The Australian National University
School of Computing
Dirk Pattinson and Victor Rivera

Semester 2, 2021
Practical Session 8

Foundations of Computation

The practical contains a number of exercises designed for the students to practice the course content. During the practical session, the tutor will work through some of these exercises while students will be responsible for completing the remaining exercises in their own time. There is no expectation that all the exercises will be covered in the practical session.

Covers: Lecture Material Week 8

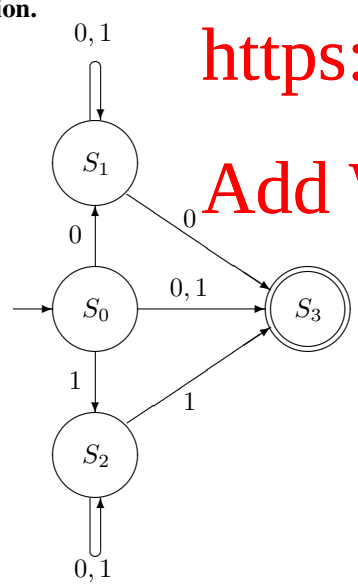
At the end of this tutorial, you will be able to

- minimise FSAs;
- construct Non-Deterministic Finite Automata from (ϵ-) Non-Deterministic Finite Automata;
- build Regular Expressions given a language;
- construct an equivalent Deterministic Finite Automaton from a Non-Deterministic Finite Automaton;
- construct an equivalent (ϵ) Non-Deterministic Finite Automaton from a Regular Expression.

Exercise 1 **NFA to DFA conversion**

1. Construct a *non-deterministic* finite state automaton that accepts the language of binary strings that begin and end with the same bit. Try to use as few states as you can (Note that strings that are one letter long satisfy this predicate, and so should be accepted).

   Solution.

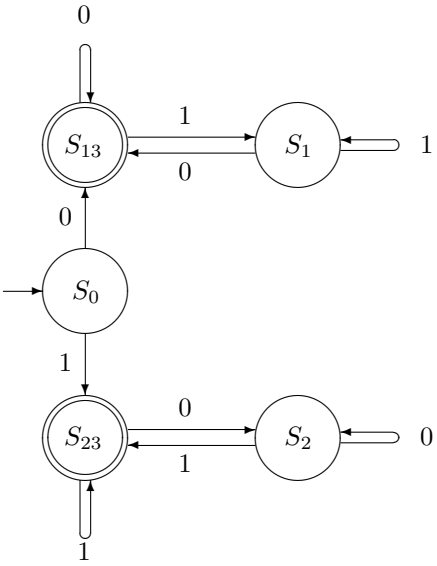


2. Use the subset construction algorithm to construct an equivalent *deterministic* FSA from your previous answer.

   Solution.

   The following table demonstrate implementation of subset construction algorithm. The state labelled with $\rightarrow$ is the initial state while the states denoted with $\odot$ are the final states.

State	0	1
$\varnothing$	$\varnothing$	$\varnothing$
$\rightarrow S_0$	S_{13}	S_{23}
$\odot S_{13}$	S_{13}	S_1
$\odot S_{23}$	S_2	S_{23}
S_1	S_{13}	S_1
S_2	S_2	S_{23}



Exercise 2 **Regular Expressions**

Consider the alphabet $\Sigma = \{a, b\}$. Build Regular Expressions for:

1. the set of strings over Σ^* that consists of alternating a's and b's;

   Solution.

   $$\epsilon \mid a)(ba)^*(a \mid b)$$

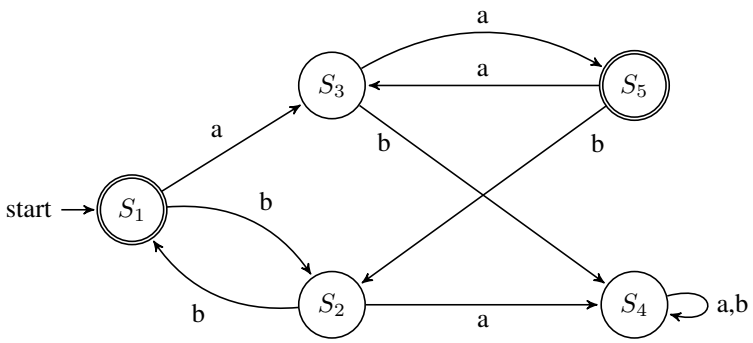
2. the set of strings over Σ^* that contains an even number of a's and b's;

   Solution.

   $$(aa \mid bb \mid (ab \mid ba)(aa \mid bb)^*(ab \mid ba))^*$$

Exercise 3 **Identifying Equivalent States**

Consider the DFA given by the following state diagram:



Identify all equivalent states of this automation using the algorithm given in the lectures.

