CS109B Notes for Lecture 4/19/95

Automata

Systems often may be modeled by a finite set of *states.*

- The system is always in one state.
- *Inputs* cause transitions from state to state.
- The system has an initial state.
- One or more states are *accepting*: they represent a successful sequence of inputs.

Why Automata?

Model important kinds of finite-state systems such as:

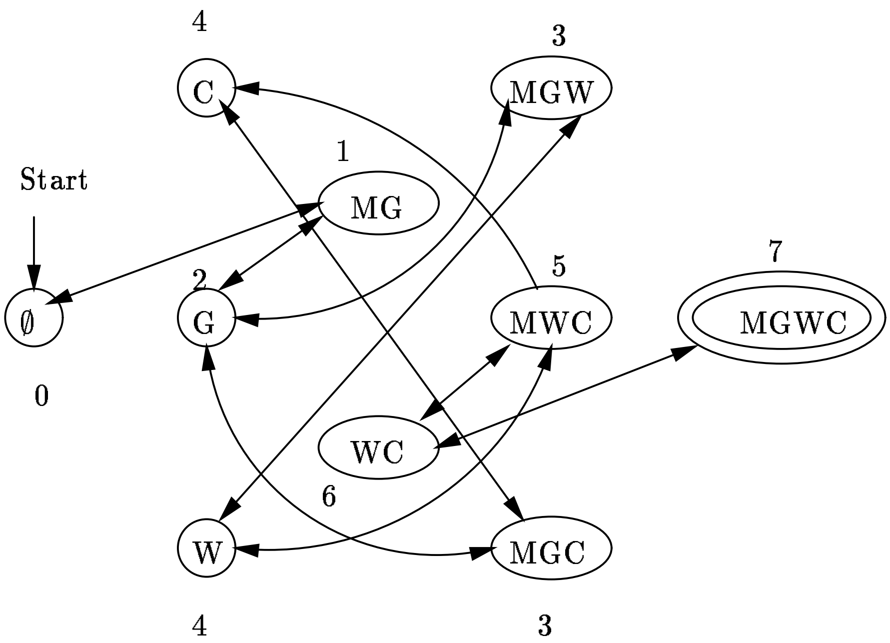
- Text-reading/processing software, especially compilers, UNIX commands like `grep` or `awk`.
- Communication protocols.
- Digital hardware components, e.g., mechanisms for sharing memory among processors.

Example: We may be familiar with the problem of the goat, wolf, and cabbage.

- A man, carrying all 3 must cross a river with a boat that can hold only him and one of the other three.
- If the wolf is left alone with the goat, the goat will be eaten; if the goat is left alone with the cabbage, the cabbage will be eaten.

Here, states are "the set of things on the far shore."

- Initial state indicated with "Start."
- Accepting state(s) indicated with double circles.
- This diagram is special in 2 ways:
    1. There is only one input: "cross."



2. All transitions are reversible; generally they are not.

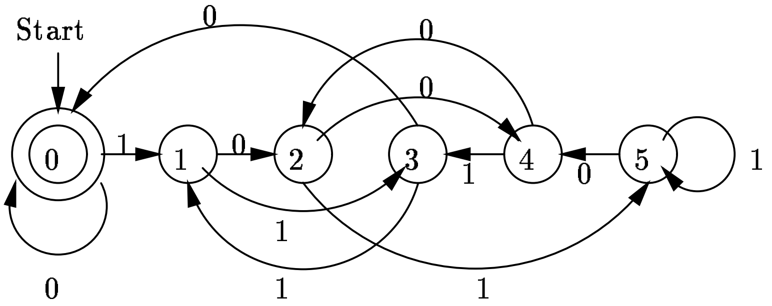
- Yet we can get some useful information, e.g., by DFS we can discover a path from the start state to the accepting state, i.e., from the state in which all 4 are on the near shore to the state in which all are on the far shore.
- We could find the length of the shortest path from the start to each state by Dijkstra's algorithm.
  - Give each arc a weight of 1.
  - Minimum distance marked on diagram.

