

Non-deterministic Finite State Automata (NDFSA)

Definition: NDFSA

A NDFSA is a tuple $\langle Q, I, \delta, q_0, F \rangle$, where Q, I, q_0, F are defined as in (D)FSA and the transition function is defined as

$$\delta : Q \times I \to \mathbb{P}(Q)$$

$\mathbb{P}$ is the powerset function (i.e. set of all possible subsets)

A NDFSA modifies the definition of a FSA to permit transitions at each stage to either zero, one, or more than one states.

The extended transition δ^* for NDFSA

the extended transition δ^* for NDFSA

Let $M = \langle Q, I, \delta, q_0, F \rangle$ be a NDFSA. We define the extended transition function as follows:

1. For every $q \in Q$, $\delta^*(q, \epsilon) = \{q\}$
2. For every $q \in Q$, every $y \in I^*$, and every $i \in I$,

$$\delta^*(q, yi) = \bigcup_{q' \in \delta^*(q,y)} \delta(q', i)$$

Acceptance by a NDFSA

Acceptance by a NDFSA

Let $M = \langle Q, I, \delta, q_0, F\rangle$ be a NDFSA, and let $x \in I^*$. The string x is accepted by M iff

$$\delta^*(q_0, x) \cap F \neq \varnothing$$

and it is rejected by M otherwise.

Notion: Among the various possible runs (with the same input) of the NDFSA, it is sufficient that one of them succeeds to accept the input string.

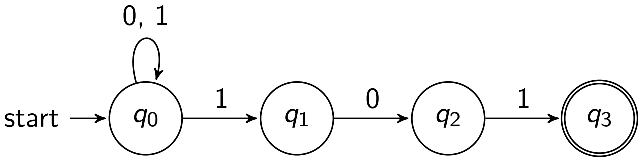
Exercises on NDFSA

Build NDFSAs that recognise the following languages:

- $L_a = \{x \in \{0,1\}^* \mid\ x \text{ ends with } 101\}$;
- $L_b = \{xy \mid x \in \{a\}^* \wedge y \in \{a,b\}^* \wedge$ y does not start with 'b' $\wedge$ every 'a' in y is followed by exactly one 'b'$\}$;
- $L_c = \{x \in \{a,b,c\}^* \mid x$ ends with either ab, bc or $ca\}$;

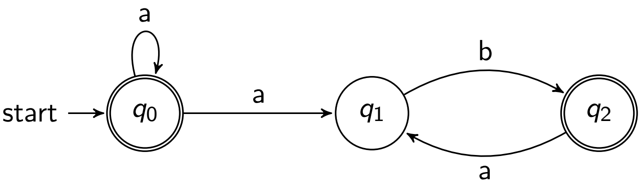
Solution (1)

NDFSA that recognises the language:
$L_a = \{x \in \{0,1\}^* \mid \ x \text{ ends with } 101\}$



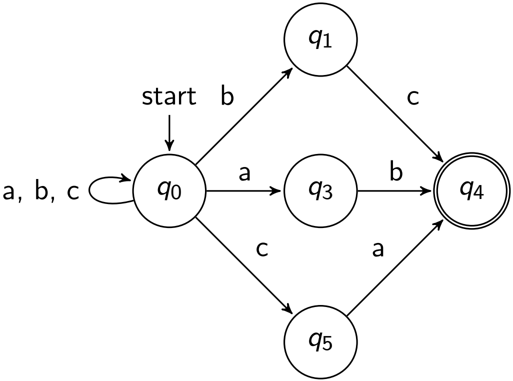
Solution (2)

NDFSA that recognises the language:
$L_b = \{xy \mid x \in \{a\}^* \wedge y \in \{a, b\}^* \wedge$ y does not start with 'b' $\wedge$ every 'a' in y is followed by exactly one 'b'$\}$



Solution (3)

NDFSA that recognises the language: $L_c = \{x \in \{a, b, c\}^* \mid x$ ends with either ab, bc or $ca\}$



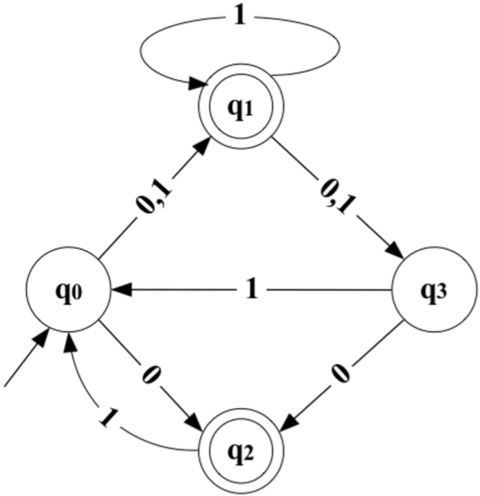
NDFSA to DFSA

Algorithm for NDFSA to DFSA

1. Create state table from the given NDFA
2. Create a blank state table under possible input alphabets for the equivalent DFA
3. Mark the start state of the DFA by q_0 (Same as the NDFA)
4. Find out the combination of States $q_0, q_1, ..., q_n$ for each possible input alphabet
5. Each time we generate a new DFA state under the input alphabet columns, we have to apply step 4 again, otherwise go to step 6
6. The states which contain any of the accepting states of the NDFA are the accepting states of the equivalent DFA

