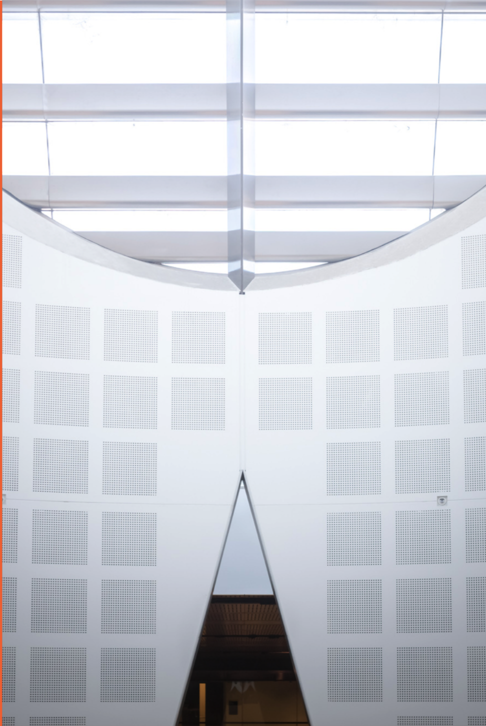
COMP2022|2922 Models of Computation

NFAs, DFAs, REs have the same expressive power

Sasha Rubin

August 10, 2022



Where are we going?

- We are going to show that DFA, NFA and regular expressions specify the same set of languages!
- We will do this with a series of transformations:

1. From Regular Expressions to NFAs
2. From NFAs to NFAs without ϵ-transitions
3. From NFAs without ϵ-transitions to DFAs
4. From DFAs to Regular Expressions

This is also covered in Sipser Chapter 1 (although he combines steps 2 and 3).

2. NFAs to NFAs without ϵ-transitions

Theorem
For every NFA N there is an NFA N' without ϵ-transitions such that $L(N) = L(N')$.

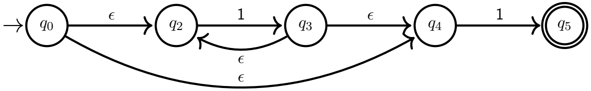
Idea: "N' skips over ϵ-transitions of N"

- So N' is like N but we remove all the ϵ-transitions and add new transitions and new final states.
- Write $q \rightsquigarrow r$ to mean that N can go from q to r using zero or more ϵ-transitions.
- When is q a final state of N'?
  - if $q \rightsquigarrow r$ and r is a final state of N.
- When is $q \xrightarrow{a} q'$ a transition of N'?
  - if $q \rightsquigarrow r$ and $r \xrightarrow{a} q'$.

How to compute if $q \rightsquigarrow r$? See Tutorial where we write $r \in silentmoves(q)$.

2. NFAs to NFAs without ϵ-transitions

Example



3. NFAs without ϵ-transitions to DFAs

Theorem
For every NFA N without ϵ-transitions there is a DFA M such that $L(M) = L(N)$.

Idea: "subset construction"

Q: How would you simulate the NFA if you were pretending to be a DFA?

A: Keep track of the set of possible states that the NFA is in.

Q: And when would you accept?

A: If at least one of the states I'm keeping track of is an accepting state of the NFA.

3. NFAs without ϵ-transitions to DFAs

Theorem

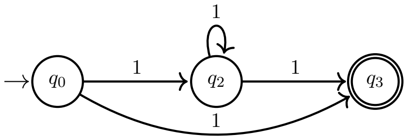
For every NFA N without ϵ-transitions there is a DFA M such that $L(M) = L(N)$.

Construction

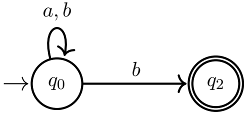
Given NFA $N = (Q, \Sigma, \delta, q_0, F)$ without ϵ-transitions the subset construction builds a DFA M:

- every set $X \subseteq Q$ is a state of M,
- the alphabet is Σ,
- for a state $X \subseteq Q$ of M, define $\delta'(X, a) = \bigcup_{q \in X} \delta(q, a)$,
- the initial state of M is the set $\{q_0\}$,
- for a state $X \subseteq Q$ of M, put X into F' if X contains an accepting state of N.

3. NFAs without ϵ-transitions to DFAs



3. NFAs without ϵ-transitions to DFAs



4. DFAs to Regular Expressions

Theorem
For every DFA M there is a regular expression R such that $L(M) = L(R)$.

