COMP30026 Models of Computation

Lecture 10: NFA Determinisation and DFA Minimisation

Mak Nazecic-Andrlon and William Umboh

Semester 2, 2024

DFAs vs NFAs

Machines are equivalent if and only if they recognize the same language.

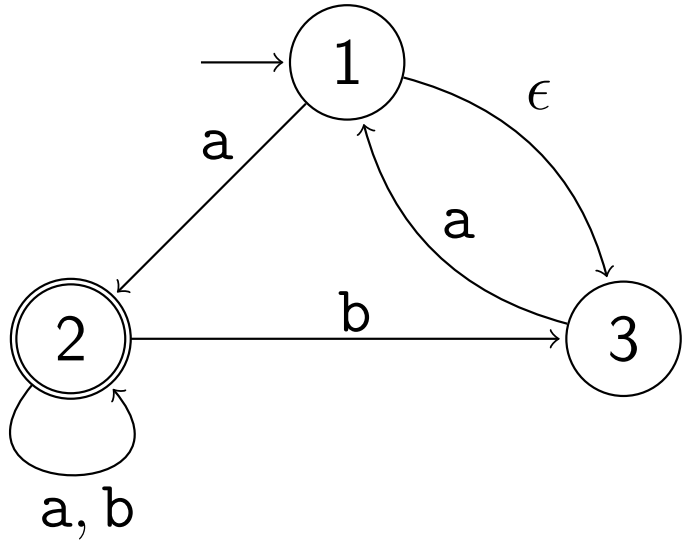
Theorem

Every NFA has an equivalent DFA.

How? Simulate an NFA using a DFA.

DFAs vs NFAs

Consider the NFA on the right. We can systematically construct an equivalent DFA from the NFA.



The DFA's start state is $\{1, 3\}$.
From $\{1, 3\}$ a takes us to $\{1, 2, 3\}$.
From $\{1, 2, 3\}$, a takes us back to $\{1, 2, 3\}$, b takes us to $\{2, 3\}$.

Any state S which contains an accept state from the NFA will be an accept state for the DFA. Here we mark accept states with a star.

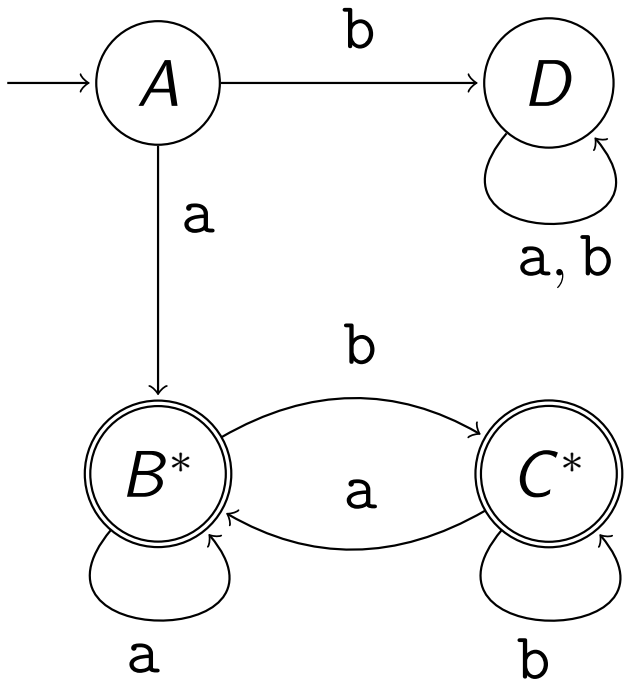
	a	b
A=$\{1,3\}$	B*	–
B*=$\{1,2,3\}$	B*	C*
C*=$\{2,3\}$	B*	C*

DFAs vs NFAs

Here is the equivalent DFA that we derive.

Any state S which contains an accept state from the NFA (in this case the NFA has just one, namely state 2) becomes an accept state for the DFA.

We add (dead) state D that corresponds to the empty set.



	a	b
$A = \{1, 3\}$	B^*	D
$B^* = \{1, 2, 3\}$	B^*	C^*
$C^* = \{2, 3\}$	B^*	C^*
$D = \emptyset$	D	D

The Subset Construction, More Formally

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA.
For all $R \subseteq Q$, let $E(R)$ be the ϵ closure of R. That is,

$$E(R) = \left\{ q \in Q \;\middle|\; \begin{array}{l} q \text{ is reachable from some } r \in R \text{ by} \\ \text{following 0 or more } \epsilon \text{ transitions} \end{array} \right\}.$$

Let $M = (Q', \Sigma, \delta', q_0', F')$ be a DFA such that

- $Q' = \mathcal{P}(Q)$,
- $q_0' = E(\{q_0\})$,
- $F' = \{R \in Q' \mid R \text{ contains an accept state of } N\}$.
- $\delta'(R, a) = \bigcup_{r \in R} E(\delta(r, a))$.

