CS480
Translators

More Lexical Analysis

Chap. 3

Oregon State University

Odds and Ends…

- Demos: Show up for your appointment!!!
- Milestone 2
  - You can make your own design decisions
- KISS☺
- Itty-Bitty Teaching Language...
- http://classes.engr.oregonstate.edu/eecs/winter2014/cs480/assignments/IBTL-Grammar.pdf

Milestone 2

Job of a Tokenizer…

```
if( peek == '\n' ) line = line + 1;
```

↓

Lexical Analyzer

↓

⟨**if**⟩ ⟨(⟩ ⟨**id**, "peek"⟩ ⟨**eq**⟩ ⟨**const**, '\n'⟩ ⟨)⟩
⟨**id**, "line"⟩ ⟨**assign**⟩ ⟨**id**, "line"⟩ ⟨+⟩ ⟨**num**, 1⟩ ⟨;⟩

↓

Patterns/Regular Expressions

- Defining tokens
  - if while assign — Keywords
  - ( ) = > — operators
  - [a-zA-Z][a-zA-Z0-9]* — ids — 0 or more
  - [0-9]+ — positive ints — 1 or more
  - [0-9]+(.[0-9]+)?(E[+-]?[0-9]+)? — 0 or 1
  - [\t\n]+ — whitespace

Languages/Regular Expressions

OPERATION	DEFINITION AND NOTATION
Union of L and M	$L \cup M = \{s \mid s \text{ is in } L \text{ or } s \text{ is in } M\}$
Concatenation of L and M	$LM = \{st \mid s \text{ is in } L \text{ and } t \text{ is in } M\}$
Kleene closure of L	$L^* = \cup_{i=0}^{\infty} L^i$
Positive closure of L	$L^+ = \cup_{i=1}^{\infty} L^i$

- Every symbol of Σ is a regular expression
- ε is a regular expression
- if **r1** and **r2** are regular expressions, so are
  (**r1**) **r1r2** **r1** | **r2** **r1***
- Nothing else is a regular expression.

Languages/Regular Expressions

- ***Given an alphabet* Σ, *the regular expressions over* Σ *and their* corresponding *regular languages are***
  a) ε, the empty string, denotes the language {ε}.
  b) for each a in Σ, a denotes { a } --- a language with one string.
  c) if R denotes L_R and S denotes L_S then R | S denotes the language L_R □ L_S , i.e, { x | x □ L_R or x □ L_S }.
  d) if R denotes L_R and S denotes L_S then RS denotes the language $L_R L_S$, that is, { xy | x □ L_R and y □ L_S }.
  e) if R denotes L_R then R* denotes the language L_R* where L* is the union of all L^i (i=0,...,∞) and L^i is just {$x_1 x_2 \ldots x_i$ | x_1 □ L, ..., x_i □ L}.
  f) if R denotes L_R then (R) denotes the same language L_R.

Implementing Regular Expressions

- Build finite automata for all patterns
- Connect start states for NDFA
- Simplify to make DFA
- Algorithm:
  - When asked for token, start in combined state
  - Read characters, advancing state, until cannot advance further
  - If needed, push back last character(s)
  - Return token associate with last state

Oregon State University

Finite State Automata

doesn't save state

- A recognizer for a language is a program that takes a string x as an input and answers "yes" if x is a sentence of the language and "no" otherwise.
- One can compile any regular expression into a recognizer by constructing a generalized transition diagram called a finite automaton.
- A finite automaton can be non-deterministic or deterministic.
- Both automata are capable of recognizing regular expressions.