Idea:

1. We convert M into a generalised nondeterministic finite automaton (GNFA) which is like an NFA except that it can have regular expressions label the transitions.
2. We successively remove states from the automaton, until there is just one transition (from the start state to the final state).
3. We return the regular expression on this last transition.

GNFAs

- A run (aka computation) of a GNFA N on string w is a sequence of transitions

$$q_0 \xrightarrow{R_1} q_1 \xrightarrow{R_2} q_2 \xrightarrow{R_3} \dots \xrightarrow{R_m} q_m$$

  in N such that w matches the regular expression $R_1 R_2 \cdots R_m$.
- The run is accepting if $q_m \in F$.
- The language recognised by N is $L(N) = \{w \in \Sigma^* : w \text{ is accepted by } N\}$.

From DFAs to Regular Expressions

Generalised NFAs. We make sure that:

1. the initial state has no incoming edges.
2. there is one final state, and it has no outgoing edges.
3. there are edges from every state (that is not final) to every state (that is not initial).

We show how to translate DFAs into GNFAs, and then successively remove states from the GNFA until there is just a single edge left.

From DFAs to GNFAs

For every DFA M there is a GNFA N such that $L(M) = L(N)$.

1. Add an initial state and ϵ-transition to the old intial state.
2. Add a final state and ϵ-transitions to it from all the old final states.
3. Add transitions for every pair of states, including from a state to itself, (except leaving the final, or entering the new).

From DFAs to GNFAs

$$\begin{tikzcd}
{} \arrow[r] & q_1 \arrow[loop above, "a"] \arrow[r, "b"] & \boxed{q_2} \arrow[loop above, "{a, b}"]
\end{tikzcd}$$

$$\begin{tikzcd}
{} \arrow[r] & q_0 \arrow[r, "\epsilon"] & q_1 \arrow[loop above, "a"] \arrow[r, "b"] & q_2 \arrow[loop above, "a \mid b"] \arrow[r, "\epsilon"] & q_3
\end{tikzcd}$$

Removing states from GNFAs

For every GNFA N with > 2 states there is a GNFA N' with one less state such that $L(N) = L(N')$.

1. Pick a state q to eliminate.
2. For every pair of remaining states (s, t), replace the label from s to t by the regular expression

$$R_{s,t} \,|(R_{s,q} R^*_{q,q} R_{q,t})$$

where $R_{x,y}$ is the regular expression labeling the transition from state x to state y.

Removing states from GNFAs

$$\begin{tikzcd}
\arrow[r] & q_0 \arrow[r, "\epsilon"] & q_1 \arrow[r, "b"] \arrow[loop above, "a"] & q_2 \arrow[r, "\epsilon"] \arrow[loop above, "a \mid b"] & q_3
\end{tikzcd}$$

Summary

1. A language is regular if it is recognised by some DFA.
2. The following models of computation describe the same languages: DFAs, NFAs, Regular Expressions.
3. The regular languages are closed under the Boolean operations union, intersection, complementation, as well as concatenation and Kleene star.

Important decisions problems about DFAs

There are natural decision problems associated with DFAs. Are there programs that solve them?

1. DFA Membership problem

   Input: DFA M, string w.
   Output: decide if $w \in L(M)$.

2. DFA Non-emptiness problem (tutorial)

   Input: DFA M.
   Output: decide if $L(M) \neq \emptyset$.

3. DFA Equivalence problem (tutorial)

   Input: DFAs M_1, M_2.
   Output: decide if $L(M_1) = L(M_2)$.

Important decisions problems about REs

Using the solutions to the DFA problems, we can solve problems about REs!

1. RE Membership problem

   Input: RE R, string w.
   Output: decide if $w \in L(R)$.
   - Idea: Convert R to DFA M and check if $w \in L(M)$.
2. RE Non-emptiness problem

   Input: RE R.
   Output: decide if $L(R) \neq \emptyset$.
   - Idea: Convert R to DFA M and check if $L(M) \neq \emptyset$.
3. RE Equivalence problem

   Input: REs R_1, R_2.
   Output: decide if $L(R_1) = L(R_2)$.
   - Idea: Convert R_i to DFA M_i and check if $L(M_1) = L(M_2)$.