Simplifying NFAs

Proof Paper Example

James Logan Mayfield

January 18, 2017

1 Problem

In this paper we consider the problem of converting an NFA with more than one accept state to an equivalent NFA with a single accept state. For example, consider the NFA shown in Figure 1.

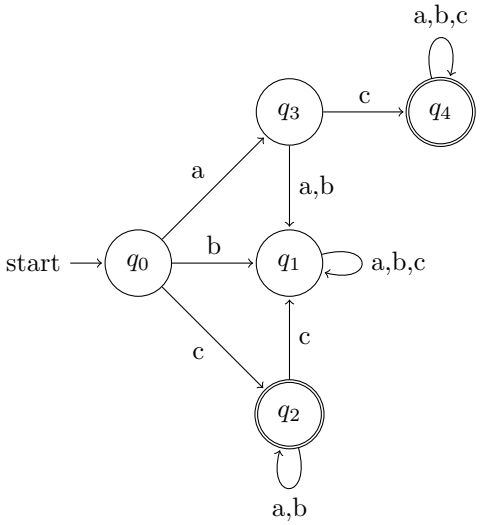

Figure 1: An NFA with Many Accept states

We might wish to unify the two accept states in this NFA in order to have a simpler machine with a single accept state and single starting state. The NFA shown in Figure 2 is such a machine and accepts the same language as that of the machine in Figure 1. The new machine makes use of ε transitions to move from old accept states to a new accept state.

The process used to generate the NFA shown in Figure 2 can be generalized for any NFA and this generalization leads to the following theorem.

Theorem 1. *For any NFA N with more than one accept state there exists an equivalent NFA N' with only one accept state.*

We now prove Theorem 1 by providing a general process for construction N' and proving the equivalence of the language of N and $N\prime$.

2 Proof Sketch

In this paper we prove Theorem (1) by giving a construction for an equivalent NFA with a single accept state. The crux of the construction is the addition of ε transitions from the original accept states to a new,

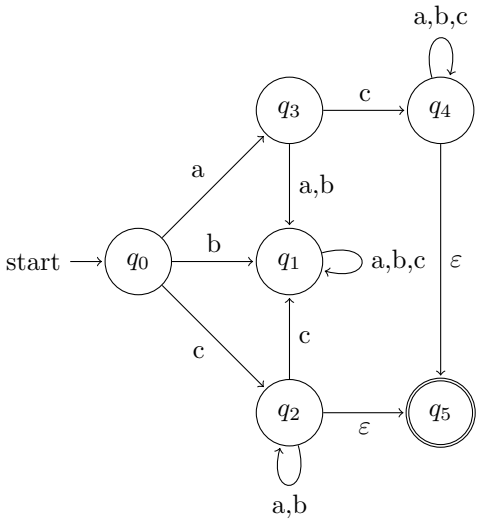

Figure 2: A Single Accept state version of the NFA shown in Figure 1

singular accept state. Any string in the language of the original machine will terminate in an old accept state. The new accept state is then reached by taking the ε transition to the new accept state. Strings not in the language that find themselves in the new accept state can be rejected by adding a transition out of the accept state to a garbage state that must be taken for any letter of the alphabet or by leaving transitions out of the new accept state undefined and thereby implying the rejection of any non-empty string in the accept state.

3 Proof

The proof of Theorem 1 takes place in two parts. The first part is a procedure for constructing a NFA with a single accept state given an arbitrary NFA N. The construction is done in terms of the formal definition of a NFA and basic set theory. It requires no deeper consideration. The second part of the proof shows that the machine constructed by this procedure is equivalent to the given machine N. The proof of machine equivalence is carried out by direct proof. It shows that the constructed machine N' accepts all strings accepted by N and rejects exactly the same set of strings as that of N. The direct proofs only require the formal definition of computation done by a NFA and basic set theory. They too require no deeper consideration.

Proof. We proceed by describing a process for the construction an NFA with a single accept state given an arbitrary NFA N.