NDFSA to FSA: example

Let us consider the following NDFSA:



NDFSA to FSA: step 1

First, we build a transition table for NDFSA:

q	δ(q,0)	δ(q,1)
$\rightarrow$q0	{q1,q2}	{q1}
*q1	{q3}	{q1, q3}
*q2	$\emptyset$	{q0}
q3	{q2}	{q0}

NDFSA to FSA: step 2

Using table from previous slide, let us create a similar table, but this time for FSA. Initially, the table is empty:

q	δ(q,0)	δ(q,1)
...	...	...

NDFSA to FSA: step 3

We begin by adding the initial state and the set of states:

q	δ(q,0)	δ(q,1)
$\rightarrow$q0	{q1,q2}	{q1}
...	...	...

NDFSA to FSA: step 4

The initial state can take us to two new states that are not yet in the table. Note that we treat a set of states as a single state now!

q	δ(q,0)	δ(q,1)
→q0	{q1,q2}	{q1}
{q1,q2}	...	...
q1	...	...
...	...	...

The next step is to find possible states for $\{q1, q2\}$ and $q3$. For $q3$ it is trivial, you just need to look it up in the original NDFSA table. However, the transition for $\{q1, q2\}$ will be a union of sets:
$\delta(q1, q2, 0) = \delta(q1, 0) \cup \delta(q2, 0) = \{q3\}$
$\delta(q1, q2, 1) = \delta(q1, 1) \cup \delta(q2, 1) = \{q0, q1, q3\}$

NDFSA to FSA: step 5

The initial state can take us to two new states that are not yet in the table. Note that we treat a set of states as a single state now!

q	δ(q,0)	δ(q,1)
→q0	{q1,q2}	{q1}
{q1,q2}	{q3}	{q0,q1,q3}
q1	...	...
...	...	...

The next step is to find possible states for $\{q1, q2\}$ and $q3$. For $q3$ it is trivial, you just need to look it up in the original NDFSA table. However, the transition for $\{q1, q2\}$ will be a union of sets:
$\delta(q1, q2, 0) = \delta(q1, 0) \cup \delta(q2, 0) = \{q3\}$
$\delta(q1, q2, 1) = \delta(q1, 1) \cup \delta(q2, 1) = \{q0, q1, q3\}$

NDFSA to FSA: step 5

Repeat the steps above until we have included all states from the original NDFSA and there are no new states.

q	δ(q,0)	δ(q,1)
→q0	{q1,q2}	{q1}
{q1,q2}	{q3}	{q0,q1,q3}
q1	{q3}	{q1,q3}
...	...	...

NDFSA to FSA: step 6

After repeating previous steps and depicting final states (step 6) we will arrive to the following table:

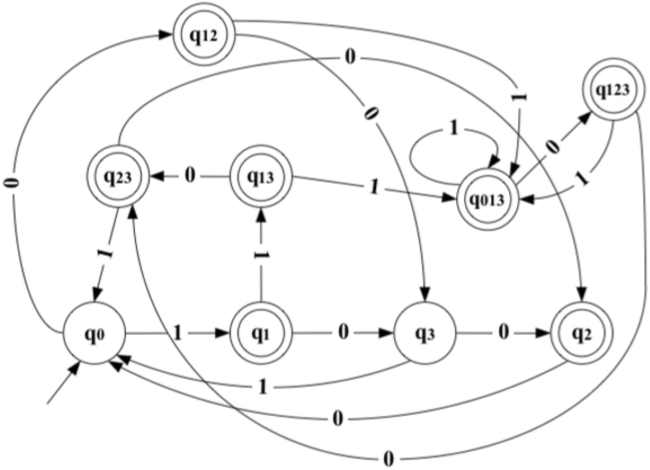
q	δ(q,0)	δ(q,1)
$\rightarrow$q0	{q1,q2}	{q1}
*{q1,q2}	{q3}	{q0,q1,q3}
*q1	{q3}	{q1,q3}
q3	{q2}	{q0}
*{q0,q1,q3}	{q1,q2,q3}	{q0,q1,q3}
*{q1,q3}	{q2,q3}	{q0,q1,q3}
*q2	$\varnothing$	{q0}
*{q1,q2,q3}	{q2,q3}	{q0,q1,q3}
*{q2,q3}	{q2}	{q0}

