Chapter Two:
Finite Automata

In theoretical computer science, **automata theory** is the study of abstract machines (or more appropriately, abstract 'mathematical' machines or systems) and the computational problems that can be solved using these machines. These abstract machines are called automata. Automata comes from the Greek word αὐτόματα meaning "self-acting".
 - Wikipedia

Finite Automata

- One way to define a language is to construct an *automaton*
  - a kind of abstract computer that takes a string as input and produces a yes-or-no answer.
- The language it defines is the set of all strings for which it says yes.

Finite Automata

- The simplest kind of automaton is the *finite* automaton.
- The more complicated automata we discuss later have some kind of *unbounded* memory to work with; in effect, they will be able to grow to whatever size necessary to handle the input string they are given.
- finite automata have no such power.
  - A finite automaton has a finite memory that is fixed in advance.
  - Whether the input string is long or short, complex or simple, the finite automaton must reach its decision using the same fixed and finite memory.

Outline

- 2.1 Man Wolf Goat Cabbage
- 2.2 Not Getting Stuck
- 2.3 Deterministic Finite Automata
- 2.4 The 5-Tuple
- 2.5 The Language Accepted by a DFA

A Classic Riddle

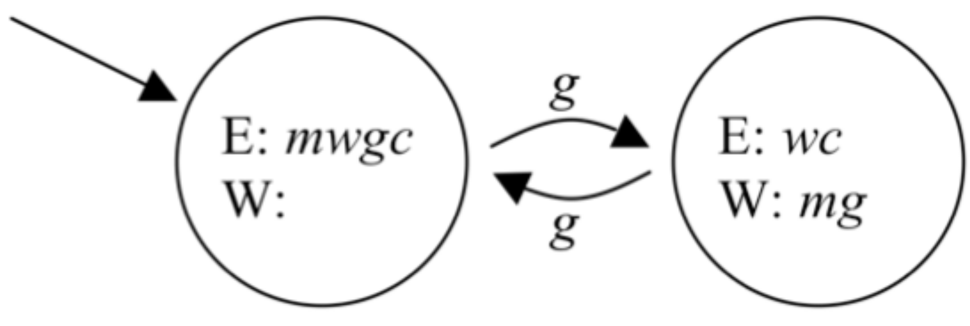
- A man travels with wolf, goat and cabbage
- Wants to cross a river from east (E) to west (W)
- A rowboat is available, but only large enough for the man plus one possession
- Wolf eats goat if left alone together
- Goat eats cabbage if left alone together
- How can the man cross without loss?

