CS 3813: Introduction to Formal Languages and Automata

State minimization and other algorithms for finite automata (Sec 2.6 - 2.7)

State minimization problem

- There can be more than one DFA that accepts the same language
- Among these equivalent DFAs, it is often useful to find the smallest, that is, the DFA with the minimum possible number of states
- This is especially important when DFAs are used for designing computer hardware circuits

Unreachable states

- A DFA sometimes contains states that cannot possibly be reached from the initial state
- These can easily be identified by
- Unreachable states can be removed without affecting the language accepted by the DFA

Example of unreachable state

Indistinguishable (equivalent) states

- States are said to be equivalent if merging them does not change the language accepted by a DFA
- Merging equivalent states is another way of simplifying a DFA without changing the language it accepts

Indistinguishable (equivalent) states

- States $q$ and $r$ of an automaton $M$ are said to be indistinguishable (written $q \equiv r$) if the automaton obtained from $M$ by making $q$ the initial state is equivalent to the automaton obtained from $M$ by making $r$ the initial state
- The relation $\equiv$ is an equivalence relation (reflexive, symmetric, and transitive) and divides the set of states of an automaton into equivalence classes
- Each equivalence class corresponds to a state in a minimal DFA
Detecting equivalence

• For any two states \( q \) and \( r \), let \( q \equiv_n r \) mean that these states are not distinguishable by any string of length less than \( n \).

• We can relate \( \equiv_n \) to \( \equiv_{n-1} \) as follows:

• For any two states \( q \) and \( r \) and integer \( n > 0 \), \( q \equiv_n r \) if and only if:
  1) \( q \equiv_{n-1} r \), and
  2) for all \( a \in \Sigma \), \( \delta(q,a) \equiv_{n-1} \delta(r,a) \).

State minimization algorithm

• Remove all unreachable states.

• Initialize the equivalence classes for \( \equiv_0 \) as \( F \) and \( Q - F \).

• Repeat for \( n = 1, 2, 3 \ldots \):
  – Compute the equivalence classes of \( \equiv_n \) from those of \( \equiv_{n-1} \).
  – Until \( \equiv_n \) is identical to \( \equiv_{n-1} \).

• These are the equivalence classes for \( \equiv \) and correspond to the states of the minimal DFA.

Exercise

Minimize this DFA. Also, minimize the DFA in the earlier example of unreachable states.

“Reversal”: A useful trick

Given any DFA, a NFA that accepts the reversal of the language accepted by the DFA can be constructed as follows:

– Reverse the direction of all the arcs in the state diagram.
– Change the start states to accepting states, and change the accepting states to start states.

Another algorithm for state minimization

• Construct an NFA that accepts the reversal of the language accepted by the DFA we want to minimize.

• Use the subset construction algorithm to create an equivalent DFA.

• Construct an NFA that accepts the reversal of the language accepted by this new DFA.

• Use the subset construction algorithm to create an equivalent DFA. This is the smallest DFA that is equivalent to the original DFA. (We don’t give a proof … but this works.)

Other algorithms for finite automata

• **Membership problem**: Given a finite automaton and string, determine whether the string is accepted by the automaton.

• **Emptiness problem**: Given a finite automaton, determine whether it accepts any string.
Algorithms for finite automata (continued)

- **Subset problem**: Given two finite automata, A and B, determine whether the language accepted by A is a subset of the language accepted by B

- **Solution hint**: First note that \( L(A) \subseteq L(B) \) if \( L(A) - L(B) = \emptyset \), and then recall that
\[
L(A) - L(B) = L(A) \cap \overline{L(B)} = \overline{L(A)} \cup L(B)
\]

Algorithms for finite automata (continued)

- **Equivalence problem**: Given two finite automata, determine whether they accept the same language

- **Solution hint**: Use the fact that automata A and B are equivalent if \( L(A) \subseteq L(B) \) and \( L(B) \subseteq L(A) \). Then use the algorithm for the subset problem.