Subset construction

Alexis Gadonneix (ag4625)
Nikhil Mehta (nm3077)

I - Problem Statement

For our final project we seek to parallelize the subset construction algorithm that is applied to a nondeterministic finite automaton (NFA) to convert it into a deterministic finite automaton (DFA). This is particularly useful in the context of string processing and compilers where one wants to check whether a given string matches a regular expression or not. Since regexes can be represented as NFAs, this is the same as asking the question of whether a given string will be accepted by an NFA. At the surface this might seem like a tricky problem since NFA's are, as their name suggests, non-deterministic. However, NFA's can always be converted into DFAs through the subset construction algorithm. This algorithm, described below, is one that has to consider a variety of possible intermediary states in the NFA at the same time. Thus, the algorithm is ripe for parallelization. As a stretch goal, we will implement non-parallelized functionality to convert a regex to an NFA using Thompson's algorithm.

II - Algorithm

The key idea of the algorithm is to focus on the set of states of the NFA that can be reached after seeing a given string (it would be a singleton for a DFA). The subset construction algorithm generates all those sets of states (those are the states of the new DFA) and connects them to each other (if I go from state A to state B by following the edges of the NFA labeled with a given character, I can add an edge with the same label between A and B in my DFA).
We also want to handle ε-moves since they appear in NFAs constructed from regular expressions. To deal with those, we have to apply an ε-closure to each set of states, i.e. we have to add all the states that can be reached through edges labeled with ε.

The pseudo-code for the algorithm:

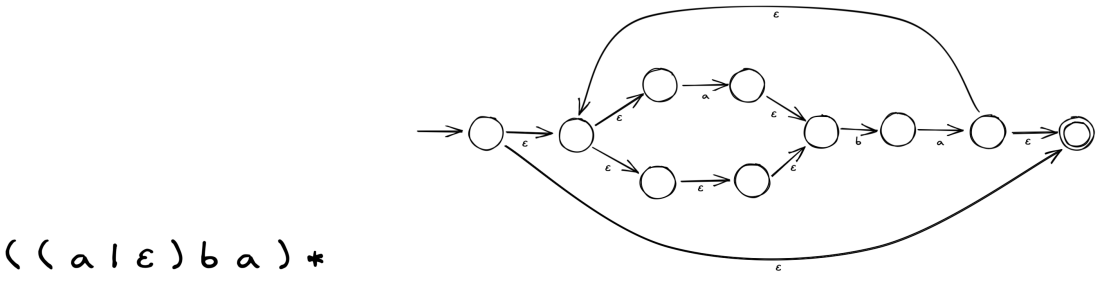
```
Init: Add ε-closure of the start states to DStates

While there is a non-visited state T in DStates do
      mark T as visited
      for each symbol c do
            compute S the set of states that can be reached from T through c
            compute SE the ε-closure of S
            if SE is not in DStates do
                  add SE to DStates as a non-visited state
                  mark SE as final if it contains a final state of the NFA
            end
            Transition[T, c] = SE
      end
end
```

The algorithm has a worst-case complexity of $O(2^n)$ since the number of subsets of an n-state NFA is 2^n. This worst-case complexity can be achieved with fairly simple languages and this will be useful to test our code against difficult instances.

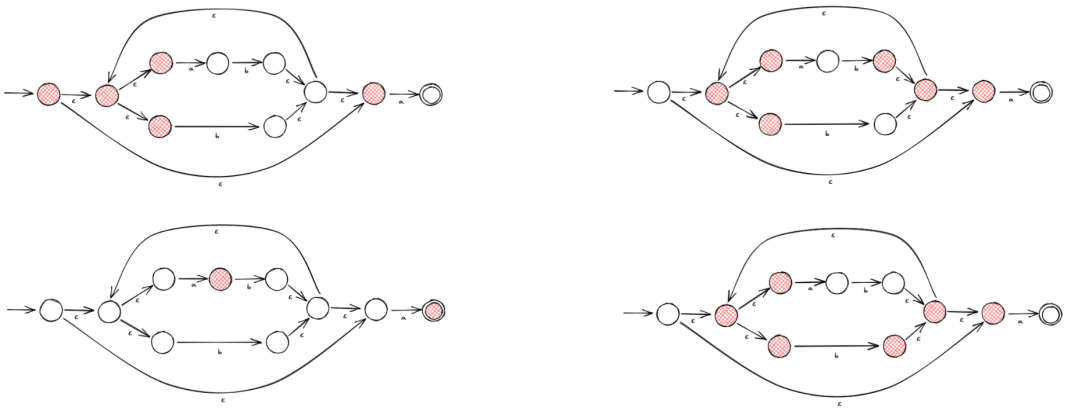
We want to parallelize the computation of all the subsets, but we will have to be careful and keep track of the states that have already been computed to avoid doing the same computations multiple times.
In order to easily build NFAs with various sizes and structures, we will also be implementing a sequential version of Thompson's construction algorithm.

III - Example

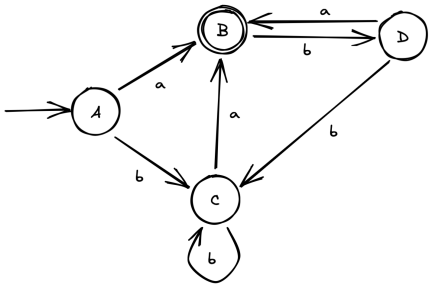
Given the regex ((a | ε) b a) * write the corresponding DFA. First, we convert the regex into an NFA using Thompson's algorithm. Then we perform



Given Regex *NFA*



Subset construction algorithm applied to NFA



Resultant DFA

IV - References

- *Programming Languages and Translators* class at Columbia. Pr. Ronghui Gu, Pr. Stephen Edwards
- M. O. Rabin and D. Scott, *Finite automata and their decision problems*, IBM J.Res. and Develop., vol.3
- Aho Alfred V., Lam Monica S., Sethi Ravi, and Ullman Jeffrey D. 2006. *Compilers: Principles, Techniques, and Tools (2nd Edition)*. Addison-Wesley Longman Publishing Co., Inc.