NFAs and Myhill-Nerode

CS154
Chris Pollett
Feb. 22, 2006.

Outline

- Bonus Questions
- Equivalence with Finite Automata
- Myhill-Nerode Theorem.

Bonus Questions

- These questions are open to anybody.
- I will only accept solutions up to the class day after the second midterm.
- The points you receive from doing them can be added to your midterm score.
- You can do either problem or both. Your midterm score can be raised to a maximum of 15pts.
- To receive points on the bonus problems you must come see me during my office hours or after class and demo your code.
- I will ask you questions and review your code to determine how much your code is worth.

Bonus Question I (page1)

Problem 1 (2pts). Write a regular expression to NFA program in Java called rex2nfa. This should be run from the command line with a line like:

java rex2nfa *regular_expression*

Here a *regular_expression* is built up out of: the lower case alphabet symbols: a,b,c,…,z; 0 -empty set; E -empty string; (rex1.rex2) -- concatenation; (rex1 U rex2) --union; and (rex1)* -- star.

The output should be a sequence of rows of the form

state; symbol > state

Bonus Question I (page2)

- state - of the form A followed by some string of digits (for instance, A02345) or R followed by some string of digits (for example, R99).
- The former are supposed to accept state the latter are reject states.
- Symbol is supposed to be either an alphabet symbol or E for the empty string.
- The first state listed in the first row output is supposed to be the start state.

Bonus Question II (page1)

Problem 2(3pts). Write an NFA to DFA program in Java called nfa2dfa. This should be run from the command line with a line like:

java nfa2dfa *filename string*

Here filename is the name of a file containing lines of NFA instructions in the format of Problem1; string is an string that the output DFA will then try to scan and either accept or reject.

Your program should output a sequence of rows for the DFA, then skip a line and output accept or reject based on whether the original NFA would have accepted or rejected that string.

Bonus Question II (page2)

The format of the output rows should be:

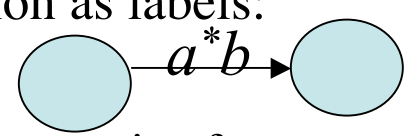
seq_of_states1; symbol > seq_of_states2

By a sequence of states, I mean a comma separated list of states where states are in the format of Problem 1.

Again, the first row should contain the start state of the DFA. I only want you to output rows which are reachable from this start state.

Proof that regular implies the language of some regular expression

- We will again split the proof into two parts:
  - We first define a new kind of finite automata called a generalized nondeterministic finite automata (GNFA) and show how to convert any DFA into a GNFA.
  - Then we show how to convert any GNFA into a regular expression.
- To begin we define a GNFA to be an NFA where we allow transition arrows to have any regular expression as labels:



- The transition function δ now takes a pair of states q,r and outputs a regular expression, R. The intended meaning is in state q reading a substring of the input of form R we transition to state r.

Converting DFAs to GNFAs

- We will be interested in GNFAs that have the following special form:
  - The start state has transition arrows to every other state but no arrows coming in from other states.
  - There is a single accept state, and it has arrows coming in from every other state but no arrows going to any other state.
  - Except for the start and accepts state, one arrow goes from from every state to every other state and also from each state to itself.
- To convert a DFA into a GNFA, we add a new start state with and ε arrow to the old start state and a new accept state with ε arrows from the old accept states.
- If any arrows have multiple labels (or if we have two or more arrows between the same two states) we replace each with a single label whose label is the union labels of the these arrows.
- Finally, we add arrows with labels $\emptyset$ between states which had no labels so as to satisfy the remaining conditions of our special form.

Converting GNFAs to Regular expressions

- Our conversion above gives a GNFA with k >= 2 states.
- If k > 2, we will construct an equivalent GNFA with k-1 states.
- To do this we pick some state q_{rip} other than the start or accept state, and we will rip it out of the machine.
- To compensate for the loss of this state, for any pair of states q_i, q_j. in this new machine we replace $\delta(q_i, q_j)$ with:

  $\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4)$

  where $\delta(q_i, q_{rip}) = R_1$; $\delta(q_{rip}, q_{rip}) = R_2$; $\delta(q_{rip}, q_j) = R_3$; $\delta(q_i, q_j) = R_4$
- This machine will be equivalent to the old machine.
- Further, by repeatedly ripping out states in this fashion we can get down to the 2-state machine with just a regular expression on the single transition between these two states.
- This regular expression will be equivalent to the original NFA.

Making DFAs as Small as Possible

- We have given a process for going from regular expressions (a widely used language for pattern matching) to DFAs.
- We would now like to optimize out DFAs and make them as small as possible.
- The Myhill-Nerode Theorem allows us to do this.

Pairwise Distinguishability

Definition: (1) We say two strings x,y are *pairwise distinguishable* by L, if some string z exists such that exactly one of xz or yz is in L. (2) We say two strings x,y are *pairwise indistinguishable* by L, if all strings z both of xz and yz is in L or both are not in L.

Fact: Pairwise Indistinguishability is an equivalence relation on strings.

Definition: Let the *index of L* be the number of equivalence classes of L with respect to pairwise indistinguishability.

Intuition

- Suppose we have a DFA for some regular language. We pick two states in this DFA, say q1 and q2 and we ask whether we could collapse them into one state or not.
- If it is the case that given any string z, that when we are state q1 when we begin reading z we do exactly the same as far as accepting/rejecting as we would do if we were in state q2 when we began reading a z, then we could collapse these states into one.
- Call two states q1 and q2 *distinguishable* if there is some string z which such that beginning in state q1 reading a z we reject; whereas, in state q2 reading a z we accept or vice-versa.
- Similarly, we can define state *indistinguishability*. This is also an equivalence relation with equivalence classes contained in those of language indistinguishability.
- So indistinguishability is measuring whether or not we can collapse states.

Myhill-Nerode Theorem (start)

- A language is regular iff it is of finite index.

Proof: Give any DFA for a language L, state indistinguishability for this DFA will have more equivalence classes then language indistinguishability for L. So if the number of language indistinguishable equivalence classes is not finite, the DFA can't have a finite number of states giving a contradiction.