COSC 341
Theory of Computing
Lecture 5
It's all the same

Stephen Cranefield
stephen.cranefield@otago.ac.nz

Lecture slides (mostly) by Michael Albert

Keywords: NFA, DFA, regular grammar, regular language

Class representative election?

Do we have any volunteers to serve as class representative?

The landscape of languages (so far)

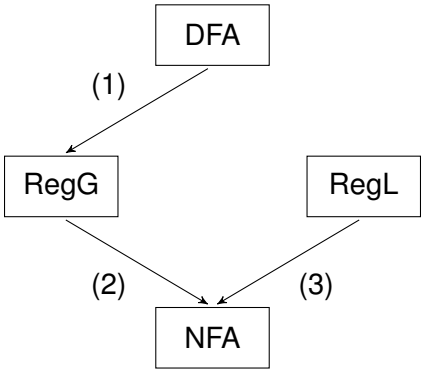
We have defined four ways of generating and/or accepting languages:

- Deterministic finite-state automata
- Regular grammars
- Non-deterministic finite-state automata
- Regular expressions

What relationships do we know about the groups of languages they define?

Known relationships

An arrow $A \to B$ means "every language that can be generated/accepted by A can also be generated/accepted by B"



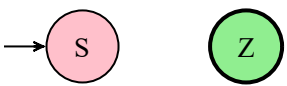
(1): Because we can replace transitions between states by grammar rules.

(2): Because we can replace grammar rules by transitions between states.

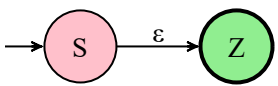
(3): Because the base cases are accepted by NFAs, and the closure properties apply.

Base cases for Regular language $\rightarrow$ NFA

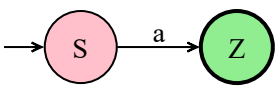
$L = \{\}$



$L = \{\epsilon\}$

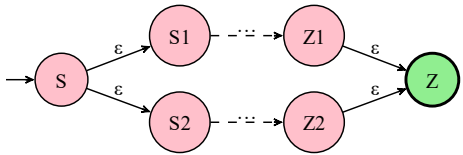


$L = \{a\}$

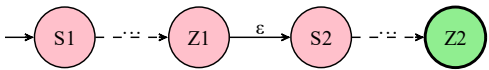


Recursive cases for Regular language $\rightarrow$ NFA

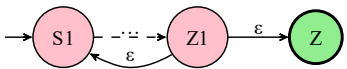
$L_1 \cup L_2$



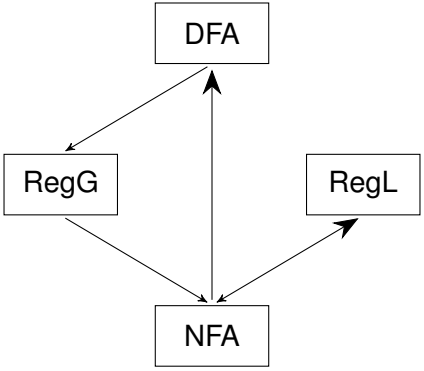
$\text{concat}(L_1, L_2)$



L_1^* (Kleene star)



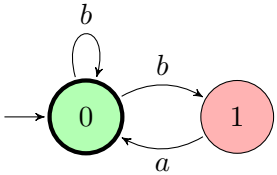
Arrows to add



Two claims

- Every language accepted by an NFA is also accepted by a DFA.
- Every language accepted by an NFA is regular.

From NFA to DFA



We want to think about “all the places we might be” when we begin in some (set of) state(s) and process a letter. We know that we start from state 0.