Solution. To simplify notation, we just write 1 for S_1 and similarly for all other states.

1. The initial partition just distinguishes final states and non-final states. That is, we start with $[[1, 5], [2, 3, 4]]$.

2. We test $[2, 3, 4]$ for splitting. Since $2 \xrightarrow{b} 1$ and $3 \xrightarrow{b} 4$ and 1 and 4 are (by the previous step) known to be non-equivalent, they need to be in different partitions.

   Similarly, $4 \xrightarrow{a} 4$ and $3 \xrightarrow{a} 5$ and 4 and 5 are (by the previous step) non-equivalent, they need to be in different paritions.

   This means that we separate all states in $[2, 3, 4]$ in the next partition and obtain $[[1, 5], [2], [3], [4]]$ as the second partition.

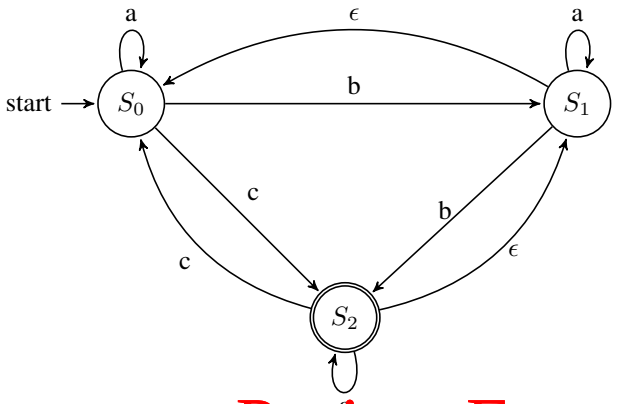
3. We now test $[1,5]$ for splitting. We have $1 \xrightarrow{a} 3$ and $5 \xrightarrow{a} 3$ so they produce equivalet states on a. Moreover, $1 \xrightarrow{b} 2$ and $5 \xrightarrow{b} 2$ so they also produce equivalent (even identical) states on b. That means that we don't need to split $[1,5]$.

4. As the singletion sets $[2]$, $[3]$ and $[4]$ cannot be split, no more splittings are possible, and $[[1,5],[2],[3],[4]]$ is the final partition.

That is, only states S_1 and S_5 are equivalent in the above automation.

Exercise 4 **From ϵ-NFAs to NFAs**

Consider the ϵ-NFA given by the following state diagram:



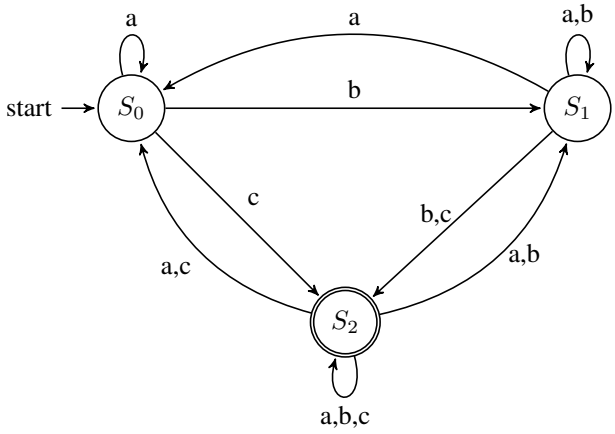
1. Compute the $\epsilon-$closure of each state.

   Solution.

