CS 3813: Introduction to Formal Languages and Automata

Equivalence of NFAs and DFAs
Sec 2.3

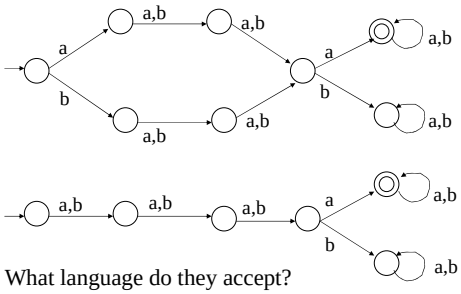
Equivalent automata

Two finite automata M_1 and M_2 are equivalent if

$L(M_1) = L(M_2)$,

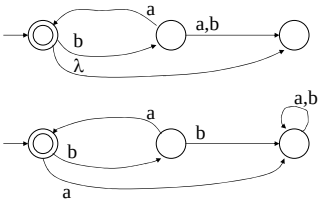
that is, if they both accept the same language.

Equivalent DFAs



What language do they accept?

Equivalent NFA and DFA



What language do they accept?

Equivalence of NFAs and DFAs

- We now show that DFAs and NFAs accept exactly the same set of languages. That is, nondeterminism does not make a finite automaton any more powerful.
- To show that NFAs and DFAs accept the same class of languages, we show two things:
  - any language accepted by a DFA can also be accepted by some NFA (this is easy to show -- how?)
  - any language accepted by a NFA can also be accepted by some DFA (this is more difficult to show -- it will take us the rest of class)

Proof strategy

- To show that any language accepted by a NFA is also accepted by some DFA, we describe an algorithm that takes any NFA and converts it into a DFA that accepts the same language
- The algorithm is called the "subset construction algorithm"
- We can use mathematical induction (on the length of a string accepted by the automaton) to prove that the DFA that is constructed accepts the same language as the NFA. (See theorem 2.2)
- You don't need to know the proof -- but you do need to remember the algorithm!

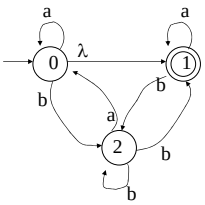
Subset construction algorithm

- *What does it do?* Given a NFA, it constructs a DFA that accepts the same language
- *What is the key idea?* The equivalent DFA simulates the NFA by keeping track of the possible states it could be in. Each state of the DFA corresponds to a subset of the set of states of the NFA -- hence, the name of the algorithm.
- If the NFA has n states, the DFA can have as many as 2^n states (why?), although it usually has many less.

Steps of subset construction algorithm

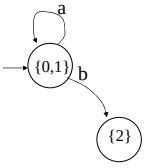
- The initial state of the DFA is the set of all states the NFA can be in without reading any input.
- For any state $\{q_i,q_j,\ldots,q_k\}$ of the DFA and any input a, the next state of the DFA is the set of all states of the NFA that can result as next states if the NFA is in any of the states $q_i,q_j,\ldots,q_k$ when it reads a. This includes states that can be reached by reading *a* followed by any number of λ-transitions. Use this rule to keep adding new states and transitions until it is no longer possible to do so.
- The accepting states of the DFA are those states that contain an accepting state of the NFA.

Example

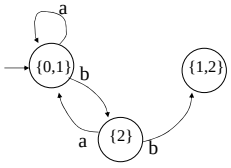


Here is a NFA that we want to convert to an equivalent DFA.

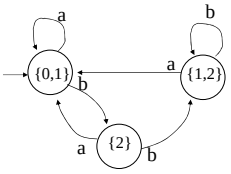
The start state of the DFA is the set of states the NFA can be in before reading any input. This includes the start state of the NFA and any states that can be reached by a e-transition.



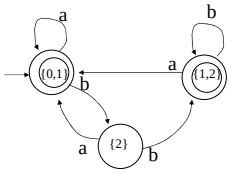
For state {0,1}, we create a transition for each possible input, a and b. In the process, we create state {2}.



For state {2}, we create a transition for each possible input, a and b. In the process, we create another state, {1,2}.

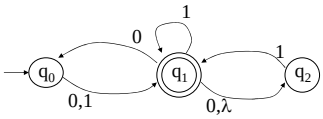


For state {1,2}, we create a transition for each possible input, a and b. At this point, a transition is defined for every state-input pair.



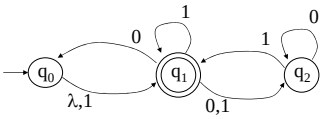
The last step is to mark the final states of the DFA.

Exercise



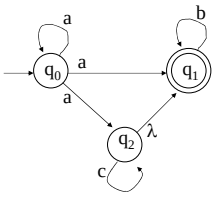
Use the subset construction algorithm to convert this NFA to an equivalent DFA.

Exercise



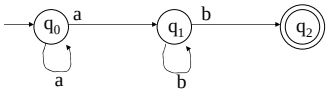
Use the subset construction algorithm to convert this NFA to an equivalent DFA.

Exercise



Use the subset construction algorithm to convert this NFA to an equivalent DFA.

Exercise



Use the subset construction algorithm to convert this NFA to an equivalent DFA.