Theorem

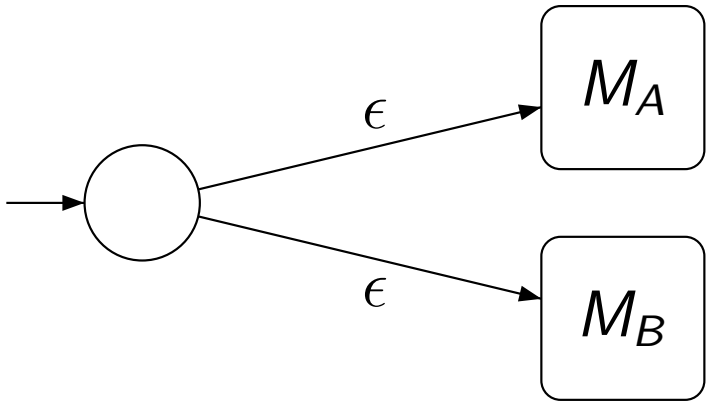
The NFA N is equivalent to the DFA M.

Closure Under Union

Theorem

Let A and B be regular languages. Then $A \cup B$ is a regular language.

Proof idea: By definition, A and B are recognised by some DFAs M_A and M_B, respectively. Then this NFA recognises $A \cup B$:



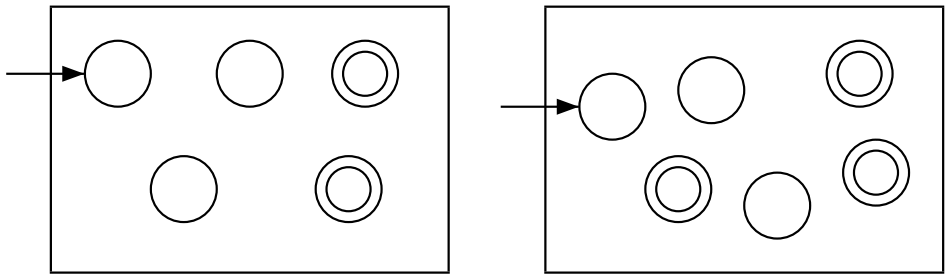
The ϵ-transitions go to the start states of M_A and M_B.

Closure Under Concatenation

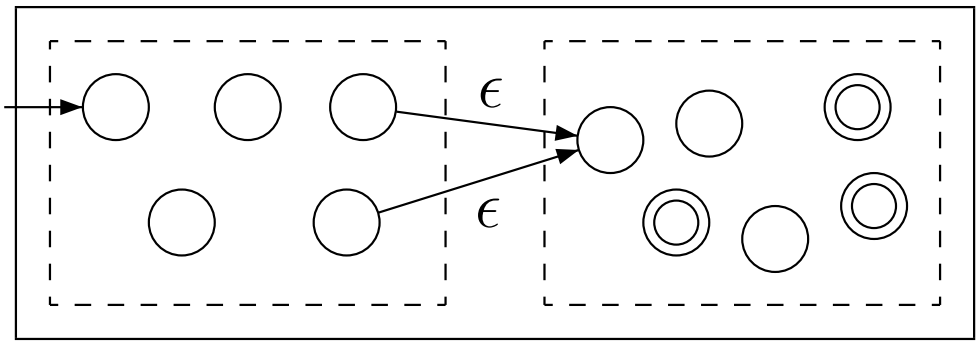
Theorem

Let A and B be regular languages. Then $A \circ B$ is a regular language.

Proof idea: Suppose these are DFAs for A and B, respectively:



Then this NFA recognises $A \circ B$:



That Last Construction, More Formally

Let recognisers for A and B be these DFAs, respectively, where Q and Q' are disjoint:

- $M_A = (Q, \Sigma, \delta, q_0, F)$
- $M_B = (Q', \Sigma, \delta', q_0', F')$

Let N be the NFA $(Q \cup Q', \Sigma, \delta'', \{q_0\}, F')$, where

$$\delta''(q, v) = \begin{cases} \{\delta'(q, v)\} & \text{if } q \in Q' \text{ and } v \in \Sigma \\ \{\delta(q, v)\} & \text{if } q \in Q \text{ and } v \in \Sigma \\ \{q_0'\} & \text{if } q \in F \text{ and } v = \epsilon \end{cases}$$

Theorem

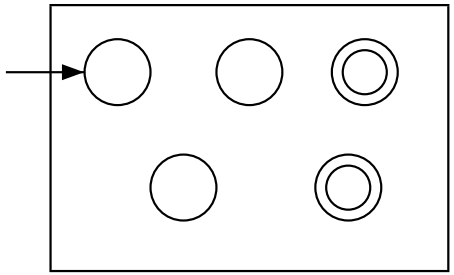
The NFA N recognises the language $A \circ B$.

