COMPSCI 250 Discussion #9: Minimizing a DFA
Individual Handout

David Mix Barrington and Mordecai Golin

3 May 2024

In Lecture 32 we presented the Myhill-Nerode Theorem, which says that every recognizable language has a **minimal DFA** that can be defined in terms of the equivalence classes of the $\equiv_L$ relation on strings. This result leads to an algorithm that takes *any* DFA as input and returns the minimal DFA for the same language. This is called **minimizing a DFA**.

The algorithm proceeds by defining a series of equivalence relations on the *states* of the original DFA. Relation 0 has two classes, one for the final and one for the non-final states. To get relation $i+1$ from relation i, we proceed as follows. If a class of relation i has only one element, it must remain a class in relation $i+1$ and we need do nothing. For all other classes, we take each state in that class and note *which class* each letter takes it to. This sequence of classes (pair of classes, when we have a two-letter alphabet) is called the **behavior** of the state with respect to relation i.

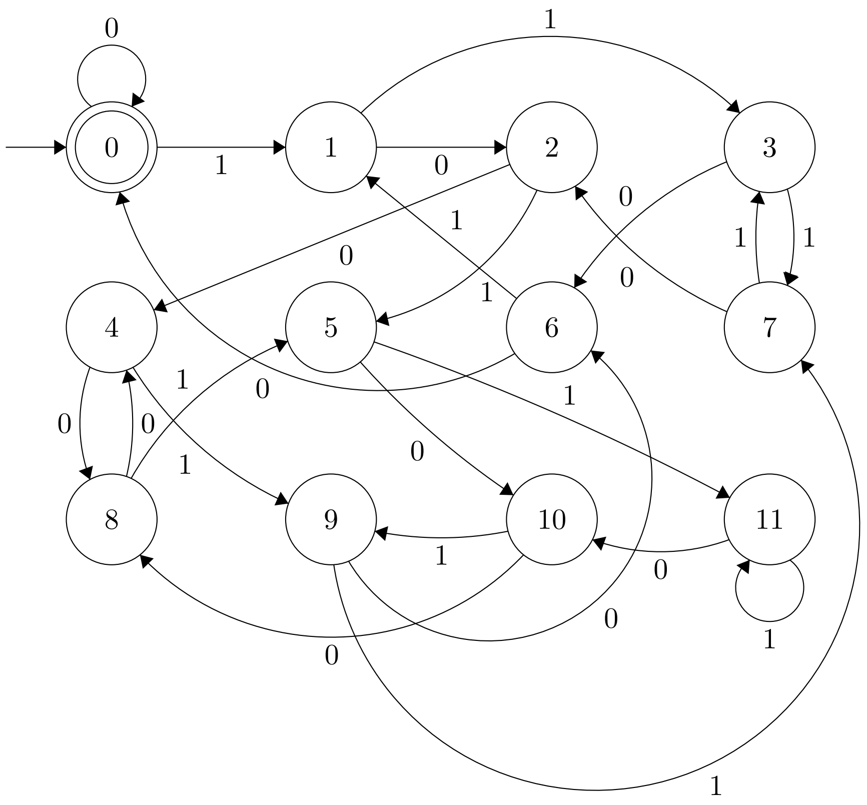
To form relation $i+1$, we divide each class of relation i into the sets of states that share a common behavior. If all the states in the class have the *same* behavior, the class does not change. (Though it could change later if some of the classes to which these states go are divided into smaller classes.) If all the classes stay the same from relation i to relation $i+1$, we are done. We may form a DFA whose states are the classes of relation i and whose arrows indicate which classes each letter takes each class.

We'll do an example on the board today for the language No$-aba$. There is a natural eight-state DFA for this language, with classes for the strings λ, a, b, aa, ab, ba, bb, and aba. The DFA goes to state aba if it has seen an aba – otherwise it goes to the state for the last two letters it has seen, or the only zero or one letter it has seen. Our relation 0 will have two classes F and N, of sizes 7 and 1 respectively, for the sets of final and of non-final states. Relation 1 will have three classes, of sizes 6, 1, and 1. Relation 2 will have four classes, of sizes 3, 3, 1, and 1, and this will turn out to be the final relation. So this language has a four-state DFA (which is the same as the one we presented in lecture this week).

(more)

Writing Exercise: Carry out the algorithm presented to get the minimal DFA equivalent to the twelve-state DFA drawn below. This DFA has alphabet $\{0, 1\}$ and determines whether a string represents a number in binary that is divisible by twelve. The state set is $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$. On input 0, the machine goes from state i to state $2i\%12$. On input 1, the machine goes from state i to state $(2i + 1)\%12$.

Give each of the equivalence relations produced by the algorithm and demonstrate that your last one is really the last one. Draw the resulting minimal DFA.


0
1
2
3
4
5
6
7
8
9
10
11
0
1