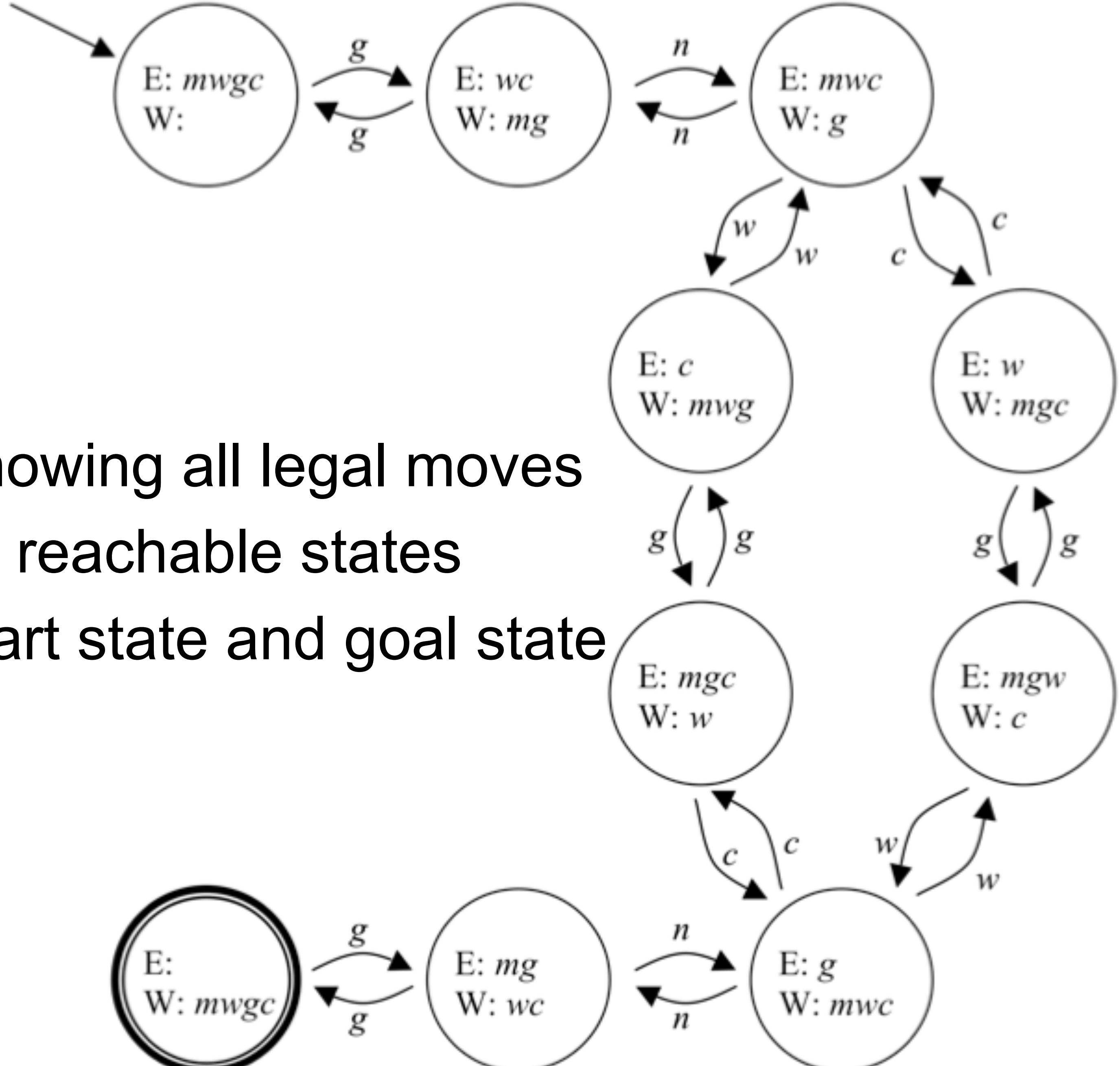
Solutions As Strings

- Four moves can be encoded as four symbols:
  - Man crosses with wolf (*w*)
  - Man crosses with goat (*g*)
  - Man crosses with cabbage (*c*)
  - Man crosses with nothing (*n*)
- Then a sequence of moves is a string, such as the solution *gnwgcng*:
  - First cross with *g*oat, then cross back with *n*othing, then cross with *w*olf, …

Moves As State Transitions

- Each move takes our puzzle universe from one state to another - a state is the configuration of occupants on each side of the river.
- For example, the *g* move is a transition between these two states:





- Showing all legal moves
- All reachable states
- Start state and goal state

The Language Of Solutions

- Every path gives some $x \in \{w,g,c,n\}^*$
- The diagram defines the language of solutions to the problem:

  $\{x \in \{w,g,c,n\}^* \mid$ starting in the start state and following the transitions of x ends up in the goal state$\}$

- Recall: A language is the set of all strings for which an automaton says yes (ends up in the goal state).
- This is an infinite language (why?)
- The two shortest strings (solutions) in the language are *gnwgcng* and *gncgwng*

Outline

- 2.1 Man Wolf Goat Cabbage
- 2.2 Not Getting Stuck
- 2.3 Deterministic Finite Automata
- 2.4 The 5-Tuple
- 2.5 The Language Accepted by a DFA

What happens if we try a string that is not in the language?

Consider *gnwn…*we get stuck with nowhere to go.