Closure Under Kleene Star

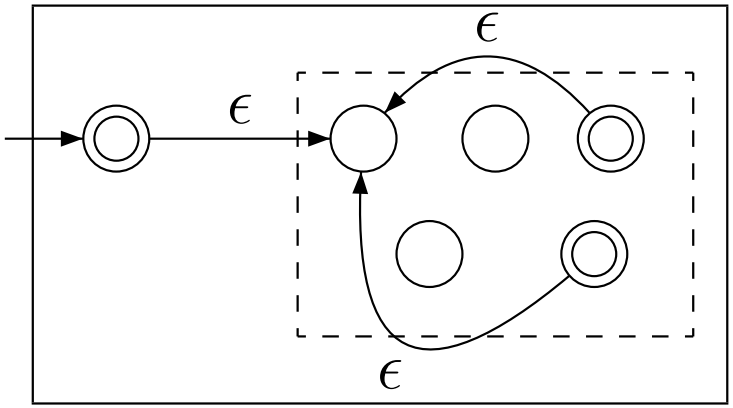
Theorem

Let A be a regular language. Then A^* is a regular language.

Proof idea: Suppose this DFA recognises A:



Here is how we construct an NFA to recognise A^*:



Closure Results for Regular Languages

Regular languages have more closure properties.

Theorem

Let A and B be regular languages over an alphabet Σ. Then, the following languages are also regular:

- $A \cap B$,
- A^c *(the complement of A, that is, $\Sigma^* \setminus A$),*
- $A \setminus B$ *(follows from the first two, since $A \setminus B = A \cap B^c$),*
- *the reversal of A.*

Corollary

The regular languages are closed under Boolean combination.

Algorithms for Manipulating Automata

For some of these closure results, we will use the tutorials to develop useful DFA manipulation algorithms.

You will see, for example, how to systematically build DFAs for languages $A \cap B$, out of DFAs for A and B.