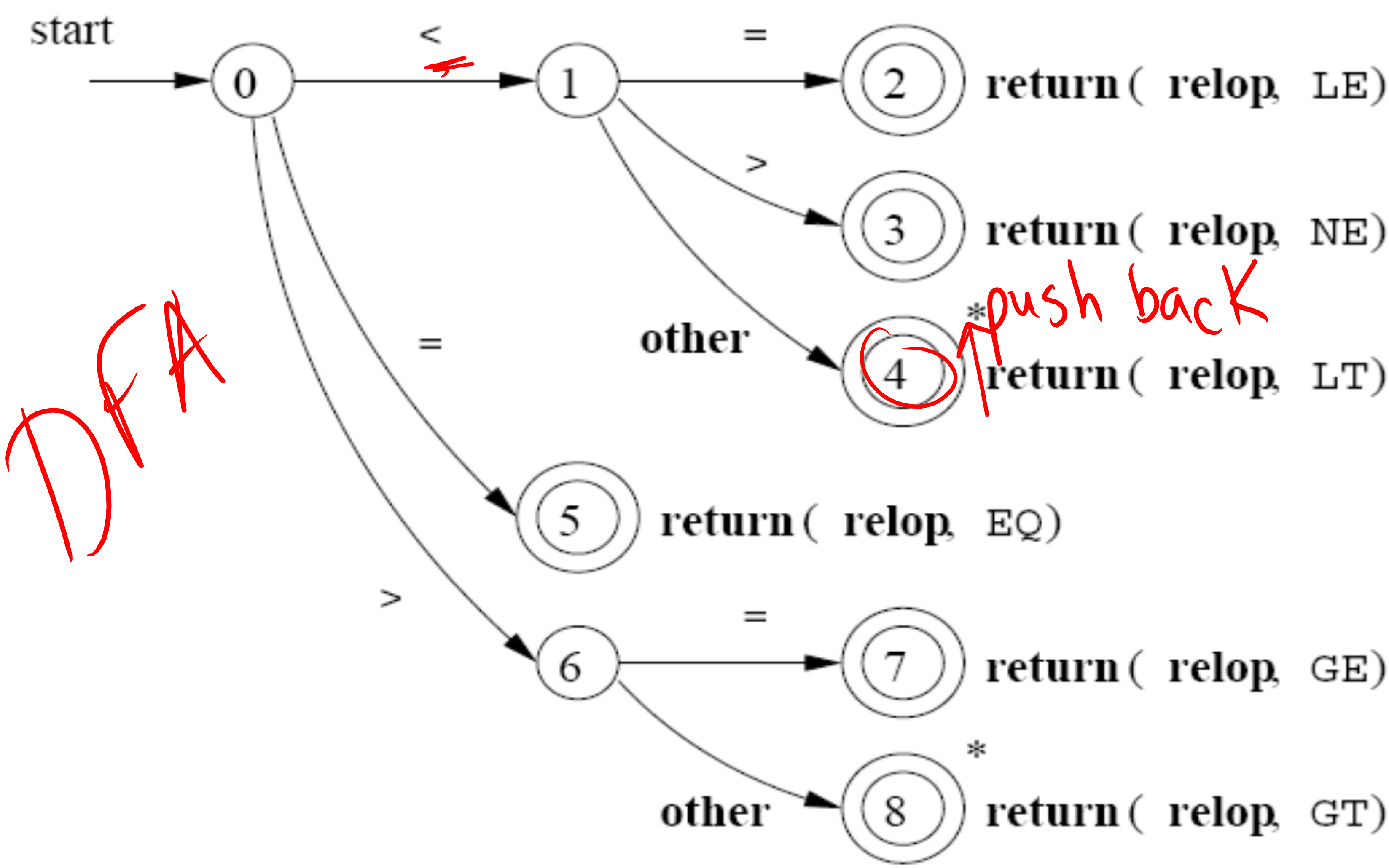
Transition Diagram

- Flowchart with **states** and **edges;** each edge is labeled with characters; certain subset of states are marked as "**final states**"
- Transition from state to state proceeds along edges according to the next **input character**
- Every string that ends up at a **final state** is accepted
- If get "stuck", there is no transition for a given character, it is an **error**