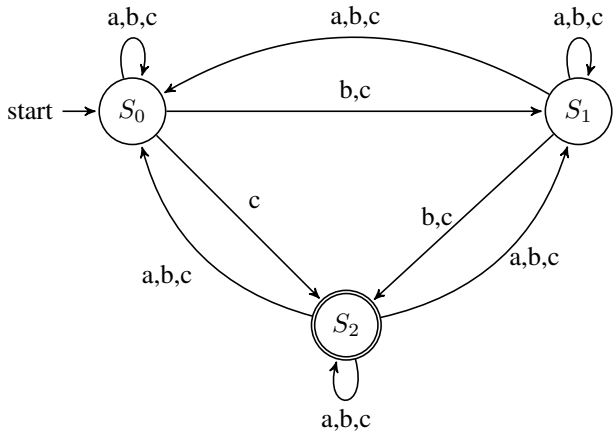
   - eclose$(S_0) = \{S_0\}$
   - eclose$(S_1) = \{S_1, S_0\}$
   - eclose$(S_2) = \{S_2, S_1, S_0\}$

2. Convert the automaton to an NFA using the algorithm given in the lectures.

   Solution.



Alternatively, you can follow the algorithm from the videos. It would yield the following NFA:



Exercise 5 **From Regular Expressions to ϵ-NFAs**

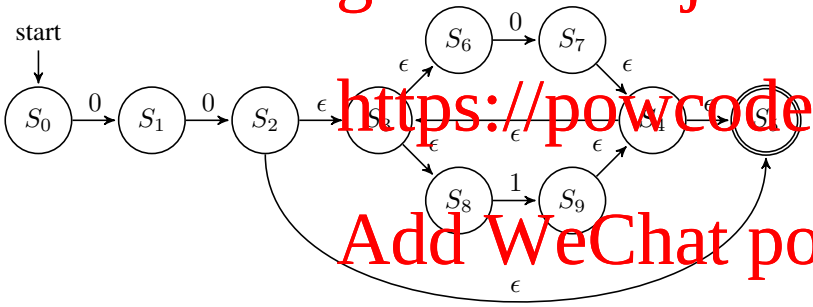
For the regular expression $00(0 \mid 1)^*$

- use the algorithm given in lectures to produce a ϵ-NFA A
- describe the language that A accepts. Phrase your answer in the form $L(A) = \{w \in \Sigma^* \mid P(w)\}$, where $P(w)$ is some predicate on words.

Solution.

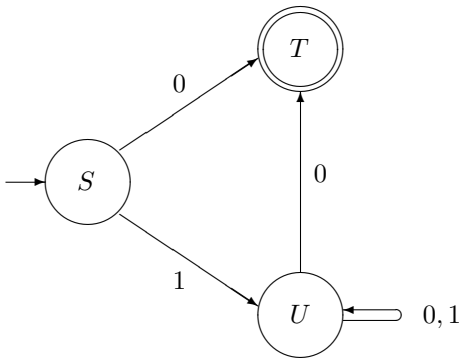
$L(A) = \{w \in \{0,1\}^* \mid w$ contains all binary numbers starting with $00\}$

$A =$



Exercise 6

Consider the non-deterministic finite automaton A:



1. Describe the language that this automaton accepts.

   Solution. The language $L(A)$ is the set of all binary integers that are *even* (or, equivalently, end in 0), but without unnecessary leading 0s.

2. Use the subset construction algorithm to construct an equivalent deterministic FSA from your previous answer.

   Solution. The following table demonstrate implementation of subset construction algorithm. State labelled with $\rightarrow$ is the initial state while the states denoted with $\odot$ are the final states.

State	0	1
$\rightarrow S_s$	S_T	S_U
$\odot S_T$	$S_\emptyset$	$S_\emptyset$
S_U	S_{UT}	S_U
$S_\emptyset$	$S_\emptyset$	$S_\emptyset$
$\odot S_{UT}$	S_{UT}	S_U

   Here, we have started the construction with the initial state, S_S, and have only added states to the transition table that are in fact reachable from the initial state.

Exercise 7 **More Regular Expressions**

Consider the alphabet $\Sigma = \{a, b\}$. Build Regular Expressions for:

1. the set of strings over Σ^* that contains an odd number of a's;

   Solution.

   $$(b \mid ab^*a)^*ab^*$$

2. the set of strings over Σ^* that ends with b and does not contains the substring aa;

   Solution.

   $$(b \mid ab)^*(a \mid \epsilon)b$$

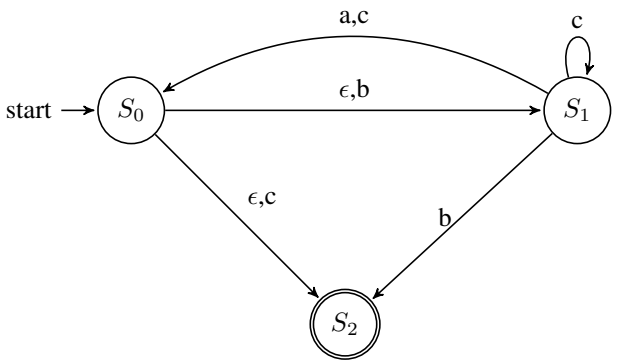
3. the set of string over Σ^* that contains a number of a's that is a multiple of 3.

