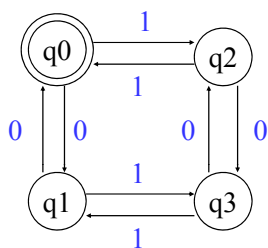


Lexical Analysis – Part II

EECS 483 – Lecture 3
University of Michigan
Monday, January 14, 2008

Class Problem From Last Time

Is this a DFA or NFA?
What strings does it recognize?



Lex Notes

- ❖ 64-bit machine compilation of flex file

- » gcc -m64 lex.yy.c -fl

- ❖ Questions from last time

- » [\t]+, there is a space here

- So this matches all white space characters except new lines

- » Flex can detect spaces if you want it to

- » The period operator, . , does match all characters except newline

Reading

- ❖ Ch 2 – Red Dragon

- » Just skim this

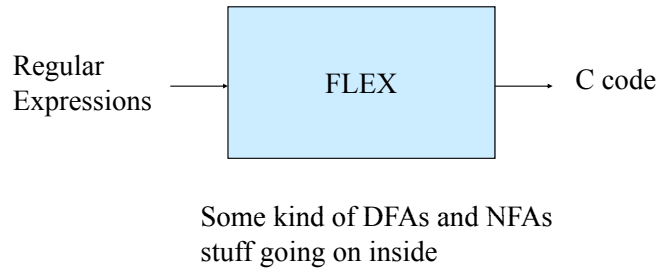
- » High-level overview of compiler, which could be useful

- ❖ Ch 3 – Red Dragon

- » Read carefully, more closely follows lecture

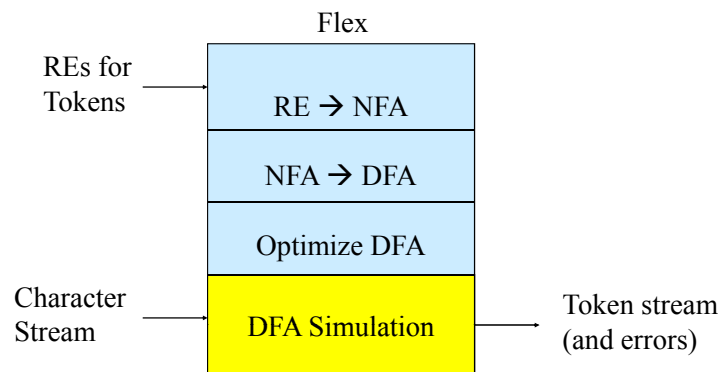
- » Go over examples

How Does Lex Work?



- 4 -

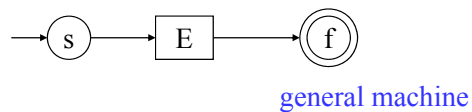
How Does Lex Work?



- 5 -

Regular Expression to NFA

- ❖ Its possible to construct an NFA from a regular expression
- ❖ Thompson's construction algorithm
 - » Build the NFA inductively
 - » Define rules for each base RE
 - » Combine for more complex RE's

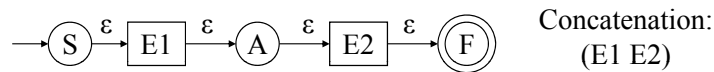
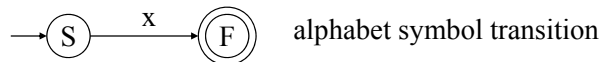
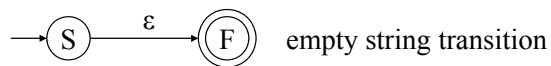


winter 2008

- 6 -

scott ed. trev

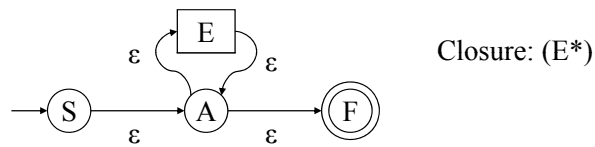
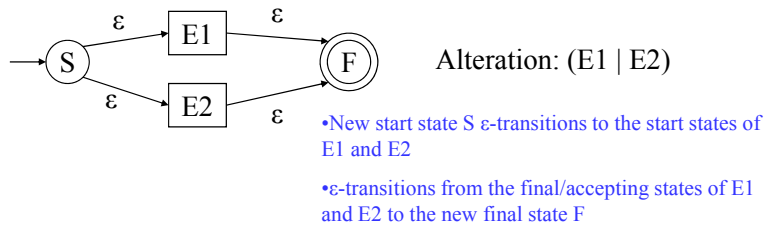
Thompson Construction



