Week 13 - Wednesday

COMP 2230

Last time

- More on formal languages

Questions?

Assignment 6

Logical warmup

- A silver prospector didn't have the cash to pay his March rent in advance
- But he owned a bar of pure silver, 31 inches long
- He made the following arrangement with his landlady:
  - He would cut the bar into smaller pieces
  - On the first day of March he would give the lady an inch of the bar, and on each day, he would add another inch to her amount of silver
  - She would keep the silver as collateral
  - At the end of the month, when the prospector expected to be able to pay his rent in full, she would return the pieces to him
- March has 31 days, so he could cut the bar into 31 one-inch sections
- But since it was difficult to cut the bar, the prospector wished to carry out his agreement
- with the fewest possible number of pieces
- For example, he might give the lady an inch on the first day, another inch the second day, then on the third day he could take back the two pieces and give her a solid 3-inch section
- What's the smallest number of pieces the prospector needs to cut his bar into so that his landlady can always have the amount needed for the given day?

Back to Formal Languages

Examples

- Let $\Sigma = \{0, 1\}$
- Find regular expressions for the following languages:
  - The language of all strings of 0's and 1's that have even length and in which the 0's and 1's alternate
  - The language consisting of all strings of 0's and 1's with an even number of 1's
  - The language consisting of all strings of 0's and 1's that do not contain two consecutive 1's
  - The language that gives all binary numbers written in normal form (that is, without leading zeroes, and the empty string is not allowed)

Practical notation

- Regular expressions are used in some programming languages (notably Perl and Javascript) and in **`grep`** and other find and replace tools
- The notation is generally extended to make it a little easier, as in the following:
- **`[A-C]`** means any character in that range,
  - **`[A-C]`** means **`(A|B|C)`**
  - **`[0-9]`** means **`(0|1|2|3|4|5|6|7|8|9)`**
- **`[ABC]`** means **`(A|B|C)`**
- **`ABC`** means the concatenation of **`A`**, **`B`**, and **`C`**
- A dot stands for any character: **`A.C`** could match **`AxC`**, **`A&C`**, **`ABC`**
- **`^`** means NOT, thus **`[^D-Z]`** means not the characters **`D`** through **`Z`**
- Repetitions:
  - **`R?`** means 0 or 1 repetitions of **`R`**
  - **`R*`** means 0 or more repetitions of **`R`**
  - **`R+`** means 1 or more repetitions of **`R`**
- Notations vary and have considerable complexity
- Use this notation to describe the regular expression for legal Java identifiers

