COMP30026 Models of Computation

Regular Languages

Lecture 15

Semester 2, 2020

This Lecture is Being Recorded

DFAs vs NFAs

The class of languages recognised by NFAs is exactly the class of regular languages.

Theore

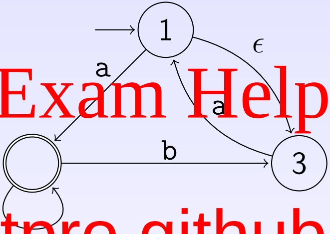
The proo

Given NFA N, we construct DFA M, eac
corresponds to a set of N-states.

If N has k states then M may have up to 2 states (but it will often have far fewer than that).

DFAs vs NFAs

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



The DFA'

From $\{1$

From $\{1$

$\{1, 2, 3\}$, b takes us to $\{2, 3\}$.

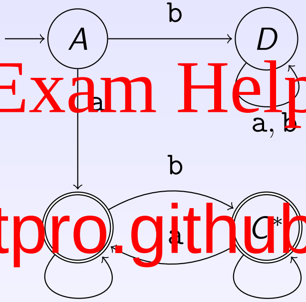
		b
		–
		C*
C*={2,3}	B*	C*

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.

Here is the equivalent DFA that we derive.

Any state … state fro… has just one, namely state 2) becomes an accept state for the DFA.

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



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

More Formally . . .

Let $N = (Q, \Sigma, \delta, q_0, F)$. Let $\xrightarrow{*}_\epsilon$ be the reflexive transitive closure of $\rightarrow_\epsilon$, which in turn is defined by $s \rightarrow_\epsilon s'$ iff $s' \in \delta(s, \epsilon)$.

Let $E(S)$ be the 'ϵ closure' of $S \subseteq Q$, that is, S together with all states reachable from states in S, using only ϵ steps:

We construct $M = (\mathcal{P}(Q), \Sigma, \delta', q_0', F')$

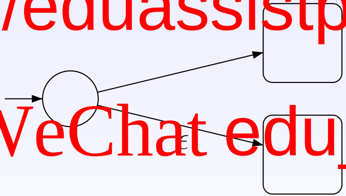
- $q_0' = E(\{q_0\})$.
- $\delta'(S, v) = \bigcup_{s \in S} E(\delta(s, v))$.
- $F' = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$.

Note: This construction may include unreachable states.

Closure Results for Regular Languages

Theorem: The class of regular languages is closed under union.

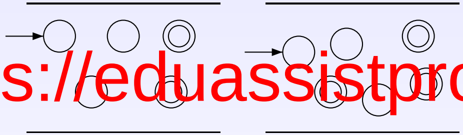
Proof: Let A and B be regular languages, with recognisers M_A and M_B. An NF



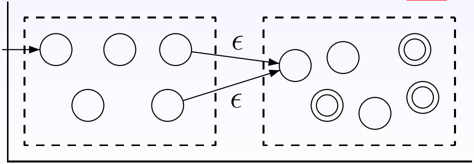
The ϵ-transitions go to the start states of M_A and M_B.

Theorem: The class of regular languages is closed under $\circ$.

Proof: Let A and B be regular languages with these recognisers:



From these we can easily construct an NFA that recognises $A \circ B$:



The Last Construction, Formally

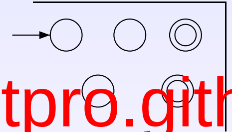
Let recognisers for A and B be these DFAs, respectively:

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

Then a rec $_0$ $F')$, where

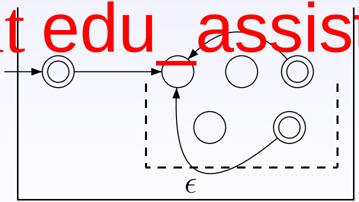
$$\delta'(q, v) = \begin{cases} \{\delta'(q, v)\} & \text{if} \\ \{\delta(q, v)\} & \text{if} \\ \{q'_0\} & \text{if } q \in F \text{ and } v = \epsilon \end{cases}$$

Theorem: The class of regular languages is closed under Kleene star.



Proof:
languag
shown on t

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



Closure Results for Regular Languages

Regular languages have several other closure properties.

They are c

- int
- complement, A^c
- difference (this follows, as $A \setminus B =$

- reversal.

For some o develop

useful DF

You will se or

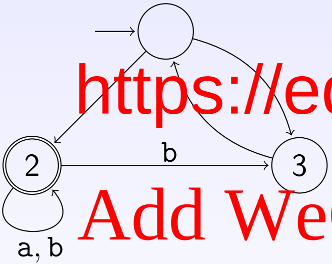
languages $A \cap B$, out of DFAs for A an

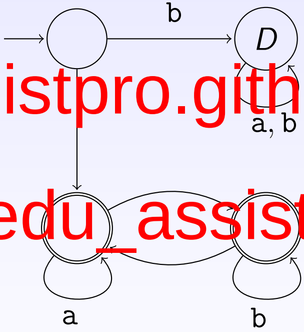
Equivalence of DFAs

We can always find a minimal DFA for a given regular language (by mini ates).

Since a DF on is total and deterministic, we can test two DFAs for by minimizing them.

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.

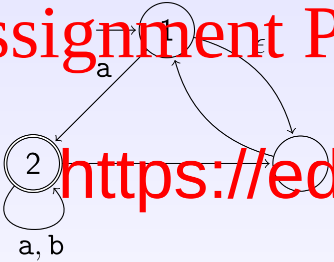
1. Reverse the NFA;
2. De
3. Re
4. De

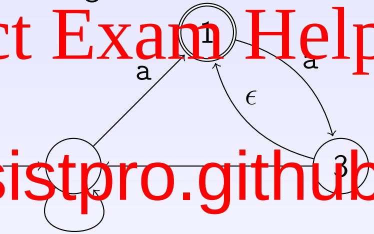
To reverse an NFA A with initial state $\neq \emptyset$:

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.

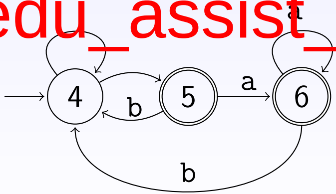
Minimization Example

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

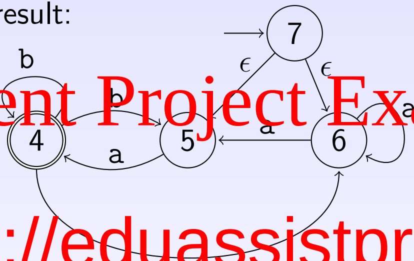




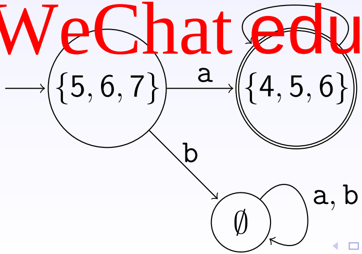
Making the reversed NFA deterministic.
We renamed the states to avoid later confusion;
4 corresponds to $\{2\}$, 5 to $\{1, 2\}$, and 6 to $\{1, 2, 3\}$.



Now reversing the result:



And finally making the result deterministic:



Next Week

We next lo he

regular la

Also, we w t a

language is not regular and introduce c