

## Day7

**Review:** The language recognized by the NFA over  $\{a, b\}$  with state diagram



is:

So far, we know:

- The collection of languages that are each recognizable by a DFA is **closed** under complementation.  
*Could we do the same construction with NFA?*

- The collection of languages that are each recognizable by a NFA is **closed** under union.  
*Could we do the same construction with DFA?*

Happily, though, an analogous claim is true!

Suppose  $A_1, A_2$  are languages over an alphabet  $\Sigma$ . **Claim:** if there is a DFA  $M_1$  such that  $L(M_1) = A_1$  and DFA  $M_2$  such that  $L(M_2) = A_2$ , then there is another DFA, let's call it  $M$ , such that  $L(M) = A_1 \cup A_2$ .  
*Theorem 1.25 in Sipser, page 45*

**Proof idea:**

**Formal construction:**

**Example:** When  $A_1 = \{w \mid w \text{ has an } a \text{ and ends in } b\}$  and  $A_2 = \{w \mid w \text{ is of even length}\}$ .



Suppose  $A_1, A_2$  are languages over an alphabet  $\Sigma$ . **Claim:** if there is a DFA  $M_1$  such that  $L(M_1) = A_1$  and DFA  $M_2$  such that  $L(M_2) = A_2$ , then there is another DFA, let's call it  $M$ , such that  $L(M) = A_1 \cap A_2$ .  
*Footnote to Sipser Theorem 1.25, page 46*

**Proof idea:**

**Formal construction:**

## Day8

So far we have that:

- If there is a DFA recognizing a language, there is a DFA recognizing its complement.
- If there are NFA recognizing two languages, there is a NFA recognizing their union.
- If there are DFA recognizing two languages, there is a DFA recognizing their union.
- If there are DFA recognizing two languages, there is a DFA recognizing their intersection.

Our goals for today are (1) prove similar results about other set operations, (2) prove that NFA and DFA are equally expressive, and therefore (3) define an important class of languages.

Suppose  $A_1, A_2$  are languages over an alphabet  $\Sigma$ . **Claim:** if there is a NFA  $N_1$  such that  $L(N_1) = A_1$  and NFA  $N_2$  such that  $L(N_2) = A_2$ , then there is another NFA, let's call it  $N$ , such that  $L(N) = A_1 \circ A_2$ .

**Proof idea:** Allow computation to move between  $N_1$  and  $N_2$  “spontaneously” when reach an accepting state of  $N_1$ , guessing that we've reached the point where the two parts of the string in the set-wise concatenation are glued together.

**Formal construction:** Let  $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$  and  $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$  and assume  $Q_1 \cap Q_2 = \emptyset$ . Construct  $N = (Q, \Sigma, \delta, q_0, F)$  where

- $Q =$
- $q_0 =$
- $F =$
- $\delta : Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)$  is defined by, for  $q \in Q$  and  $a \in \Sigma_\varepsilon$ :

$$\delta((q, a)) = \begin{cases} \delta_1((q, a)) & \text{if } q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1((q, a)) & \text{if } q \in F_1 \text{ and } a \in \Sigma \\ \delta_1((q, a)) \cup \{q_2\} & \text{if } q \in F_1 \text{ and } a = \varepsilon \\ \delta_2((q, a)) & \text{if } q \in Q_2 \end{cases}$$

*Proof of correctness would prove that  $L(N) = A_1 \circ A_2$  by considering an arbitrary string accepted by  $N$ , tracing an accepting computation of  $N$  on it, and using that trace to prove the string can be written as the result of concatenating two strings, the first in  $A_1$  and the second in  $A_2$ ; then, taking an arbitrary string in  $A_1 \circ A_2$  and proving that it is accepted by  $N$ . Details left for extra practice.*

Suppose  $A$  is a language over an alphabet  $\Sigma$ . **Claim:** if there is a NFA  $N$  such that  $L(N) = A$ , then there is another NFA, let's call it  $N'$ , such that  $L(N') = A^*$ .

**Proof idea:** Add a fresh start state, which is an accept state. Add spontaneous moves from each (old) accept state to the old start state.