Finite-State Automata

Three-Sentence Summary

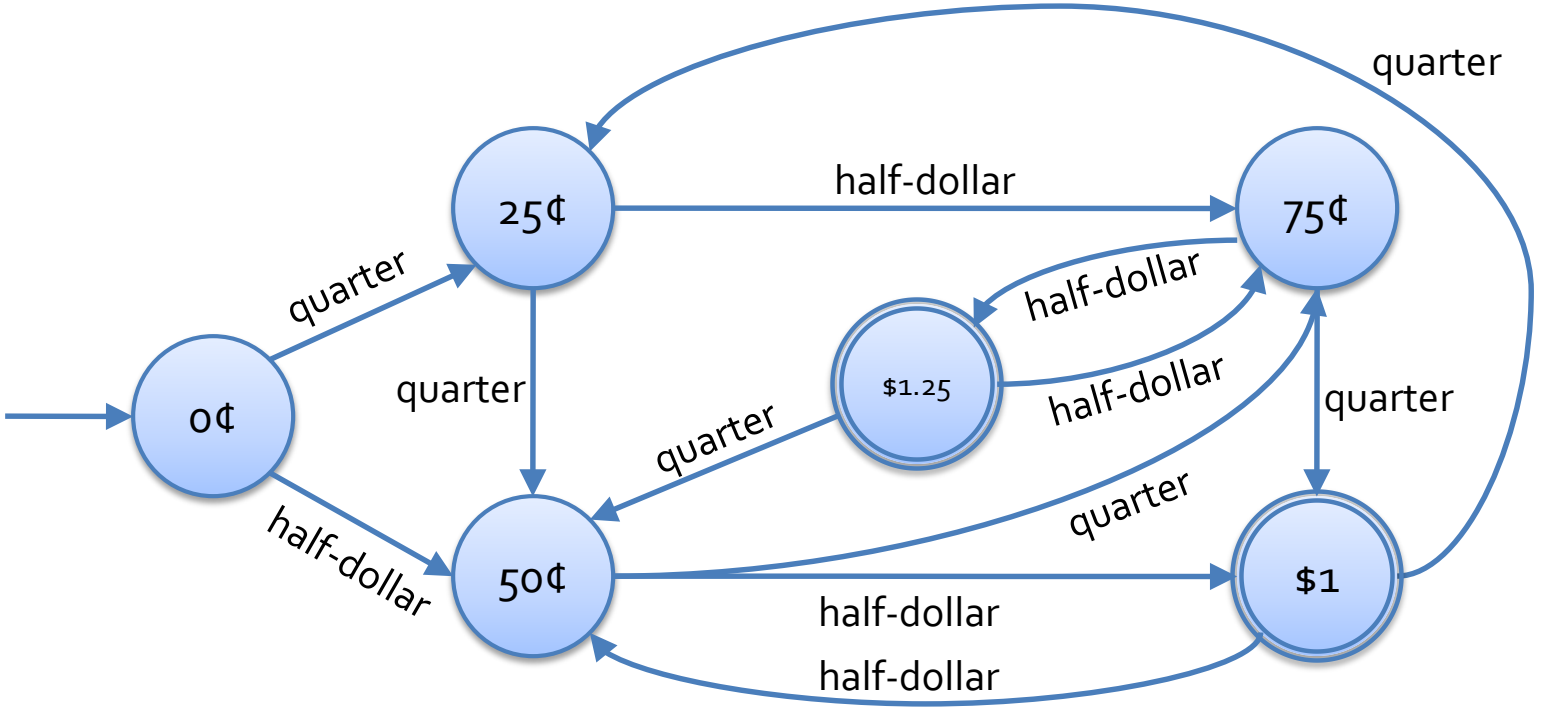
Finite-State Automata

Finite-state automaton

- A finite-state automaton is an idealized machine composed of five objects:
  1. A finite set I, called the **input alphabet**, of input symbols
  2. A set S of **states** the automaton can be in
  3. A designated state s_0 called the **initial state**
  4. A designed set of states called the set of **accepting states**
  5. A **next-state function** N: $S \times I \rightarrow S$ that maps a current state with current input to the next state

