DFAs and NFAs

Max Randall
Chapman University

March 2, 2025

Contents

1 Introduction

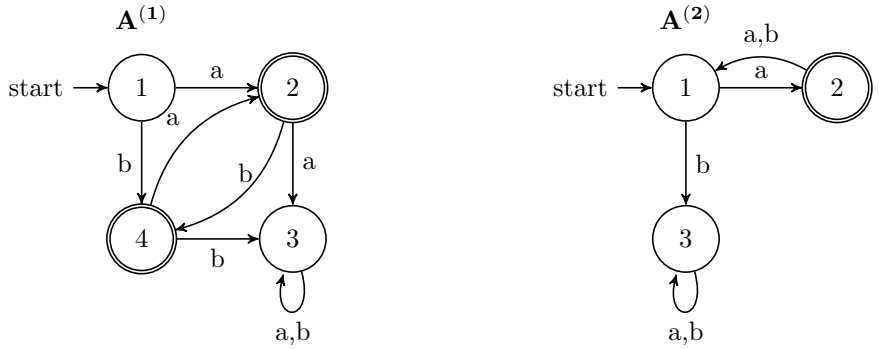
This week we covered ways to combine automata. We focused on the set theory used in the combination as well as some notation. Lastly, we introduced Nondeterministic Finite Automatas, or NFAs.

2 Exersizes

2.1 Extended Transition Functions

Brief summary of the question: Given the two DFAs below:

- Compute the extended transition functions $\hat{\delta}^{(1)}(1,\, abaa)$ and $\hat{\delta}^{(2)}(1,\, abba)$, showing all steps.
- Describe the language accepted by each automaton.



1. Languages Accepted by Each Automaton

$$L\big(A^{(1)}\big) \;=\; \big\{\, w \in \{a,b\}^{+} \;\mid\; \text{no two consecutive symbols in } w \text{ are the same} \big\}.$$

$L\big(A^{(2)}\big) \;=\; a\,(\,(\,a \mid b\,)\,a\,)^{*} \;=\; \big\{ w \in \{a,b\}^{*} \;\mid\; w \text{ has odd length, starts with } a, \text{ and every odd position is } a \big\}.$

2. Extended Transition Functions

$$\hat{\delta}^{(1)}\big(1,\; abaa\big) \;=\; 3$$
$$\hat{\delta}^{(2)}\big(1,\; abba\big) \;=\; 3$$

2.2 Intersection Automaton A for $\mathbf{A^{(1)}}$ and $\mathbf{A^{(2)}}$

We form the product (a.k.a. intersection) automaton

$$A \;=\; \big(Q^{(1)} \times Q^{(2)},\; \Sigma,\; \delta,\; (q_0^{(1)}, q_0^{(2)}),\; F^{(1)} \times F^{(2)}\big),$$

where

- $Q^{(1)} = \{1,2,3,4\}$ with start state $q_0^{(1)} = 1$ and final states $F^{(1)} = \{2,4\}$,
- $Q^{(2)} = \{1,2,3\}$ with start state $q_0^{(2)} = 1$ and final states $F^{(2)} = \{2\}$,
- $\Sigma = \{a,b\}$,
- the transition function δ on a pair (p,q) and symbol $x \in \{a,b\}$ is

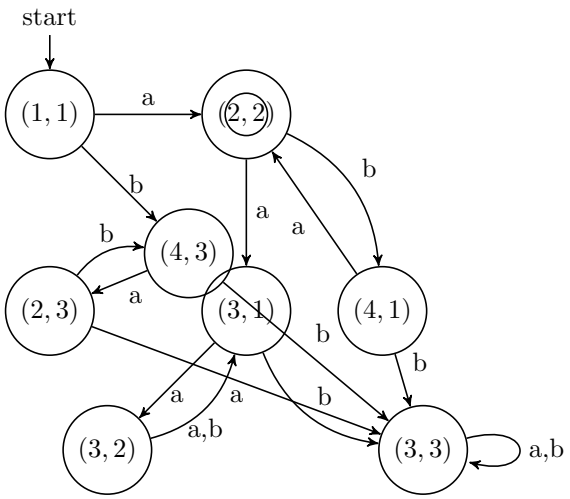
$$\delta\big((p,q),\, x\big) \;=\; \big(\delta^{(1)}(p,x),\; \delta^{(2)}(q,x)\big).$$

- the final (accepting) states are all pairs $(p,q) \in Q^{(1)} \times Q^{(2)}$ with $p \in F^{(1)}$ *and* $q \in F^{(2)}$.

Transition Diagram (Reachable Part)

Intersection Automaton Construction.
Construct the intersection automaton A for $A^{(1)}$ and $A^{(2)}$ as given above by drawing its transition diagram. (If there are unreachable states, you do not have to draw them.)

Below is the transition diagram of A, including only the reachable states from $\big(1,1\big)$. Recall that the *only* accepting pairs are those whose first coordinate is in $\{2,4\}$ and second coordinate is 2. (Here we will see that $(4,2)$ is actually *unreachable*, so $(2,2)$ is the only reachable accepting state.)



Why $L(A) = L(A^{(1)}) \cap L(A^{(2)})$?