- New start state S ϵ -transition to the start state of E1
- ϵ -transition from final/accepting state of E1 to A, ϵ -transition from A to start state of E2
- ϵ -transitions from the final/accepting state E2 to the new final state F

- 7 -

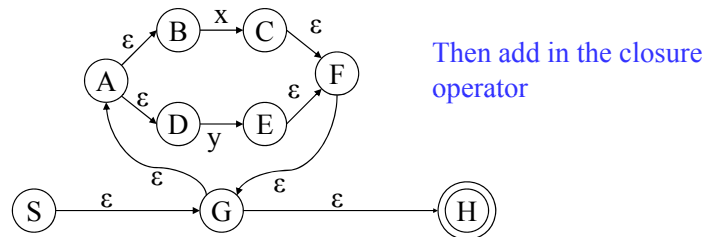
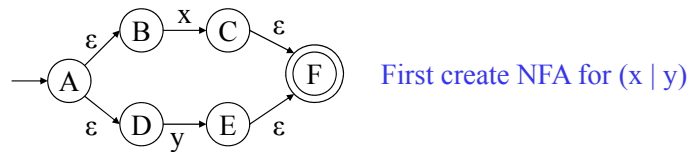
Thompson Construction - Continued



- 8 -

Thompson Construction - Example

Develop an NFA for the RE: $(x \mid y)^*$



- 9 -

Class Problem

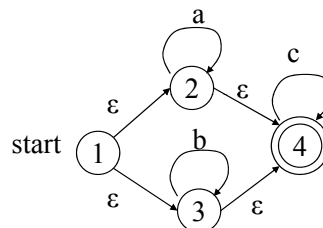
Develop an NFA for the RE: $(\backslash+? | -?) d+$

- 10 -

NFA to DFA

- ❖ Remove the non-determinism
- ❖ 2 problems
 - » States with multiple outgoing edges due to same input
 - » ϵ transitions

$(a^* | b^*) c^*$



winter 2008

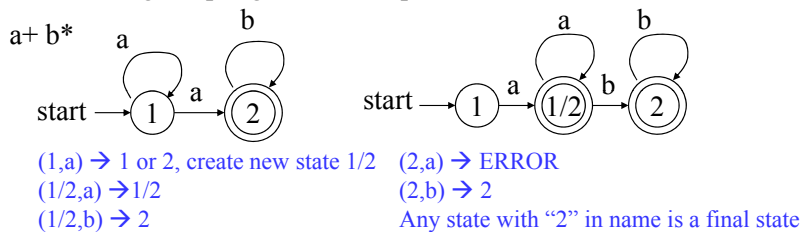
- 11 -

scott ed. trev

NFA to DFA (2)

❖ Problem 1: Multiple transitions

- » Solve by subset construction
- » Build new DFA based upon the power set of states on the NFA
- » Move (S,a) is relabeled to target a new state whenever single input goes to multiple states



winter 2008

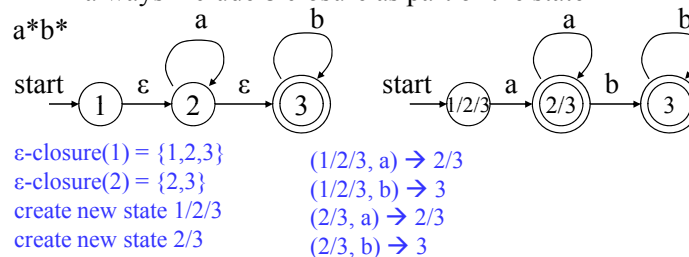
- 12 -

scott ed. trev

NFA to DFA (3)

❖ Problem 2: ϵ transitions

- » Any state reachable by an ϵ transition is "part of the state"
- » ϵ -closure - Any state reachable from S by ϵ transitions is in the ϵ -closure; treat ϵ -closure as 1 big state, always include ϵ -closure as part of the state