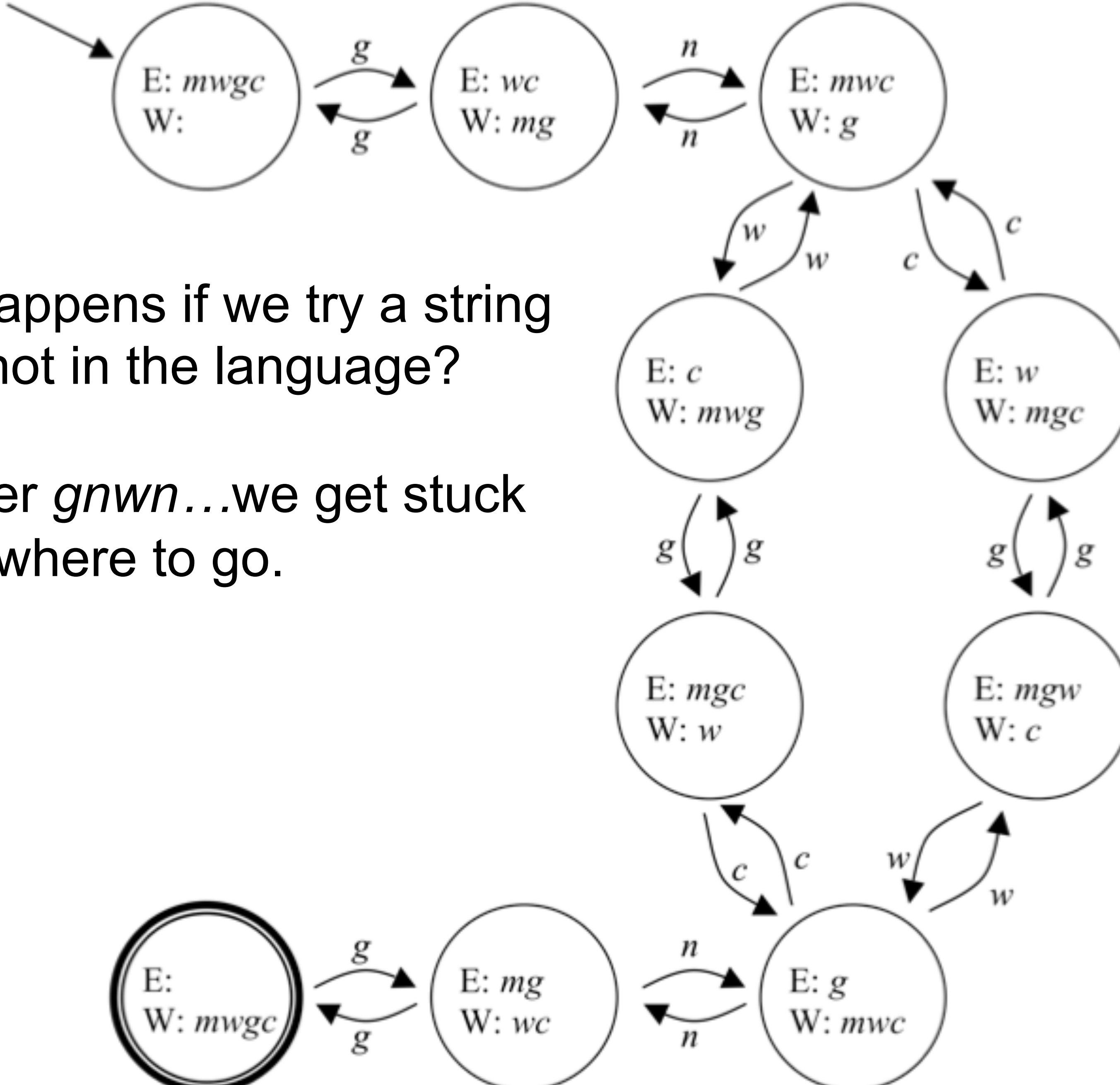
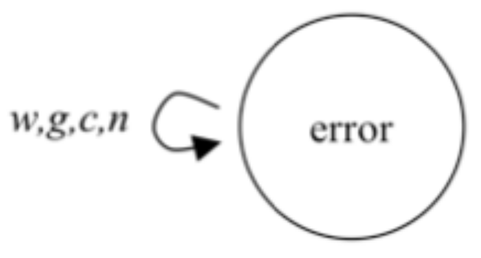
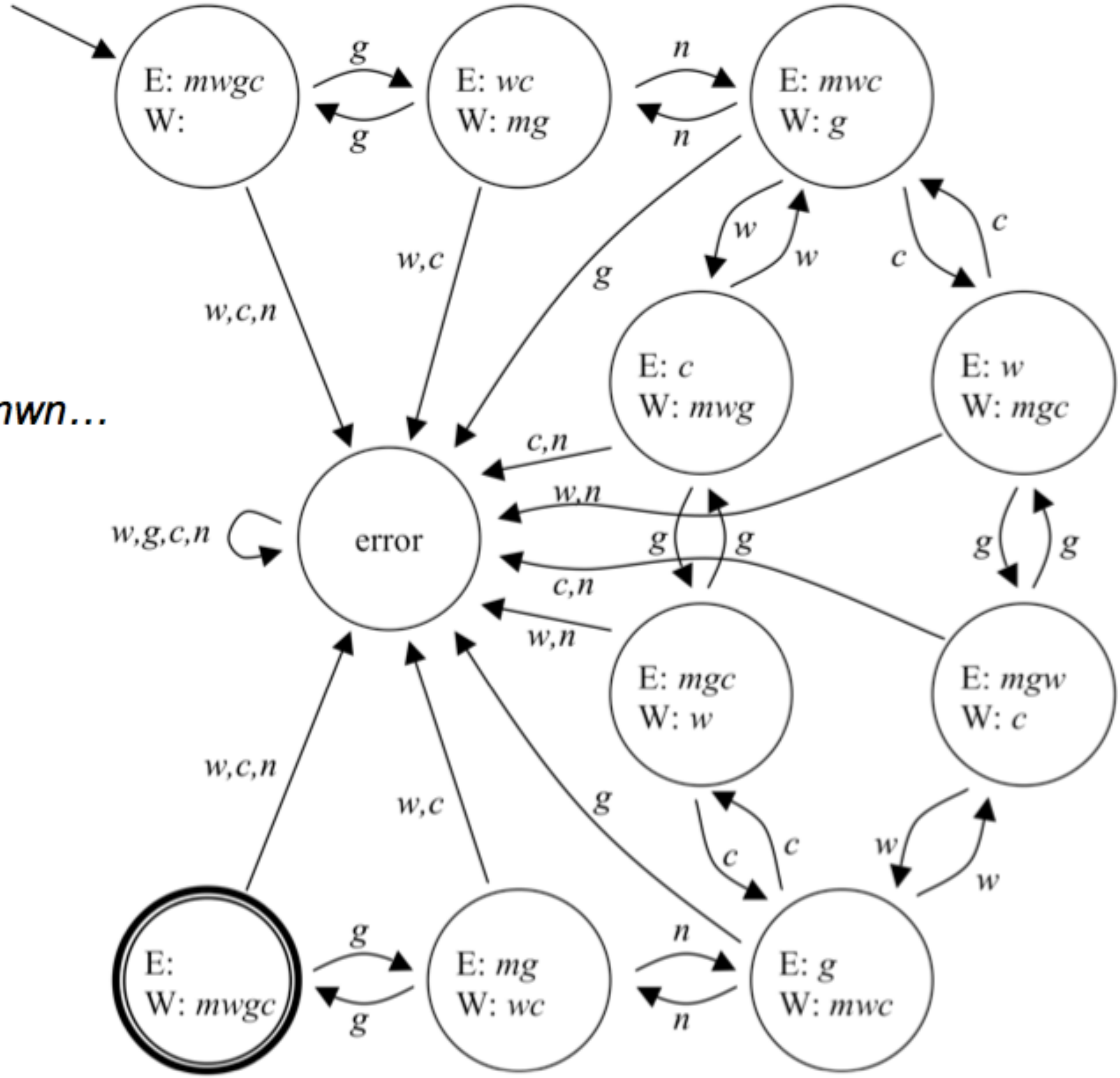