Example: Suppose we wish to read a binary number left-to-right and test for divisiblity by 6.

- States = remainder mod 6 of integer read so far.
- Transition rule: on input 0 $s \rightarrow (2s \; mod \; 6)$; on input 1: $s \rightarrow (2s+1) \; mod \; 6$.

Nondeterministic Automata

The "divisible by 6" automaton has the property that no state has more than one transition out on any one input.



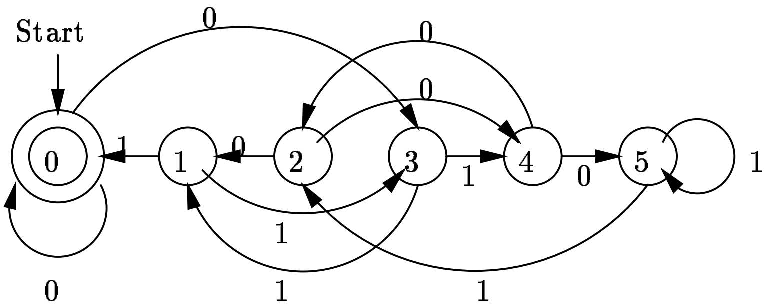
- Such an automaton is *deterministic* (a DFA).
  - □ Deterministic automata can be simulated easily, given a sequence of inputs.

The MWGC automaton has several transitions from each state on the lone input.

- Call such an automaton *nondeterministic* (an NFA).
  - □ It is hard to simulate such automata by programs, but they are often a help in design.

Example: Suppose we want to check "divisible by 6" but reading from right to left.

- Just reverse all the arcs on the automaton above and swap initial and accepting state (they are the same in this example).



- Oops — the automaton is no longer deterministic.

Why Nondeterminism?

- Easier design. Example above. Also lexical analyzers (the UNIX command `lex`, e.g.) use nondeterminism in designing compilers.

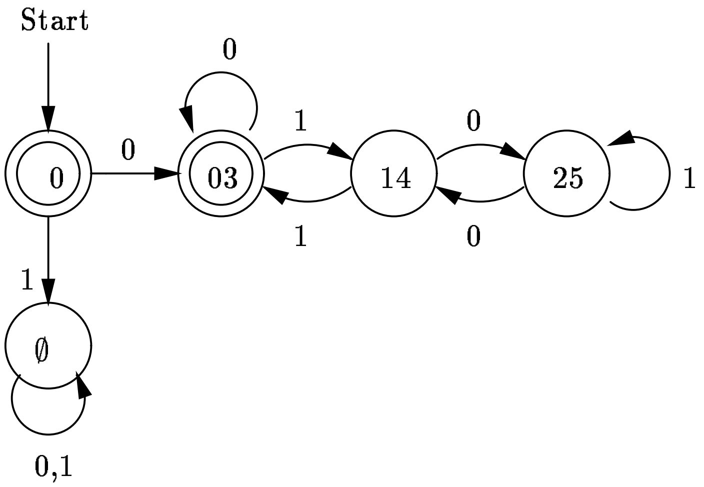
- Nondeterministic programs (not NFA's) are used to describe solutions to NP-complete problems.

The Subset Construction

We can convert any NFA N to a DFA D.

- The states of D are the sets of states of N.
- The start state of D is the set containing only the start state of N.
- The accepting states of D are the sets that include at least one accepting state of N.
- The transitions of D:
  1. Let S be a state of D (= a set of states of N). Let a be an input.
  2. The transition from S on input a in DFA D is to the set of N's states $T = \{t \mid$ for some s in S, N has a transition on a from s to $t\}$.

Example:



- Hint: instead of computing all 2^n states of D (if N has n states), compute the states and transitions "on demand," beginning from the start state.