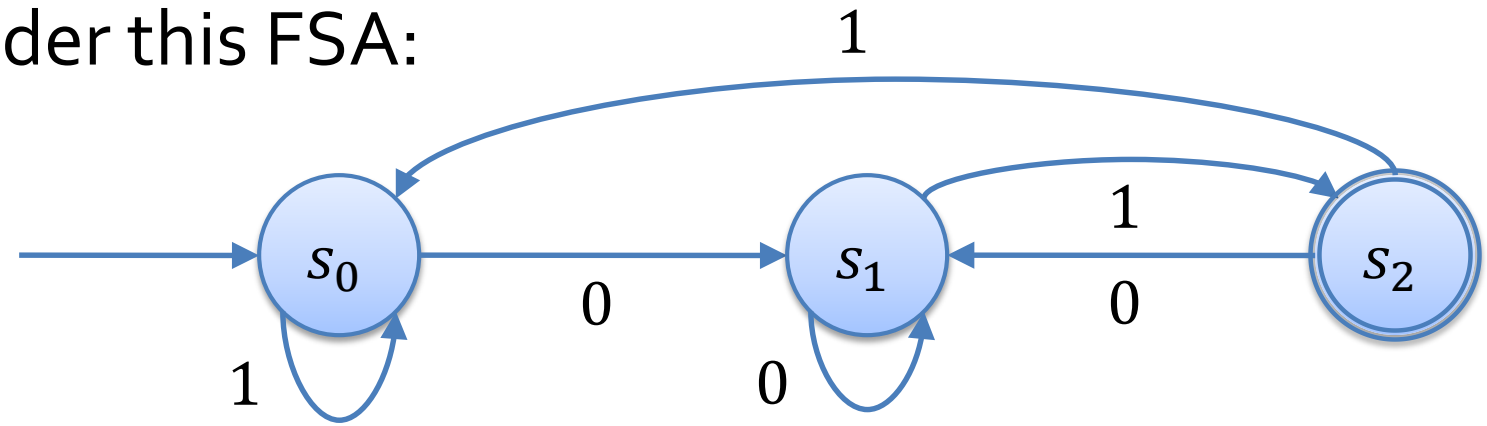
Transition diagram

- FSA's are often described with a state transition diagram
  - The starting state has an arrow
  - The accepting states are marked with circles
  - Each rule is represented by a labeled transition arrow
- The following FSA represents a vending machine



FSA example

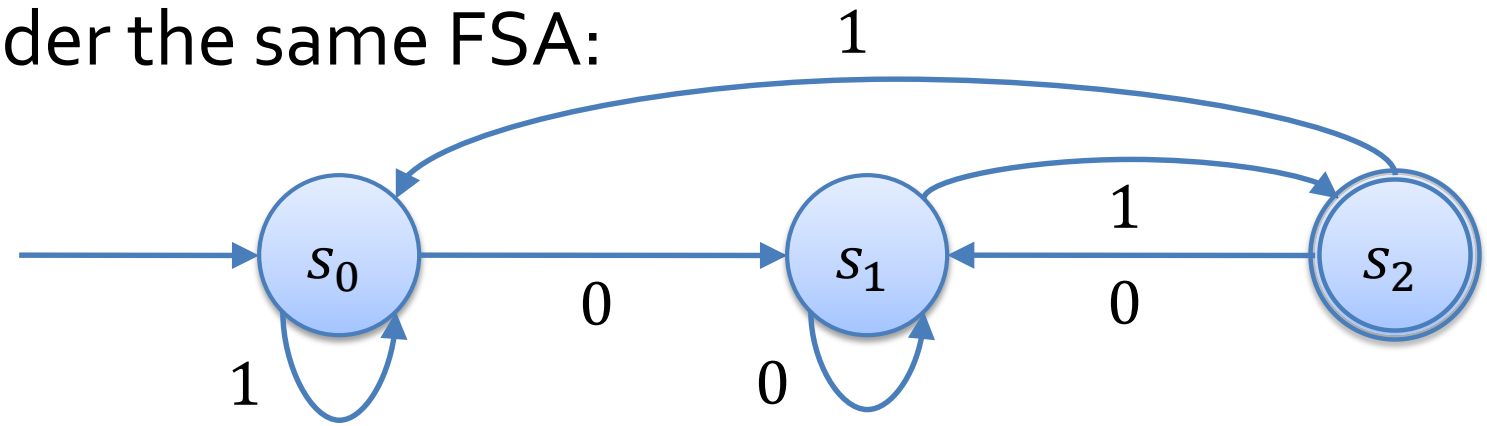
- Consider this FSA:



- What are its states?
- What are its input symbols?
- What is the initial state?
- What are the accepting states?
- What is $N(s_1, 1)$?
- What's a verbal description for the strings accepted?