Diagram Gets Stuck

- On many strings that are not solutions, the previous diagram gets stuck
- Automata that never get stuck are easier to work with
- We'll need one additional state to use when an error has been found in a solution



Now try *gnwn…*


E: mwgc
W:
g
g
E: wc
W: mg
n
n
E: mwc
W: g
w,c,n
w,c
g
w
w
c
c
E: c
W: mwg
E: w
W: mgc
w,g,c,n
error
c,n
w,n
g
g
g
g
c,n
w,n
E: mgc
W: w
E: mgw
W: c
w,c,n
w,c
g
c
c
w
w
E:
W: mwgc
g
g
E: mg
W: wc
n
n
E: g
W: mwc

Complete Specification

- The diagram shows exactly one transition from every state on every symbol in Σ
- It gives a computational procedure for deciding whether a given string is a solution:
  - Start in the start state
  - Make one transition for each symbol in the string
  - If you end in the goal state, accept; if not, reject

Outline

- 2.1 Man Wolf Goat Cabbage
- 2.2 Not Getting Stuck
- 2.3 Deterministic Finite Automata
- 2.4 The 5-Tuple
- 2.5 The Language Accepted by a DFA

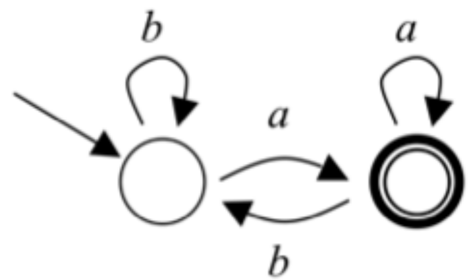
DFA: Deterministic Finite Automaton

- An informal definition (formal version later):
  - A diagram with a finite number of states represented by circles
  - An arrow points to one of the states, the unique *start state*
  - Double circles mark any number of the states as *accepting states*
  - For every state, for every symbol in Σ, there is exactly one arrow labeled with that symbol going to another state (or back to the same state)