   Solution.

   $$b^*(ab^*ab^*ab^*)^*$$

Exercise 8 **More from ϵ-NFAs to NFAs**

Consider the ϵ-NFA given by the following state diagram:



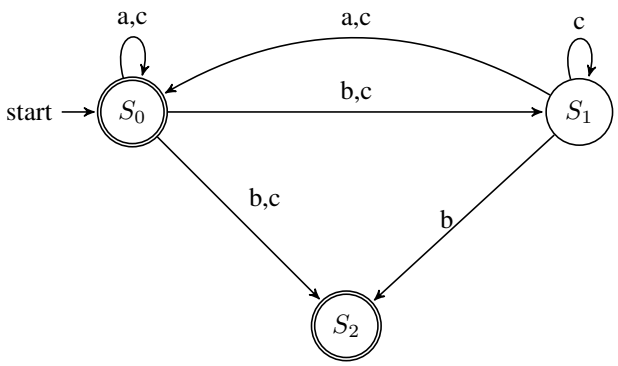
1. Compute the $\epsilon-$closure of each state.

   Solution.

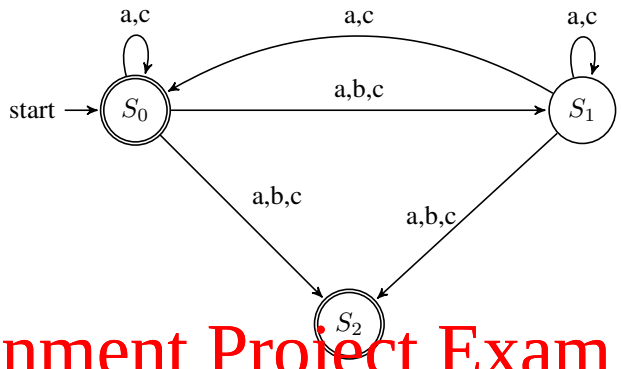
   - eclose$(S_0) = \{S_0, S_1, S_2\}$
   - eclose$(S_1) = \{S_1\}$
   - eclose$(S_2) = \{S_2\}$

2. Convert the automaton to an NFA using the algorithm given in the lectures.

   Solution.



Alternatively, you can follow the algorithm from the videos. It would yield the following NFA:



Exercise 9 **From Regular Expressions to ϵ-NFAs**

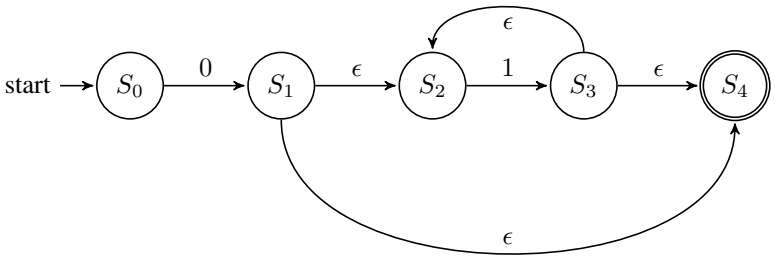
For each of the following regular expressions:

- use the algorithm given in lectures to produce a ϵ-NFA A
- describe the language that A accepts. Phrase your answer in the form $L(A) = \{w \in \Sigma^* \mid P(w)\}$, where $P(w)$ is some predicate on words.

1. 01^*;

   Solution. $L(A) = \{w \in \{0,1\}^* \mid w$ begins with 0 followed by 1's$\}$

   $A =$



2. $(0 \mid 1)01$;

   Solution.

   $L(A) = \{w \in \{0,1\}^* \mid w$ is the 3-bit binary representation of either 1 or 5$\}$

   $A =$