joshuacamacho / cs311-project-1

Branch: master ▾ cs311-project-1 / src / cs311 / project / pkg1 / Dfsa.java Find file Copy path

joshuacamacho cleanup 33b63cd 11 minutes ago

1 contributor

116 lines (104 sloc) 3.48 KB

```
1  package cs311.project.pkg1;
2
3  import java.util.ArrayList;
4  import java.util.HashMap;
5  import java.util.List;
6  import java.util.Map;
7
8  /**
9   *
10  * @author Joshua Camacho
11  */
12 public class Dfsa {
13     private int num_states;
14     private List final_states;
15     private Map alphabet;
16     private Map delta;
17     private int currentState;
18
19     Dfsa(){
20         num_states = 0;
21         final_states = new ArrayList();
22         alphabet = new HashMap();
23         delta = new HashMap();
24         currentState=0;
25     }
26
27     public void setFinalStates(String final_states){
28         String [] stringArr = final_states.split(" ");
29         for(int i=0; i<stringArr.length; i++){
30             this.final_states.add( Integer.parseInt(stringArr[i]) );
31         }
32     }
33
34     public void setNumStates(String num){
35         num_states = Integer.parseInt(num);
36     }
37
38     public void setAlphabet(String alphabet){
39         alphabet = alphabet.replaceAll(" ", "");
40         for(int i=0; i<alphabet.length(); i++){
41             this.alphabet.put( alphabet.charAt(i), i);
42         }
43     }
44
45
46     //Input letter from alphabet
47     //Return numerical representation of that letter
48     public int getLetterValue(char letter){
49         if(alphabet.containsKey(letter))
50         return (int)this.alphabet.get(letter);
51         else System.out.println("alphabet didnt have "+letter);
52         return 0;
53     }
54
```

```
    // Gives a value for delta
    //  A sequence of transitions of the form (p a q)
    // p current state
    // a input
    // q next state
    public void deltaPush(String set){
        if(set.charAt(0)=='('){
            set =  set.substring(1);
        }
        if(set.charAt(set.length()-1)==')'){
            set = set.substring(0, set.length()-1);
        }
        //allow for easier state transition definitions
        //a can be multiple symbols, setting multiple rules
        String [] items = set.split(" ");
        if(items[1].length()>1){
            String state = items[0];
            int nextState = Integer.parseInt(items[2]);
            String inputs = items[1];
            for(int i=0; i<inputs.length(); i++){
                String temp = state + " "+inputs.charAt(i);
                delta.put(temp, nextState);
            }
        }else{
            String stateInput = items[0] + " " + items[1];
            int nextState = Integer.parseInt(items[2]);
            delta.put(stateInput, nextState);
        }
    }

    public String evalString(String input){
        currentState=0;
        for(int i=0; i<input.length(); i++){
            String stateInput = Integer.toString(currentState)+" "+ input.charAt(i);
            if(delta.containsKey(stateInput)){
                currentState = (int)delta.get(stateInput);
            }else{
                //if a deadend was reached return false
                //this allows partial language definitions
                return "Reject";
            }
        }
        if(final_states.contains(currentState)){
            return "Accept";
        } else {
            return "Reject";
        }
    }

    public void dump(){
        System.out.println("(1) number of states: "+num_states);
        System.out.print("(2) final states: ");
        for(int i=0; i<final_states.size(); i++){
            System.out.print(final_states.get(i)+ " ");
        }
        System.out.print("\n(3) alphabet: ");
        alphabet.forEach((k,v)->System.out.print(k+", "));
        System.out.println("\n(4) transitions");
        delta.forEach((k,v)->System.out.println("(" + k + " " + v+")"));
    }
}
```