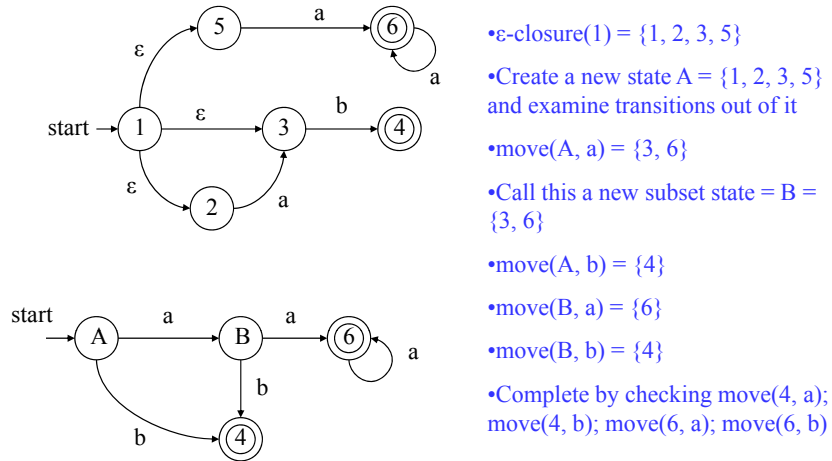


winter 2008

- 13 -

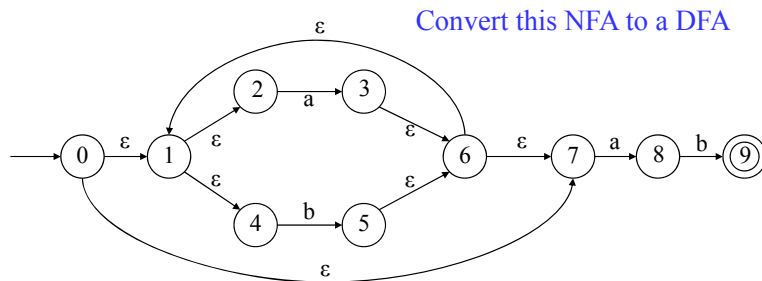
scott ed. trev

NFA to DFA - Example



- 14 -

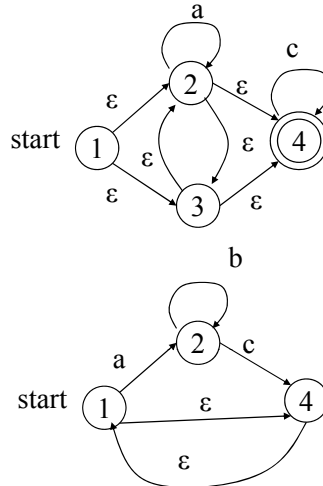
Class Problem



- 15 -

NFA to DFA Optimizations

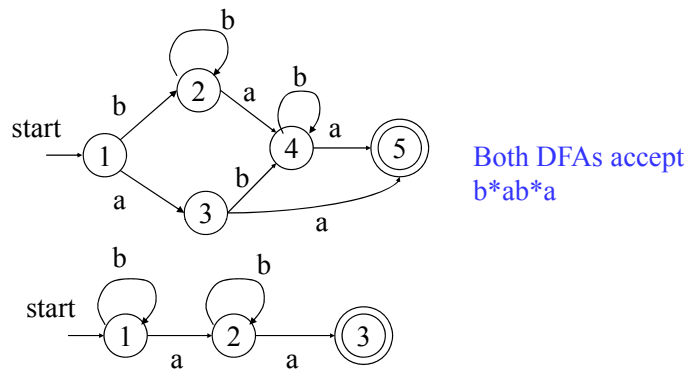
- ❖ Prior to NFA to DFA conversion:
- ❖ Empty cycle removal
 - » Combine nodes that comprise cycle
 - » Combine 2 and 3
- ❖ Empty transition removal
 - » Remove state 4, change transition 2-4 to 2-1



- 16 -

State Minimization

- ❖ Resulting DFA can be quite large
 - » Contains redundant or equivalent states



winter 2008

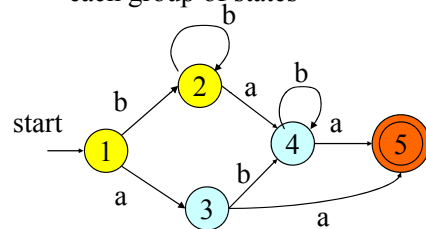
- 17 -

scott ed. trev

State Minimization (2)

❖ Idea – find groups of equivalent states and merge them

- » All transitions from states in group G1 go to states in another group G2
- » Construct minimized DFA such that there is 1 state for each group of states



