Proofs, Strings, and Finite Automata

CS154

Chris Pollett

Feb 5, 2007.

Outline

- Proofs and Proof Strategies
- Strings

Finding proofs

- **Example:** For every graph G, the sum of the degrees of all the nodes in G is an even number.
  - Might approach problem by checking cases like when graph has a small number of vertices. One might then notice each edge contributes two to the total sum and that the sum of degrees = 2·(number of edges in graph).
- **Types of proofs**:
  - **by construction**: example, there is a graph consisting of n nodes with only one cycle. Proof: let $V=\{1,..,n\}$, let $E=\{\{1,2\}, \{2,3\}, \ldots\{n-1,n\}\}\cup\{\{1,n\}\}$.
  - **by contradiction**: example $2^{1/2}$ is not a rational number. Idea

    if not can assume $2^{1/2} = m/n$ where m and n share no common factor. In which case, one is odd, the other even. Squaring both sides gives: $2n^2 = m^2$, so m is even because square of an odd number is odd. So $m=2k$, so $2n^2 = (2k)^2= 4k^2$. So $n^2= 2k^2$ implying n is also even, giving a contradiction.
  - **by induction**: Show $\sum^{n}_{i=1} i = n(n+1)/2$

    Base case: For n=1 we have $\sum^{1}_{i=1} i =1 = 1(1+1)/2$.

    Induction step: Assume $\sum^{n}_{i=1} i = n(n+1)/2$ holds, we want to show $\sum^{n+1}_{i=1} i = (n+1)(n+2)/2$. Notice $\sum^{n+1}_{i=1} i = \sum^{n}_{i=1} i + (n+1)$. By our hypothesis, this in turn equals $n(n+1)/2 + (n+1)$. Making the denominators the same, this is:

    $$[n(n+1) + 2(n+1)]/2 = (n+1)(n+2)/2.$$

    Conclude the induction holds. So for all n, $\sum^{n}_{i=1} i = n(n+1)/2$ is true.

Strings

- Strings of characters are one of the fundamental building blocks of computer science.
- For this class, we will define an **alphabet** to be some nonempty finite set. For example,

  $\Sigma = \{0, 1\}$

  $\Sigma = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$
- The members of this set are called the **symbols** of the alphabet.
- A **string** over an alphabet is a finite sequence of symbols from that alphabet. For example, 0100. Here 0111 abbreviates the formal sequence (0,1,1,1)
- The **length** of a string w, $|w|$ is the number of symbols it contains. For example, $|0100| = 4$
- The **empty string** is written as λ (in many other books it is written as ε).

Strings and Languages

- The **reverse** of a string w, w^R, is the string consists of the symbols of w in reverse order: $001^R = 100$.
- A string is z is a **substring** of w if z appear consecutively within w. So 011 is a substring 10**011**01.
- The **concatenation** of two string x and y, xy, is the string consisting of the symbols in x followed by the symbols of y.
- In the string, xy, x would be called the **prefix; y** would be called the **suffix.**
- We write x^k to denote x concatenated to itself k times.
- A **language** is a set of string.

Example Languages

- Suppose $\Sigma=\{0,1\}$ is our alphabet.
- Then L= {1, 11, 101} is an example language.
- We write Σ^* for the set of all strings over Σ.
- Given a language L, we define:

  $L^0 = \{\lambda\}$,

  $L^{n+1} = \{wv \mid w \in L^n \wedge v \in L\}$.

  $L^* = \bigcup_{n\geq 0} L^n$.

  $L^+ = L \; L^*$

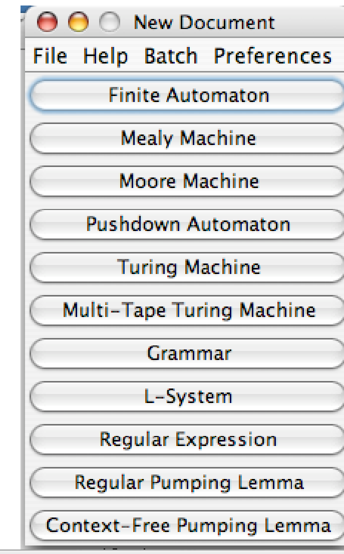
- Given these definitions for the L above, we have λ is L^*, but not in L^+, we have 11101 is in L^2, we have 1110111 is in L^3, L^*, and L^+, but not L^2.

