Automaton minimization

June Pecherskaya Artem Shinkarov

March 5, 2012

A set of words that are recognized by a certain regular expression are also recognized by a group of automata. These automata might have different state names and even different number of states. It is natural that in a parser we would want to use the automaton with the smallest possible amount of states to minimize the storage required for it.

As the naming of the states is unimportant, we will say that two automata are the same up to state names if one can be transformed to another be simply renaming the states.It turns out that for each regular language exists a unique (up to state names) automaton with a minimal number of states[1].

Before being able to continue to the actual minimization of automata, a definitions is required. We will say that string x distinguishes state s from state t if only one state reached from t and s by following x is an accepting state. So two states are distinguishable if there exists a string that distiguishes them. Any accepting state is distinguishable from any nonaccepting state by an empty string (a state cannot be accepting and nonaccepting at the same time).

The minimization algorithm breaks the automata states into groups that cannot be distinguished. That is, it creates groups of states that are equivalent to each other and therefore can be united to make a single state. By the course of work, states are partitioned into groups that cannot yet (or at all) be distinguished, and any two states from two different group are distinguishable. Upon the next iteration the current groups are broken into smaller ones in case the group has distinguishable states. The algorithm stops as soon as no groups can be partitioned anymore.

Before the algorithm starts working, the states are divided in two groups - accepting states and nonaccepting states, which are distinguishable by an empty string. Then we take a group from the current partition and check whether its states can be distinguished by some input character - whether some input character leads to two or more different state groups. If it does, new groups are created so that two states are grouped together if and only if they go to the same group on the same input character. The process is repeated for all groups in the current partition, then again for the new partition, until no group can be split further.

Then a representative is selected from each group so that:

[1] Do I have to give the proof that the minimal automaton exists?

1. The start state of the new automaton is the representative of the group containing the start state of the old automaton.

2. The accepting states of the new automaton are representatives of the groups that contain accepting states of the old automaton.

3. If state s is a representative of some group G and it has a transition to state t of the old automaton. Let r be the representative of the group containing t. Then we replace t in all of the transitions from s to the representative state r.

Therefore we create a new automaton with the minimum number of states.