Algorithms on Finite Automata and Regular Languages

Prakash Panangaden

26th September 2021

One is often called upon to write algorithms to determine properties of regular languages or finite automata. In this note I will sketch the basic ideas for such algorithms. The algorithmic descriptions will be high-level and will use basic algorithms taught in class as building blocks. When you write up solutions to questions on the homework **do not** give details of algorithms done in class. It is **not** a virtue to give details when you are being asked for a high-level description. Needless to say writing code in some programming language is **completely wrong**; I will give you zero without bothering to read your code.

Here are the basic building blocks:

- From a regular expression one can construct an NFA, perhaps with ε-moves.
- An NFA with ε-moves can be converted into an ordinary NFA.
- From an NFA, one can construct an equivalent DFA: this is called *determinization*.
- From a DFA, one can construct a *minimal* DFA, this is called *minimization*.
- Given a DFA, standard graph algorithms[1] can tell if a state is reachable from the start state.

Here is a basic question I could ask: Given an NFA design an algorithm to decide whether there is any word accepted by it? Answer: Take the NFA as given and run a reachability algorithm from each start state to see if any

[1] Which you should have learned in COMP 251

accept state is reachable. If there is a reachable accept state then there is some word that must be accepted by the NFA otherwise not. End of answer.

You see how short the answer is. You see that I don't care what reachabilty algorithm you used. And you see that I am not interested in the details of the reachability algorithm.

Here is another question: Given a DFA does it accept an infinte language? Answer: Run a graph algorithm for finding cycles[2] in the transition graph. If there is no cycle then the DFA definitely does **not** accept an infinite language. If there is a cycle, run a reachability algorithm to see if any of the states in the cycle are reachable from the start state, then run a reachability algorithm to see if any of the states in the cycles can reach an accept state. If any of these reachability tests fail then the language of the DFA is finite; if all the reachability tests succeed then the language is certainly infinite.

Finally, do not assume that you can do things like "test every word" in an algorithm. For example, I have, in the past asked questions like: devise an algorithm to decide if two DFA's accept the same language. A completly wrong answer is: "check every word in Σ^* to see if it is accepted or rejected by both automata." Can you really check every word when there are infinitely many words. If you write any such thing I will give you zero; it means you do not even understand what an algorithm is.

[2]Something you would have learned in COMP 251