DFAs Define Languages

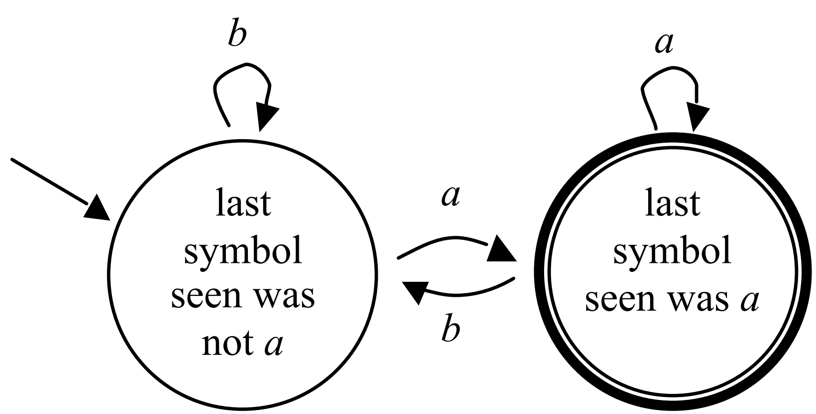
- Given any string over Σ, a DFA can read the string and follow its state-to-state transitions
- At the end of the string, if it is in an accepting state, we say it accepts the string
- Otherwise it rejects
- The language defined by a DFA is the set of strings in Σ^* that it accepts

Example



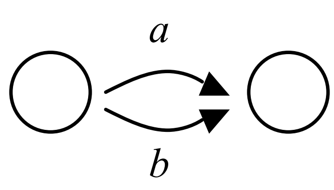
Consider the Strings:
-aba
-bab

- This DFA defines $\{xa \mid x \in \{a,b\}^*\}$
- No labels on states (unlike man-wolf-goat-cabbage)
- Labels can be added, but they have no effect, like program comments:

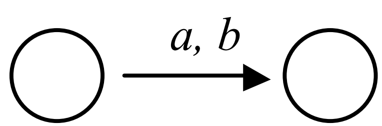


A DFA Convention

- We don't draw multiple arrows with the same source and destination states:



- Instead, we draw one arrow with a list of symbols:



Outline

- 2.1 Man Wolf Goat Cabbage
- 2.2 Not Getting Stuck
- 2.3 Deterministic Finite Automata
- 2.4 The 5-Tuple
- 2.5 The Language Accepted by a DFA

The 5-Tuple (Formal Definition)

A DFA M is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where:

- Q is the finite set of states
- Σ is the alphabet (that is, a finite set of symbols)
- $\delta \in (Q \times \Sigma \rightarrow Q)$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accepting states

- Q is the set of states
  - Drawn as circles in the diagram
  - We often refer to individual states as q_i
  - The definition requires at least one: q_0, the start state
- F is the set of all those in Q that are accepting states
  - Drawn as double circles in the diagram

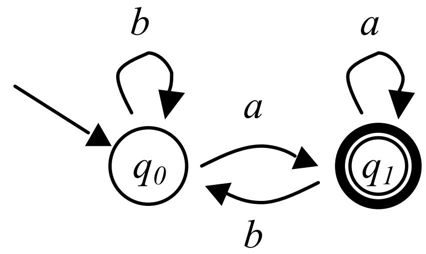
The 5-Tuple (Formal Definition)

A DFA M is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where:

- Q is the finite set of states
- Σ is the alphabet (that is, a finite set of symbols)
- $\delta \in (Q \times \Sigma \rightarrow Q)$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accepting states

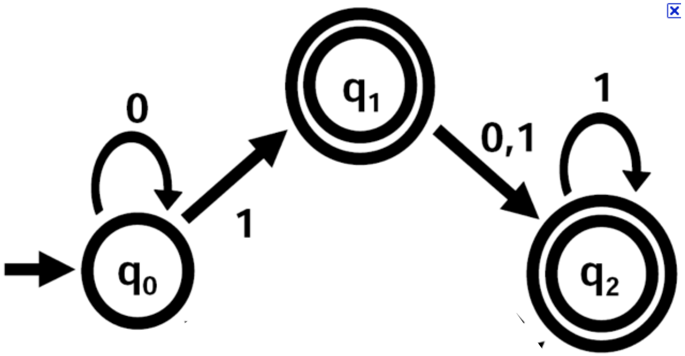
- δ is the transition function
  - A function $\delta(q,a)$ that takes the current state q and next input symbol a, and returns the next state
  - Represents the same information as the arrows in the diagram