**Formal construction:** Let  $N = (Q, \Sigma, \delta, q_1, F)$  and assume  $q_0 \notin Q$ . Construct  $N' = (Q', \Sigma, \delta', q_0, F')$  where

- $Q' = Q \cup \{q_0\}$
- $F' = F \cup \{q_0\}$
- $\delta' : Q' \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q')$  is defined by, for  $q \in Q'$  and  $a \in \Sigma_\varepsilon$ :

$$\delta'((q, a)) = \begin{cases} \delta((q, a)) & \text{if } q \in Q \text{ and } q \notin F \\ \delta((q, a)) & \text{if } q \in F \text{ and } a \in \Sigma \\ \delta((q, a)) \cup \{q_1\} & \text{if } q \in F \text{ and } a = \varepsilon \\ \{q_1\} & \text{if } q = q_0 \text{ and } a = \varepsilon \\ \emptyset & \text{if } q = q_0 \text{ and } a \in \Sigma \end{cases}$$

*Proof of correctness would prove that  $L(N') = A^*$  by considering an arbitrary string accepted by  $N'$ , tracing an accepting computation of  $N'$  on it, and using that trace to prove the string can be written as the result of concatenating some number of strings, each of which is in  $A$ ; then, taking an arbitrary string in  $A^*$  and proving that it is accepted by  $N'$ . Details left for extra practice.*

**Application:** A state diagram for a NFA over  $\Sigma = \{a, b\}$  that recognizes  $L((a^*b)^*)$ :

Suppose  $A$  is a language over an alphabet  $\Sigma$ . **Claim:** if there is a NFA  $N$  such that  $L(N) = A$  then there is a DFA  $M$  such that  $L(M) = A$ .

**Proof idea:** States in  $M$  are “macro-states” – collections of states from  $N$  – that represent the set of possible states a computation of  $N$  might be in.

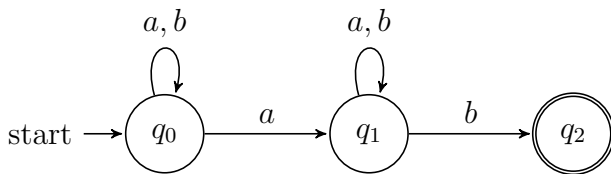
**Formal construction:** Let  $N = (Q, \Sigma, \delta, q_0, F)$ . Define

$$M = ( \mathcal{P}(Q), \Sigma, \delta', q', \{X \subseteq Q \mid X \cap F \neq \emptyset\} )$$

where  $q' = \{q \in Q \mid q = q_0 \text{ or is accessible from } q_0 \text{ by spontaneous moves in } N\}$  and

$\delta'((X, x)) = \{q \in Q \mid q \in \delta(r, x) \text{ for some } r \in X \text{ or is accessible from such an } r \text{ by spontaneous moves in } N\}$

Consider the state diagram of an NFA over  $\{a, b\}$ . Use the “macro-state” construction to find an equivalent DFA.



Consider the state diagram of an NFA over  $\{0, 1\}$ . Use the “macro-state” construction to find an equivalent DFA.



Note: We can often prune the DFAs that result from the “macro-state” constructions to get an equivalent DFA with fewer states (e.g. only the “macro-states” reachable from the start state).

## The class of regular languages

Fix an alphabet  $\Sigma$ . For each language  $L$  over  $\Sigma$ :

**There is a DFA over  $\Sigma$  that recognizes  $L$**   $\exists M$  ( $M$  is a DFA and  $L(M) = A$ )  
*if and only if*

**There is a NFA over  $\Sigma$  that recognizes  $L$**   $\exists N$  ( $N$  is a NFA and  $L(N) = A$ )  
*if and only if*

**There is a regular expression over  $\Sigma$  that describes  $L$**   $\exists R$  ( $R$  is a regular expression and  $L(R) = A$ )

A language is called **regular** when any (hence all) of the above three conditions are met.

We already proved that DFAs and NFAs are equally expressive. It remains to prove that regular expressions are too.

Part 1: Suppose  $A$  is a language over an alphabet  $\Sigma$ . If there is a regular expression  $R$  such that  $L(R) = A$ , then there is a NFA, let's call it  $N$ , such that  $L(N) = A$ .

**Structural induction:** Regular expression is built from basis regular expressions using inductive steps (union, concatenation, Kleene star symbols). Use constructions to mirror these in NFAs.

**Application:** A state diagram for a NFA over  $\{a, b\}$  that recognizes  $L(a^*(ab)^*)$ :

Part 2: Suppose  $A$  is a language over an alphabet  $\Sigma$ . If there is a DFA  $M$  such that  $L(M) = A$ , then there is a regular expression, let's call it  $R$ , such that  $L(R) = A$ .

**Proof idea:** Trace all possible paths from start state to accept state. Express labels of these paths as regular expressions, and union them all.

1. Add new start state with  $\varepsilon$  arrow to old start state.
2. Add new accept state with  $\varepsilon$  arrow from old accept states. Make old accept states non-accept.
3. Remove one (of the old) states at a time: modify regular expressions on arrows that went through removed state to restore language recognized by machine.

**Application:** Find a regular expression describing the language recognized by the DFA with state diagram