NDFSA to FSA: result (table representation)

q	δ(q,0)	δ(q,1)
$\rightarrow$q0	{q1,q2}	{q1}
*q12	{q3}	{q0,q1,q3}
*q1	{q3}	{q1,q3}
q3	{q2}	{q0}
*q013	{q1,q2,q3}	{q0,q1,q3}
*q13	{q2,q3}	{q0,q1,q3}
*q2	$\varnothing$	{q0}
*q123	{q2,q3}	{q0,q1,q3}
*q23	{q2}	{q0}

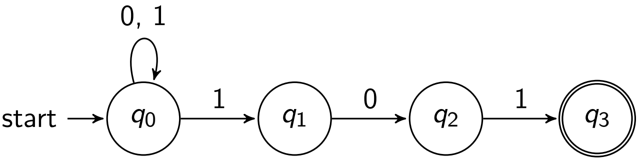
NDFSA to FSA: result (graphical representation)

Finally, we can build the resulting DFSA:



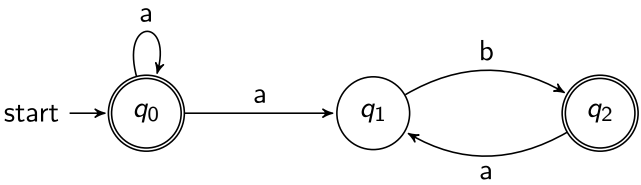
Exercise 1

Build DFSA from the NDFSA that recognizes the language:
$L_1 = \{x \in \{0,1\}^* \mid \ x \text{ ends with } 101\}$



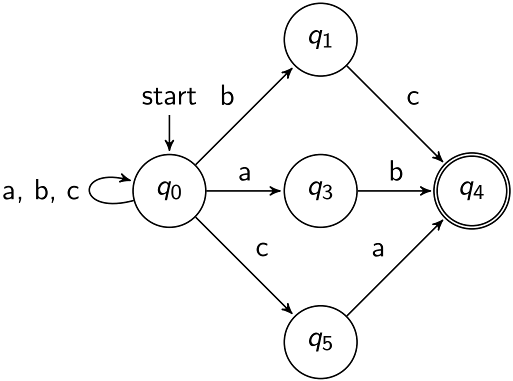
Exercise 2

Build DFSA from the NDFSA that recognizes the language:
$L_2 = \{xy \mid x \in \{a\}^* \wedge y \in \{a, b\}^* \wedge$ y does not start with 'b' $\wedge$ every 'a' in y is followed by exactly one 'b'$\}$



Exercise 3

Build DFSA from the NDFSA that recognizes the language:
$L_3 = \{x \in \{a, b, c\}^* \mid x \text{ ends with either } ab,\ bc \text{ or } ca\}$



Practice Exercises on NDFSA

Build complete NDFSAs that recognize the following languages:
Let Σ be the alphabet $\Sigma = \{0, 1\}$

- $L_0 = \{x \in \Sigma^* \mid \ x \text{ starts with } 1\}$;
- $L_1 = \{x \in \Sigma^* \mid \ x \text{ does not begin with } 1\}$;
- $L_2 = \{x \in \Sigma^* \mid \ \text{any } 0 \text{ in } x \text{ is followed by at least a } 1\}$.
  Strings example: 010111, 1111, 01110111011.
- $L_3 = \{x \in \Sigma^* \mid \ x \text{ ends with } 00\}$;
- $L_4 = \{x \in \Sigma^* \mid \ x \text{ contains exactly } 3 \text{ zeros}\}$;

Practice Exercises on NDFSA

Build complete NDFSAs that recognize the following languages:
Let Σ be the alphabet $\Sigma = \{a, b\}$

- $L_5 = \{x \in \Sigma^* \mid$ every a in x (if there are any) is followed immediately by $bb\}$.
- $L_6 = \{x \in \Sigma^* \mid$ x ends with b and does not contain the substring $aa\}$.
- $L_7 = \{x \in \Sigma^* \mid$ x contains the substring $abbaab\}$;
- $L_8 = \{x \in \Sigma^* \mid$ x has an even number of a's and an even number of b's$\}$;

Practice Exercises on NDFSA

Build complete NDFSAs accepting the following languages over the alphabet $\Sigma = \{0, 1\}$

- $L_a = \{x \in \Sigma^* \mid x$ is a binary representation of an integer divisible by 5 and it begins with $1\}$;
- $L_b = \{x \in \Sigma^* \mid |x| \geq 2 \wedge$ final two symbols are the same$\}$;

Build a complete FSA accepting the following language over the alphabet $\Sigma = \{a, b, c\}$

- $L_c = \{x \in \Sigma^* \mid$ the substring abc in x occurs an odd number of times$\}$.

Practice Exercises on NDFSA

NDFSAs are no more powerful than FSAs. An FSA can be turned into an NDFSA that accepts the same language. Provide equivalent FSAs for previous exercises using the algorithm seen during the lecture.