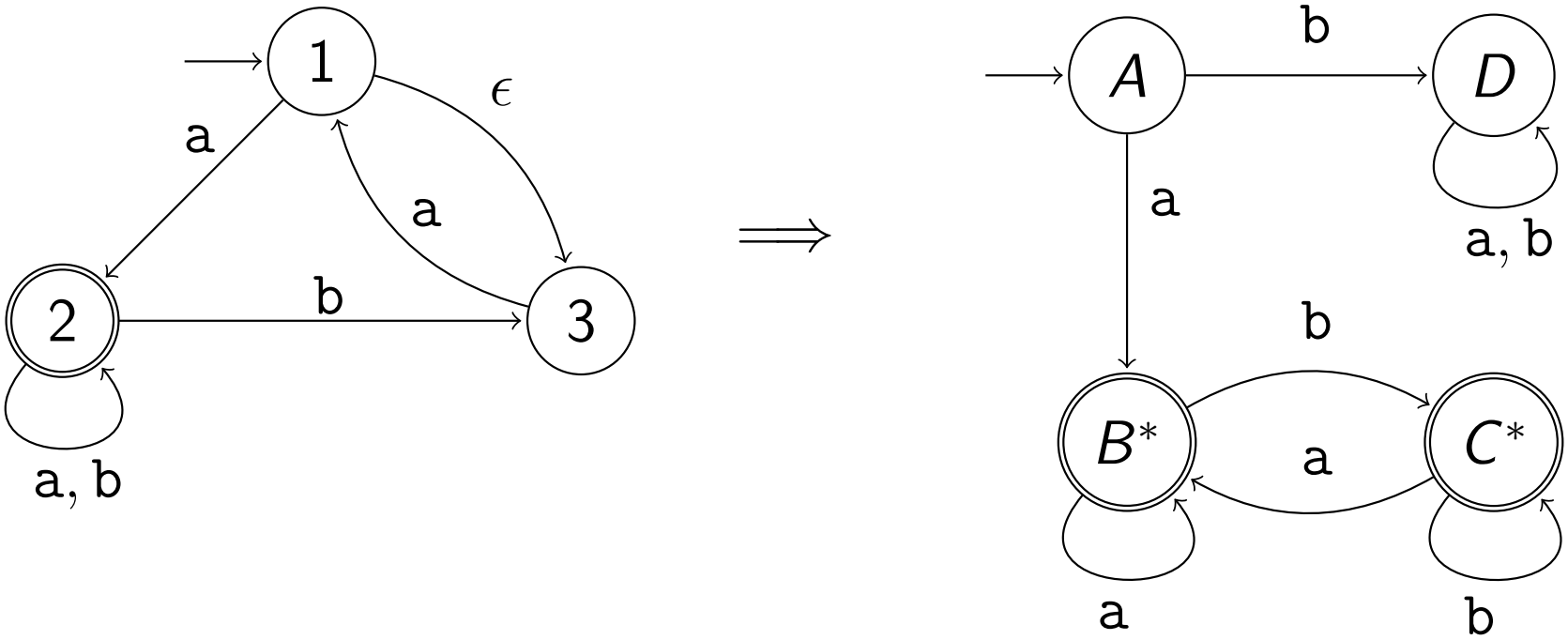
Equivalence of DFAs

We can always find a minimal DFA for a given regular language (by minimal we mean having the smallest possible number of states).

Since a DFA has a unique start state and the transition function is total and deterministic, we can test two DFAs for equivalence by minimizing them.

Minimizing DFAs

There is no guarantee that DFAs that are produced by the various algorithms, such as the subset construction method, will be minimal.



$A = \{1, 3\}$, $B^* = \{1, 2, 3\}$, $C^* = \{2, 3\}$, and $D = \emptyset$.

Generating a Minimal DFA

The following algorithm takes an NFA and produces an equivalent minimal DFA. Of course the input can also be a DFA.

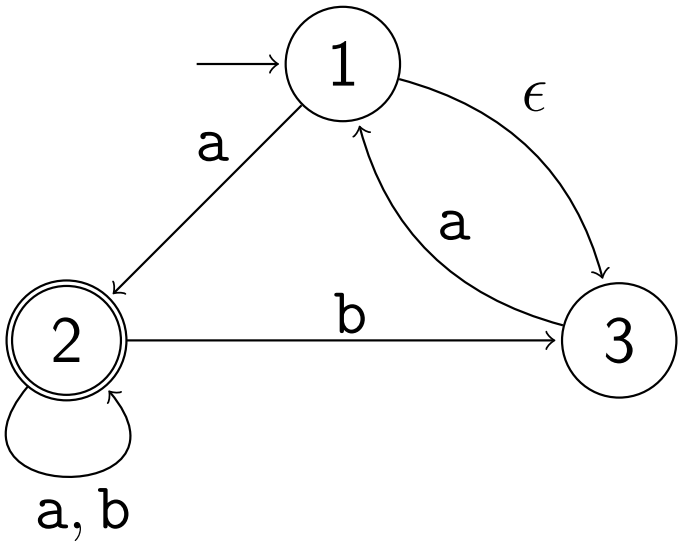
1. Reverse the NFA;
2. Determinize the result;
3. Reverse again;
4. Determinize.

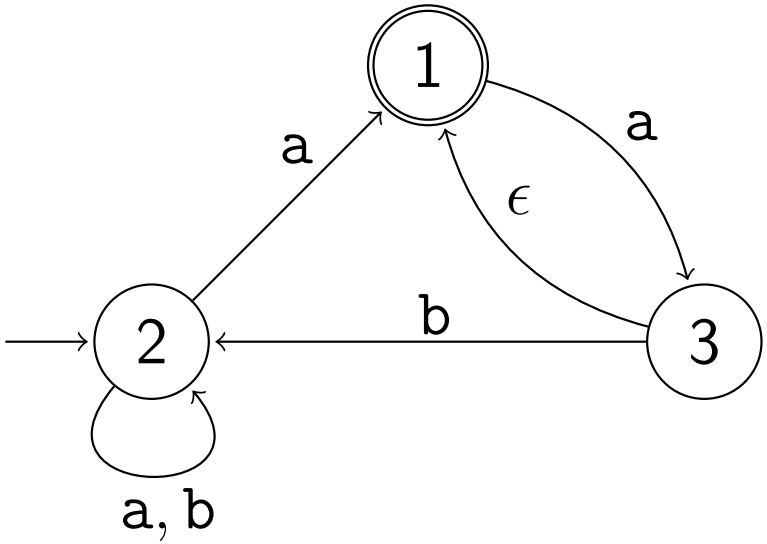
To reverse an NFA A with initial state q_0 and accept states $F \neq \emptyset$:

1. If $F = \{q\}$, let q be the accept state.
2. Otherwise add a new state q which becomes the only accept state; then, for each state in F, add an ϵ transition to q.
3. Reverse every transition in the resulting NFA, making q the initial state and q_0 the (only) accept state.