Annotated next-state tables

- Consider the same FSA:



- We can also describe an FSA using an **annotated next-state table**
- A next-state table gives shows what the transition is for each state for all possible input
- An annotated next-state table also marks the initial state and accepting states
- Find the annotated next-state table for this FSA

Table to transition diagram

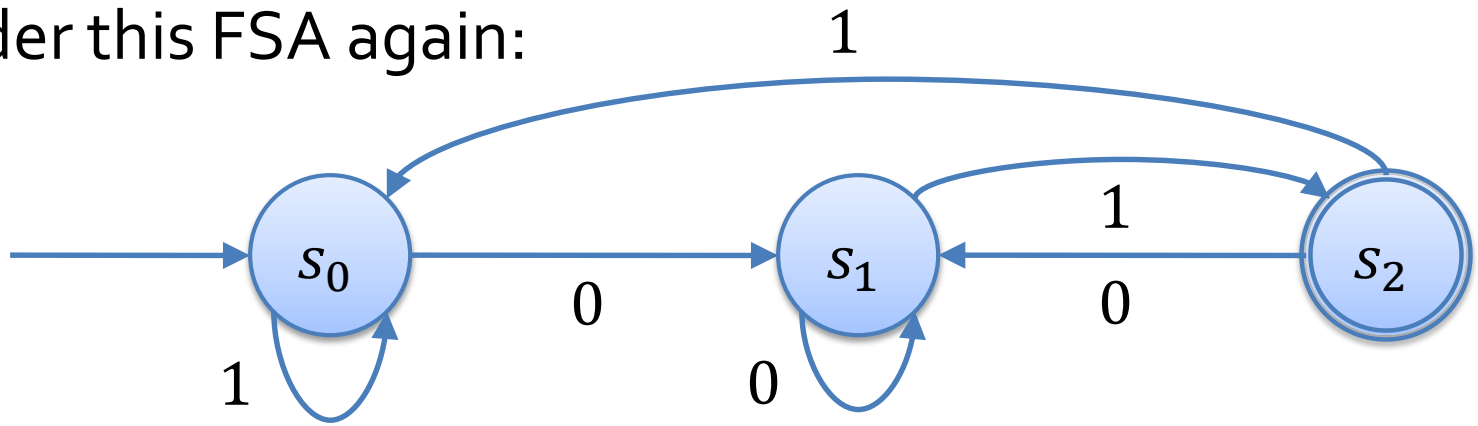
- Consider the following annotated next-state table
  - → marks initial state
  - • marks accepting states:

		a	b	c
→	U	Z	Y	Y
•	V	V	V	V
	Y	Z	V	Y
•	Z	Z	Z	Z

- Draw the corresponding transition state diagram

FSA example

- Consider this FSA again:



- Which state will be reached on the following inputs:
  i. 01
  ii. 0011
  iii. 0101100
  iv. 10101
- What's a verbal description for the strings accepted?

Eventual-state function

- Let A be a FSA with a set of states S, set of input symbols I, and next-state function N: $X \times I \rightarrow S$
- Let I^* be the set of all strings over I
- The **eventual-state function** N^*: $X \times I^* \rightarrow S$ is the following
  - $N^*(s, w)$ = the state that A goes to if the symbols of w are input to A in sequence, starting with A in state s
- All of this is just a notational convenience so that we have a way of talking about the state that a *whole string* will transition an FSA to
- We say that w **is accepted by** A iff $N^*(s_0, w)$ is an accepting state of A
- The language of A, $L(A) = \{ w \in I^* \mid w \text{ is accepted by } A \}$

Designing automata

- Design a finite-state automaton that accepts the set of all strings of 0's and 1's such that the number of 1's in the string is divisible by 3
- Make a regular expression for this language

- Design a finite-state automaton that accepts the set of all strings of 0's and 1's that contain exactly one 1
- Make a regular expression for this language

Upcoming

Next time…

- Simplifying finite state automata

Reminders

- Keep working on Assignment 6
- Read 12.3 for Friday
  - Prepare a three-sentence summary
  - Extra credit if you get called on