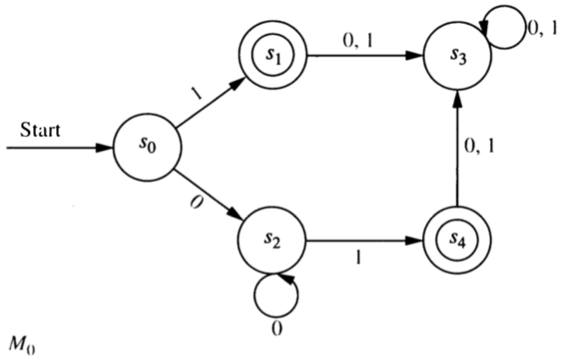
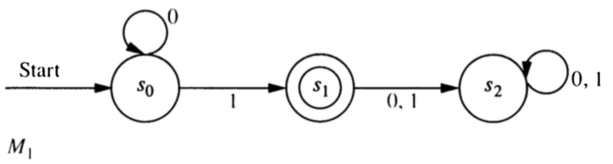
3.4 Equivalent Automata and Grammars

We will say that two automata M and N are equivalent if $L(M) = L(N)$, that is, if they recognize the same language. Similarly, two grammars are equivalent if they generate the same language.

7. What must you do to show that two automata are **not** equivalent?
8. What must you do to show that two automata **are** equivalent.
9. Are the two automata represented below equivalent? Explain.





3.5 Nondeterministic Automata

The finite state automata we have seen so far are often called **deterministic finite-state automata** or DFAs. There is a generalization called a **non-deterministic finite-state automaton** or NFA. The only difference is how the transition function is specified. For an NFA, the transition function has the form

$$f : S \times I \to P(S)$$

where S is the set of states, I is the input alphabet, and $P(S)$ is the powerset of S. (The powerset of S is the set of all subsets of S.) So the transition function for an NFA returns a *set* of states. That set could contain just one state, or multiple states, or no states at all (empty set).

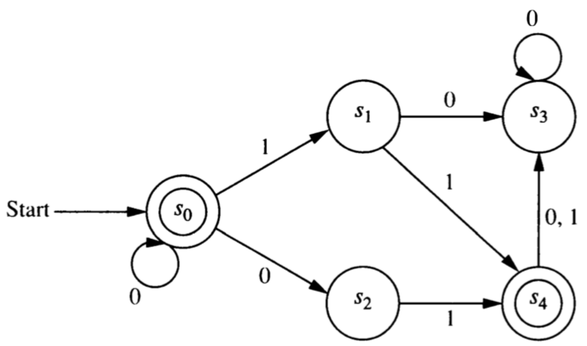
Here is a tabular description of the transition function for an NFA N_0 with start state A and accepting states C and D.

N_0								
state	A	A	B	B	C	C	D	D
letter	0	1	0	1	0	1	0	1
f	$\{A, B\}$	$\{D\}$	$\{A\}$	$\{B, D\}$	$\{\}$	$\{A, C\}$	$\{A, B, C\}$	$\{B\}$

10. Draw the graph representation. How will the graph of an NFA differ from the graph for a DFA?

11. How do we define the language recognized by an NFA? [Check your answer before proceeding.]
12. Which of these strings are accepted by N_0?
    a. 01
    b. 011
    c. 00
    d. 0000
    e. λ
    f. 010101

Here is the graph of an NFA N_1.



13. Make a table showing the transition function for N_1.
14. Which of the following strings are accepted by N_1?
    a. λ
    b. 00
    c. 01
    d. 010
    e. 011
    f. 001
15. Which is harder: to convince someone that a string *is* accepted by an NFA or that it *is not* accepted by an NFA? Why?
16. What is $L(N_1)$?
17. For each of the following languages
    - Construct an NFA that recognizes the language.
    - Construct a DFA that recognizes the language.

    In each case, try to do this with as few states and transitionas as you can. You may find it helpful to give your states good names that reflect what they represent.

    a. The set of bitstrings that either start 01 or start 10.
    b. The set of bitstrings that end with two 0's
    c. The set of bitstrings that either end 01 or end 10.

    Which do you find easier to construct: NFAs or DFAs? Why?
18. Create a DFA that recognizes $L(N_1)$.
19. Given an NFA N, is it always possible to create a DFA M that recognizes the same language? If so, explain how. If not, provide an example N and explain why $L(N)$ cannot be recognized by a DFA.