By construction, A simulates both $A^{(1)}$ and $A^{(2)}$ in parallel:

- Starting in $\big(q_0^{(1)}, q_0^{(2)}\big)$, reading each symbol $x \in \{a,b\}$ goes to $\big(\delta^{(1)}(p,x),\, \delta^{(2)}(q,x)\big)$.
- A string w is accepted by A if and only if, after reading w, the first coordinate is in $F^{(1)}$ *and* the second coordinate is in $F^{(2)}$.

Hence a string is accepted precisely when *both* $A^{(1)}$ and $A^{(2)}$ accept it. Thus $L(A) = L(A^{(1)}) \cap L(A^{(2)})$.

Changing A to Obtain Union

If we want *union* rather than intersection, we can keep the same set of states, start state, and transitions, but change the accepting condition so that a pair (p,q) is final whenever *either* $p \in F^{(1)}$ **or** $q \in F^{(2)}$. Concretely,

$$F' \;=\; \big(F^{(1)} \times Q^{(2)}\big) \;\cup\; \big(Q^{(1)} \times F^{(2)}\big).$$

Then the resulting DFA $A' = (Q, \Sigma, \delta, (q_0^{(1)}, q_0^{(2)}), F')$ accepts

$$L(A') = L(A^{(1)}) \;\cup\; L(A^{(2)}).$$

2.3 Exercise 2.2.7

Claim. Let A be a DFA, and let q be a particular state of A such that $\delta(q, a) = q$ for *every* input symbol a in the alphabet. Then for any input string w, we have $\delta(q, w) = q$.

Proof (by induction on the length of w):

- **Base Case:** If $|w| = 0$, then $w = \varepsilon$. By definition of the extended transition function, $\delta(q, \varepsilon) = q$. Hence the statement holds for all strings of length 0.
- **Inductive Step:** Suppose the statement holds for all strings of length n. Let w be any string of length $n + 1$. We can write $w = x\,a$, where $|x| = n$ and a is a single input symbol. By the definition of the extended transition function,

  $$\delta(q, w) \;=\; \delta\big(q,\; x\,a\big) \;=\; \delta\big(\delta(q, x),\; a\big).$$

  By the inductive hypothesis, $\delta(q, x) = q$. Therefore

  $$\delta\big(\delta(q, x),\; a\big) \;=\; \delta(q, a) \;=\; q,$$

  because $\delta(q, a) = q$ for every input symbol a by assumption.

Thus, by mathematical induction, $\delta(q, w) = q$ for every string $w \in \Sigma^*$. $\square$

3 Chapter 2.3

The text introduces the *extended transition function* $\hat{\delta}$ for an NFA. Recall that δ itself only covers a single state and input symbol, while $\hat{\delta}$ handles a state (or set of states) and an entire string. We define $\hat{\delta}$ by the rules:

- $\hat{\delta}(q, \varepsilon) = \{\, q\}$. (If we consume no input, we remain in the same state.)
- $\hat{\delta}(q,\, xa) \;=\; \bigcup_{p \,\in\, \hat{\delta}(q,\, x)} \delta(p,\, a)$. (If $w = xa$, first move on x, then from each possible state, move on symbol a.)

This captures **nondeterminism** because we allow multiple possible next states. An NFA *accepts* a string w if at least one of the states in $\hat{\delta}(q_0, w)$ is in the set of accepting states. Several examples illustrate the step-by-step use of $\hat{\delta}$, showing that an NFA may branch along several paths of computation.

Subset Construction. We can convert any NFA N into a DFA D recognizing the same language by the *subset construction*, in which each subset of N's states becomes a single state in D. Concretely:

- The states of D are all subsets of N's state set.
- D's start state is the subset $\{\, q_0\}$ containing only the NFA's start state.
- A subset S of N's states is *accepting* in D if S contains at least one of N's accepting states.
- For each subset $S \subseteq Q$ and each symbol $a \in \Sigma$,

  $$\delta_D(S, a) \;=\; \bigcup_{p \in S} \delta_N(p, a).$$

This process ensures $L(D) = L(N)$. Although the subset construction can produce a worst-case exponential blowup in the number of states, in practice it often yields fewer states than the worst-case bound. Thus, we see every language recognized by an NFA also has a DFA recognizer, proving the equivalence of NFAs and DFAs.

4 Conclusion

Throughout these problems and readings, we have deepened our understanding of both deterministic and nondeterministic finite automata. We explored extended transition functions, which formalize how automata process entire strings. We examined the product automaton construction, which combines two DFAs for intersection. We also saw that modifying acceptance states can yield union. Through the subset construction, we established the equivalence of NFAs and DFAs: any NFA can be systematically converted into a DFA that accepts the same language. Consequently, these powerful concepts and proofs underscore the robustness of finite automata theory, vital across both theory and practice, and widely used today.

Interesting question: Maximal Blow-Up in the Subset Construction.
Can you describe a concrete NFA of n states that forces its equivalent DFA (from the subset construction) to use all 2^n possible subsets as distinct states? Moreover, how can we prove that no smaller DFA can recognize the same language?

References

[HMU] . E. Hopcroft, R. Motwani, J. D. Ullman: *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*, Archive.org Link.