Basic strategy: identify distinguishing transitions

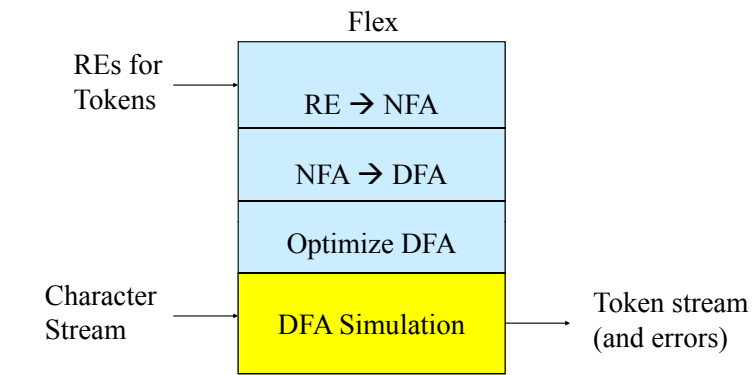
winter 2008

- 18 -

scott ed. trev

Putting It All Together

Remaining issues: how to Simulate, multiple REs, producing a token stream, longest match, rule priority



- 19 -

Simulating the DFA

- * Straight-forward translation of DFA to C program
- * Transitions from each state/input can be represented as table
 - Table lookup tells where to go based on current state/input

```
trans_table[NSTATES][NINPUTS];
accept_states[NSTATES];
state = INITIAL;

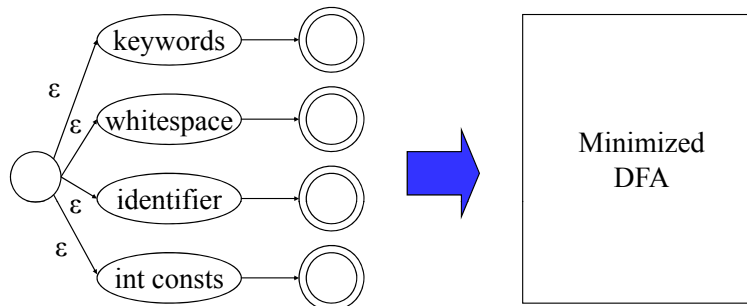
while (state != ERROR) {
    c = input.read();
    if (c == EOF) break;
    state = trans_table[state][c];
}
return accept_states[state];
```

Not quite
this simple
but close!

- 20 -

Handling Multiple REs

Combine the NFAs of all the regular expressions
into a single NFA



- 21 -

Remaining Issues

- ❖ Token stream at output
 - » Associate tokens with final states
 - » Output corresponding token when reach final state
- ❖ Longest match
 - » When in a final state, look if there is a further transition. If no, return the token for the current final state
- ❖ Rule priority
 - » Same longest matching token when there is a final state corresponding to multiple tokens
 - » Associate that final state to the token with highest priority

Project 1

- ❖ P1 handout available under projects link on course webpage by the weekend of January 26th
 - » Base file has a bunch of links, so make sure you get everything
- ❖ Your job is to write a lexical analyzer and parser for a language called C - -
 - » Flex and bison will be used to construct these
 - We'll talk about bison next week
 - Can start on the flex part immediately
 - » You will produce a stylized output explained in the Spec
 - » Detect various simple errors
 - » Due Wednesday, March 5th

C - - (A few of the Highlights)

❖ Subset of C

- » Allowed keywords: char, else, extern, for, while, if, int, return, void
- » So, no floating point, structs, unions, switch, continue, break, and a bunch of other stuff
- » All C punctuation/operators are supported (including ++) with the exception of '?' operators
- » No include files – manually declare libc/libm functions with externs
- » Only 1 level of pointers, ie int *x, not int **x

Project Grading

❖ You'll turn in 2 files

- » unquename.l, unquename.y

❖ Details of grading still to be worked out

- » But, as a rough estimate
 - Grade = explanation * (features + correctness)
 - Correctness: do you pass the testcases, we provide some to you, but not all
 - Features: how much of the spec did you implement
 - Explanation: Interview, do you understand the concepts, can you explain the source code

Doing Your Own Work

- ❖ Each person should work independently on Project 1
 - » You are encouraged to help each other with flex/bison syntax, operation, etc.
 - » You can discuss the project, corner cases, etc.
 - » But the code should be yours
- ❖ We will not police this
 - » But, it will be obvious in the interview who did not write the code or did not understand what they wrote