Example:



- This DFA defines $\{xa \mid x \in \{a,b\}^*\}$
- Formally, $M = (Q, \Sigma, \delta, q_0, F)$, where
  - $Q = \{q_0,q_1\}$
  - $\Sigma = \{a,b\}$
  - $F = \{q_1\}$
  - $\delta(q_0,a) = q_1$, $\delta(q_0,b) = q_0$, $\delta(q_1,a) = q_1$, $\delta(q_1,b) = q_0$
- Names are conventional, but the order is what counts in a tuple
- We could just say $M = (\{q_0,q_1\}, \{a,b\}, \delta, q_0, \{q_1\})$

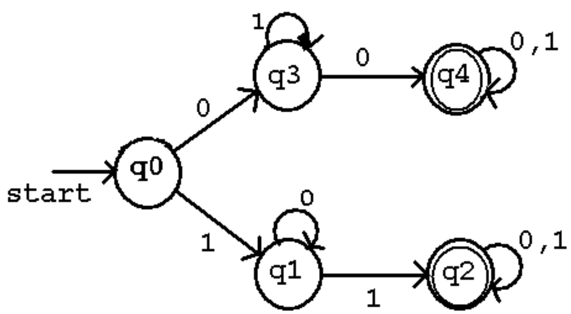
Another DFA



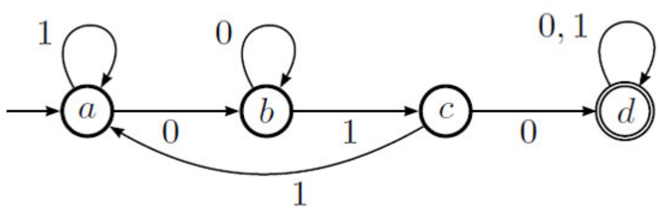
- What is the alphabet?
- Informally describe the language of this DFA
- Write down the formal definition of this DFA.

More DFAs

a)



b)



For each of these DFAs:

- What is the alphabet?
- Informally describe the language of this DFA
- Write down the formal definition of this DFA.

Languages

- For each of the following languages construct a DFA that recognizes it:
  - $\{x \in \{a, b\}^* \mid |x| \leq 2\}$
  - $\{x \in \{a, b\}^* \mid$ x is a string with 0 or more *a*'s followed by 0 or more *b*'s$\}$
  - $\{x \in \{a, b\}^* \mid x$ contains one *a* and two *b*s$\}$

Outline

- 2.1 Man Wolf Goat Cabbage
- 2.2 Not Getting Stuck
- 2.3 Deterministic Finite Automata
- 2.4 The 5-Tuple
- 2.5 The Language Accepted by a DFA

The δ^* Function

- The δ function gives 1-symbol moves
- We'll define δ^* so it gives whole-string results (by applying zero or more δ moves)
- A recursive definition:
  - $\delta^*(q,\varepsilon) = q$
  - $\delta^*(q,xa) = \delta(\delta^*(q,x),a)$
- That is:
  - For the empty string, no moves
  - For any string *xa* (*x* is any string and *a* is any final symbol) first make the moves on *x*, then one final move on *a*

M Accepts x

- Now $\delta^*(q,x)$ is the state M ends up in, starting from state q and reading all of string x
- So $\delta^*(q_0,x)$ tells us whether M accepts x:

A string $x \in \Sigma^*$ is accepted by a DFA $M = (Q, \Sigma, \delta, q_0, F)$ if and only if $\delta^*(q_0, x) \in F$.

Regular Languages

For any DFA $M = (Q, \Sigma, \delta, q_0, F)$, $L(M)$ denotes the language accepted by M, which is
$L(M) = \{x \in \Sigma^* \mid \delta^*(q_0, x) \in F\}$.

A *regular language* is one that is $L(M)$ for some DFA M.

- To show that a language is regular, give a DFA for it; we'll see additional ways later
- To show that a language is *not* regular we have to show that it is not possible to construct a DFA for it (this is typically much more difficult - we'll see a proof technique for this later)

Are these Languages Regular?

- $\{(ab)^n \mid n > 0\}$
- $\{\mathrm{a}^m\mathrm{b}^n \mid m,n > 0\}$
- $\{a^n b^n \mid n > 0\}$

Assignment #1

- See course website