$$0 \xrightarrow{a} \emptyset$$

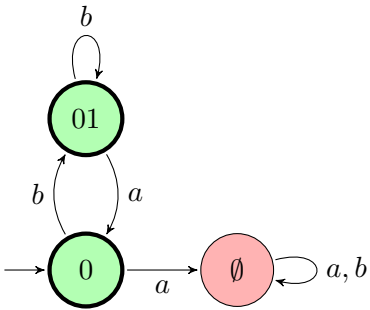
$$0 \xrightarrow{b} 01$$

$$01 \xrightarrow{a} 0$$

$$01 \xrightarrow{b} 01$$

$$\emptyset \xrightarrow{a,b} \emptyset$$

The equivalent DFA

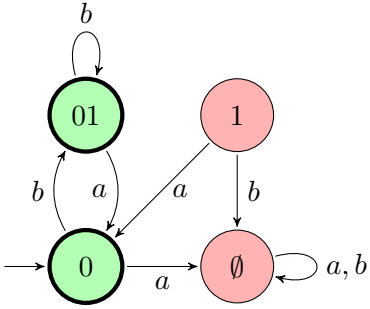


The subset construction algorithm

Let an NFA $\mathbf{A}$ with state set $\mathcal{S}$ and alphabet Σ be given. Define a DFA, $\mathbf{D}$ over the same alphabet as follows:

- The states of $\mathbf{D}$ are precisely the subsets of $\mathcal{S}$ (usually denoted $2^{\mathcal{S}}$, or $\mathcal{P}(\mathcal{S})$).
- The initial state of $\mathbf{D}$ is the ϵ-closure of the initial state of $\mathbf{A}$ (this is the set of all states that can be reached from the initial state of $\mathbf{A}$ using only ϵ-transitions).
- If $X \subseteq \mathcal{S}$ and $a \in \Sigma$, then the a-transition from X in $\mathbf{D}$ is to the set Y which is the ϵ-closure of all the states reachable from any state in X by an a-transition.
- The accepting states of $\mathbf{D}$ are all those subsets of $\mathcal{S}$ that contain at least one accepting state.

Formal version of previous example



We didn't see state 1 when we constructed the previous example because it's unreachable – but it's easier to include states like this in describing the general construction.

From NFAs to regular languages (enriched labels)

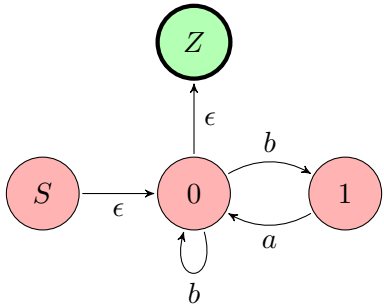
The fundamental idea in showing that the language accepted by any NFA is regular is to extend the state transitions from ϵ or letters of Σ to languages themselves.

For instance, rather than having two edges labelled a and b from one state to another, we could use a single edge labelled $a + b$ to represent the same situation.

Using this idea we can gradually prune away states in an NFA in its initial form until we reach a two-state machine accepting the same language with a regular language on its unique arrow.

Back to our example . . .

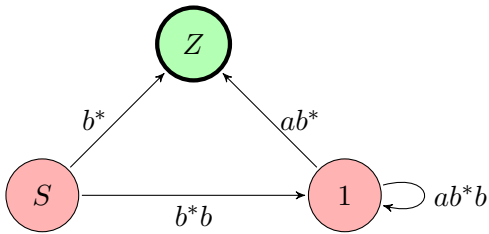
NFA to regular language



It would be easiest to eliminate node 1 first, but to show the general ideas I'll start with node 0.

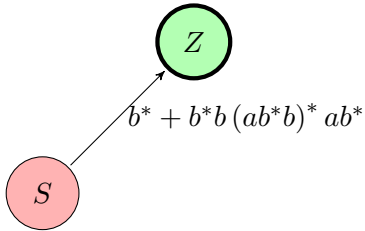
For each path $X \to 0 \to Y$ we enhance (or add) the edge from $X \to Y$ with the concatenation of the label on $X \to 0$, the Kleene-star of the loop-label on 0, and the label from 0 to Y. That captures all the ways we could get from X to Y via 0.

Elimination of state 0



Next we can eliminate state 1.

Elimination of state 1



Now we're done.
Since we've only adjusted labels on edges by adding new stuff that we obtain by concatenating together previous labels and their Kleene-stars, then all the labels must always be regular languages.

If we'd eliminated 1 first the regular language we got would have been:

$$(b + ba)^*.$$

Convince yourself that this is the same thing!