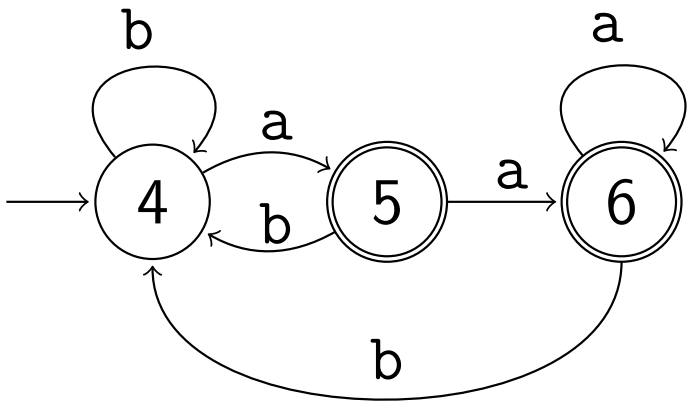
Minimization Example

Consider again the NFA that we determinized two slides ago. Here it is on the left, with its reversal on the right:



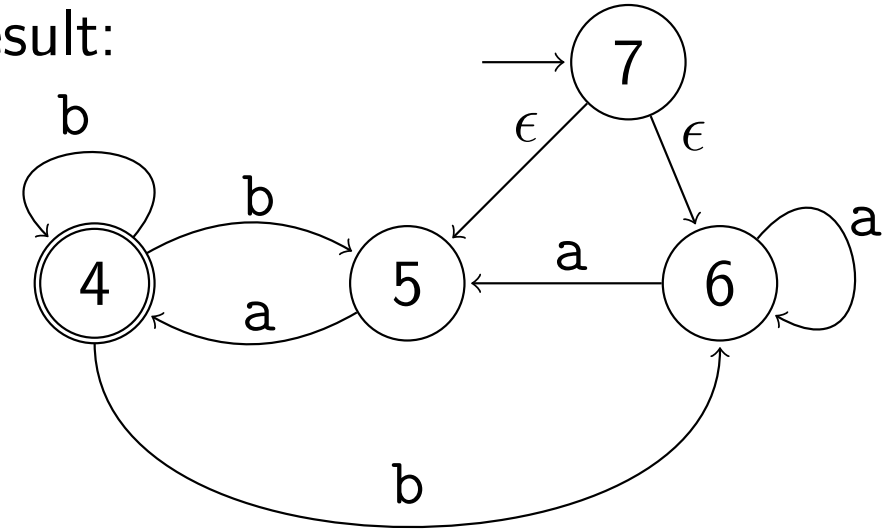


Now make the reversed NFA deterministic (we have renamed the states to avoid later confusion: 4 corresponds to $\{2\}$, 5 to $\{1, 2\}$, and 6 to $\{1, 2, 3\}$).

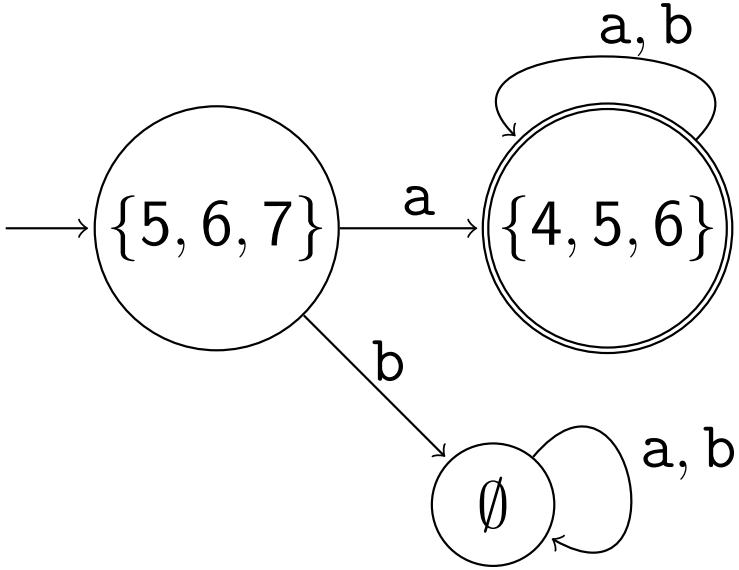


Minimization Example

Now reverse the result:



Finally make the result deterministic:



Next Week

We next look at regular expressions—another way to capture the regular languages

Also, we will use a technique called "pumping", for proving that a language is not regular and introduce context-free languages.