1. **The Construction** Let $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$ be an NFA recognizing some language A. We will construct a NFA $N' = (Q_{N'}, \Sigma, \delta_{N'}, q_{N'}, F_{N'})$ from N as follows:

   i. $Q_{N'} = Q_N \cup \{q_f\}$

   ii. $q_{N'} = q_N$

   iii. $F_{N'} = \{q_f\}$

   iv. $\delta_{N'}$ is given as follows:

$$\delta_{N'}(q,a) = \begin{cases} \delta_N(q,a) & q \in Q_N, a \in \Sigma \\ q_f & q \in F_N, a = \varepsilon \\ \emptyset & q = q_f, a \in \Sigma \end{cases}$$

The NFA N' clearly satisfies the requirement of having a single accept state. The alphabet and start state remain unchanged from that of N. The new accept state is added to the set of states and old accept state are not longer part of the set of accept states. Transitions to and from the new state are defined such that any string in A can end in an old accept state then transition to the new accept state via ε transition. Strings not in A are still rejected by virtue of not ending in an accept state or getting stuck in the new accept state with characters remaining and no defined transitions for those characters.

2. *Equivalence of the constructed NFA*

   We now show that an NFA N' constructed from NFA N as shown above also accepts the language $L(N) = A$. We do this by first showing that N' will accept all of the strings in A. We then show that any string not in A will be rejected by N'.

   i. If $w = y_1 y_2 \dots y_m$ is in the language A, then there exists a sequence of states $r_0, r_1, \dots, r_m$ such that $r_0 = q_N$, $r_m \in F_N$ and $r_{i+1} \in \delta_N(r_i, y_i)$, for $i = 0, \dots, m-1$. According to the construction of N' this sequence may also be visited when w is fed to N' as all the states Q and transition function δ from N are preserved by N'. Thus, the NFA N' can accept string w with the following sequence: $r_0, r_1, \dots, r_m, q_f$ because the transition $\delta_{N'}(r_m, \varepsilon) = q_f$ is available once r_m is reached.

   ii. If $w = y_1 y_2 \dots y_m$ is not in the language A, then there does not exist a sequence of states $r_0, r_1, \dots, r_m$ such that $r_0 = q_N$, $r_m \in F_N$ and $r_{i+1} \in \delta_N(r_i, y_i)$, for $i = 0, \dots, m-1$. More specifically, any reading of w will terminate in a state $r_m \notin F_N$. The addition of q_f and the ε transitions leading from states in F_N to q_f means that any w that reaches $r_i \in F_N$ on step $i < m$ may transition to the accept state q_f. However, there are no further transitions from q_f available for the remaining $m - i$ characters in w and N' therefore rejects w in this case. This means the only other readings available for w are the same as the readings performed by N, which clearly do not end at q_f as it was not in the original NFA N.

   Given that N' accepts strings from language A and rejects all other strings, it is clear that the language of N' is exactly that of N and that the two machines are equivalent.

□

4 Conclusion

Given the proof of Theorem 1, we can now assume that any NFA, other than those that accept an empty language and have no accept states, has one and only one accept state. This can be useful in constructing complex machines by concatenating smaller, simpler machines. It also provides a kind of simplicity that is nice to have at times. Furthermore, we can extend this result using Theorem 1.39 from Sipser to show that every DFA has an equivalent NFA with a single accept state[1]. It is not clear, however, if we can go as far as to say that every NFA and DFA has an equivalent DFA with only a single accept state as the construction used here requires ε transitions that are not available to a DFA.

It is also worth pointing out that due to the nature of regular languages, finite automaton, and the construction given here, we were able to attack the NFA equivalence proof by showing that the set of rejected strings for both machines is the same. Alternatively, we could have tried to show that not only does the constructed machine N' accept all the strings accepted by N, but that the reverse is true as well. This is a more standard approach to showing two sets, in this case languages, are equivalent and doesn't require dealing with the complement of the set in question, in this case strings not in the language.

References

[1] Michael Sipser. *Introduction to the Theory of Computation.* Cengage Learning, 3rd edition, 2013.