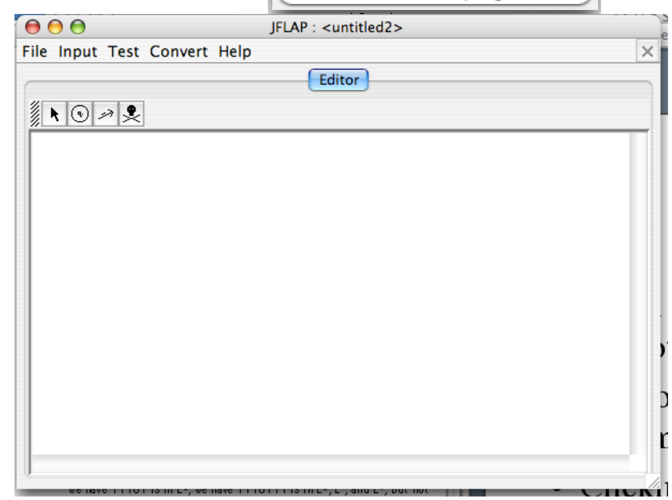
Machines

- We will now begin our study of how to build machines which can recognize languages.
- As a prelude, I will demo JFLAP now.
- To download JFLAP please use the link on the class page.

Running JFLAP

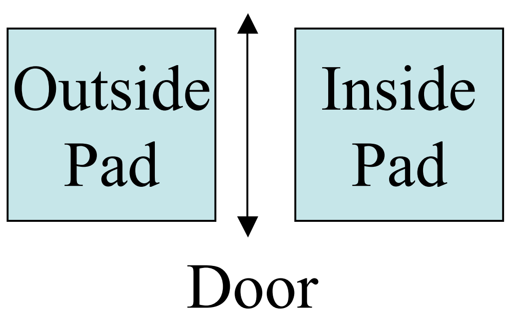
- When you launch JFLAP you get a window like:
- For now we will be using the Finite Automaton button.
- Clicking it will give a window like:
- The four buttons across the top of the edit area allow one to: (1) select a state, (2) create a state, (3) create a transitions, (4) delete a state or transition
- You can save the automaton you make using the File menu.



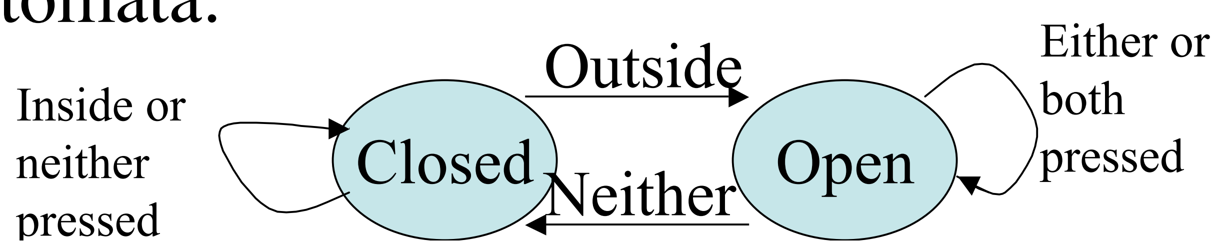


Introductory Examples

- Finite automata are computer models which are useful when one has very limited memory availability.
- Consider an automatic door say at a grocery store.



- We can model the door state this using a finite automata:



More on Door Example

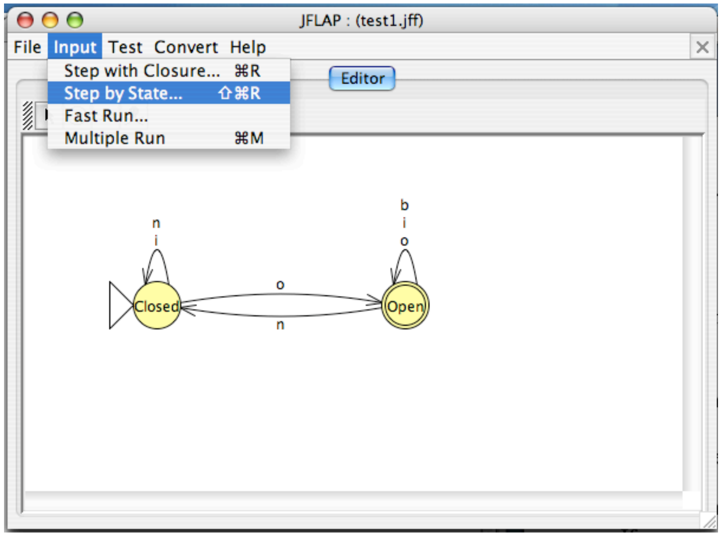
- The controller might start in a CLOSED state and receive the signals: OUTSIDE, INSIDE, NEITHER, INSIDE, BOTH, OUTSIDE, INSIDE NEITHER.
- It would then transition between the states CLOSED (start), OPEN, OPEN, CLOSED, CLOSED, CLOSED, OPEN, OPEN, CLOSED.
- Notice only need 1-bit of memory to keep track of state.
- It is also straightforward to represent transitions in a table:

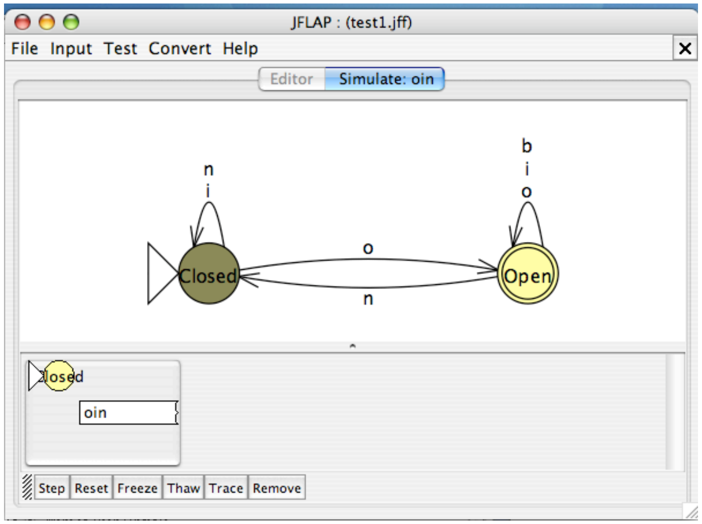
	Neither	Outside	Inside	Both
Closed	Closed	Open	Closed	Closed
Open	Closed	Open	Open	Open

- Finite automata and the their probabilistic counterparts called **Markov chains** are also useful for pattern recognition. For example, recognizing keywords in programming languages. Or figuring out which word English is likely based on the previous ones seen.

Running an Automaton in JFLAP on Different Inputs

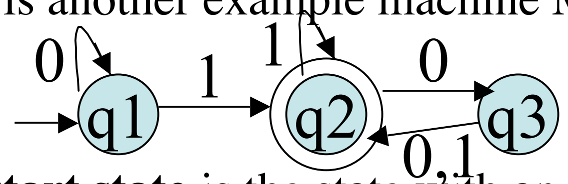
- We can build the automaton we just discussed in JFLAP.
- We'll use i for inside, o for outside, n for neither, and b for both.
- To test this automaton on some inputs we first need to say what state it starts in by right clicking on a state and setting it to be a start state.
- If we want to say what final states should be viewed as good (aka accepting) we can also do this by right-clicking.
- Then using the Input menu, we can select to run the automaton Step by State.
- You will be prompted for an input to the automaton, at which point you then get a window that let's you step through its computation.





Names for things

- The picture we drew of our automata a couple slides back is called a **state diagram**.
- We will usually use the variables M, N,… for machines.
- Here is another example machine M_1:



- The **start state** is the state with an arrow going from nowhere into it.
- If we are recognizing strings then we stop processing when we get to the end of a string of inputs.
- If we are in a double circled state at that point we accept the string otherwise we reject it. So double circled states called **accept states**.
- Arrows going from one state to another are called **transitions**.
- You might want to see if you can figure out if the above automata accepts each of the following strings: 000, 0110, 1101.

Formal Definition

- A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where
  1. Q is a finite set called the **states**.
  2. Σ is a finite set called the **alphabet**.
  3. $\delta: Q \times \Sigma \rightarrow Q$ is the **transition function**.
  4. $q_0 \in Q$ is the **start state**, and
  5. $F \subseteq Q$ is the **set of accept states**.
- The transition function tells us if we are in a given state reading a given symbol what is the next state to go to.