and you are not in an accept state

OSU Oregon State University

$relop \rightarrow\ <\ |\ >\ |\ <=\ |\ >=\ |\ =\ |\ <>$



Oregon State University

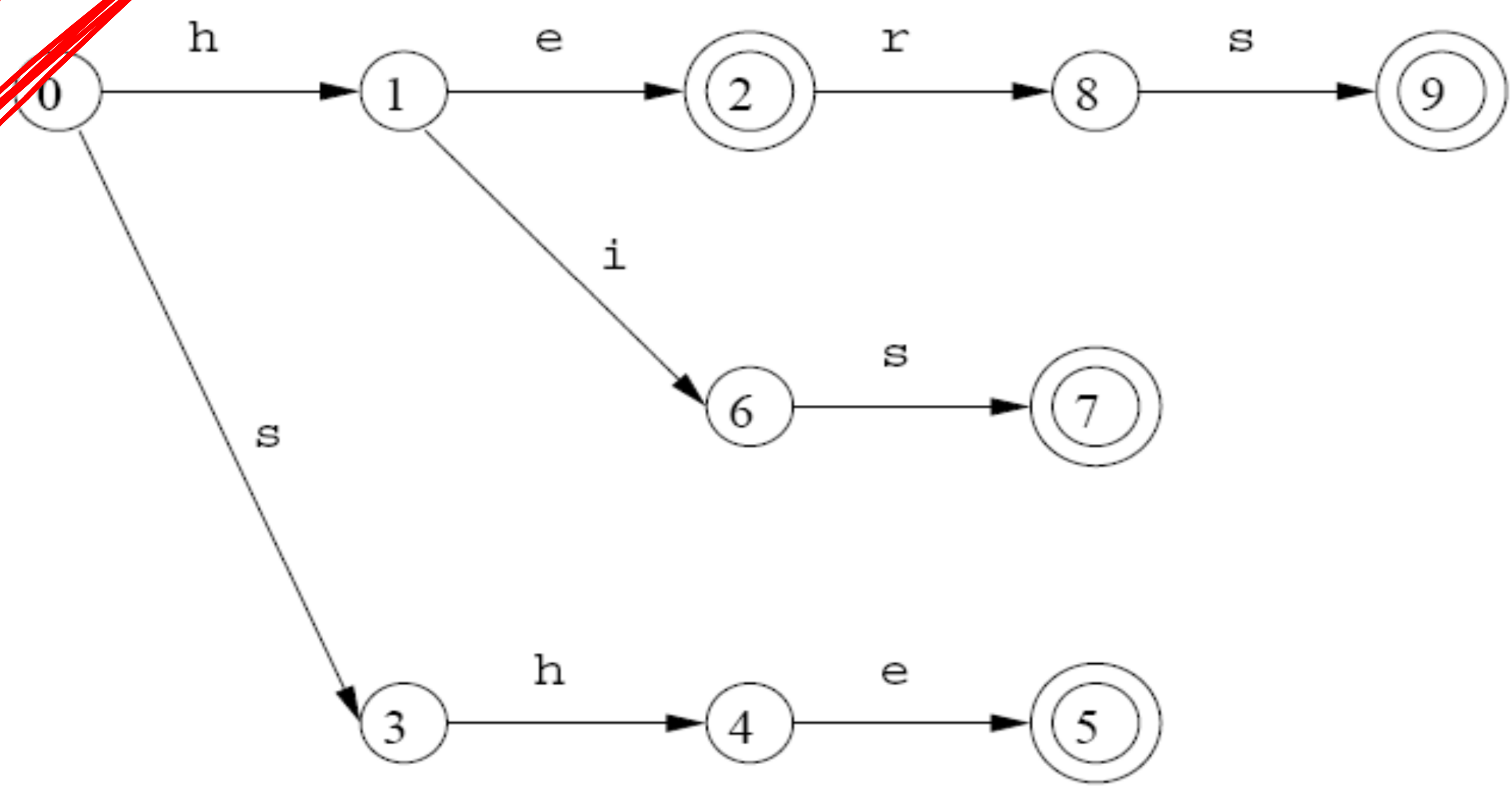
return

relop → < | > | <= | >= | = | <>

```
TOKEN getRelop()
{
    TOKEN retToken = new(RELOP);
    while(1) { /* repeat character processing until a return
                  or failure occurs */
        switch(state) {
            case 0: c = nextChar();
                    if ( c == '<' ) state = 1;
                    else if ( c == '=' ) state = 5;
                    else if ( c == '>' ) state = 6;
                    else fail(); /* lexeme is not a relop */
                    break;
            case 1: ...
            ...
            case 8: retract(); ← push back
                    retToken.attribute = GT;
                    return(retToken);
        }
    }
}
```

Goal!

Collection of Keywords



NFA vs. DFA

- **Nondeterministic Finite Automata (NFA) :**
  - ε can label edges (these edges are called ε**-transitions**)
  - some character can label 2 or more edges out of the same state w/ same char.
- **Deterministic Finite Automata (DFA) :**
  - no edges are labeled with ε
  - each character can label at most **one** edge out of the same state
- **NFA** and **DFA** accepts string **x** if there exists a path from the start state to a final state labeled with characters in **x**

(a|b)*abb

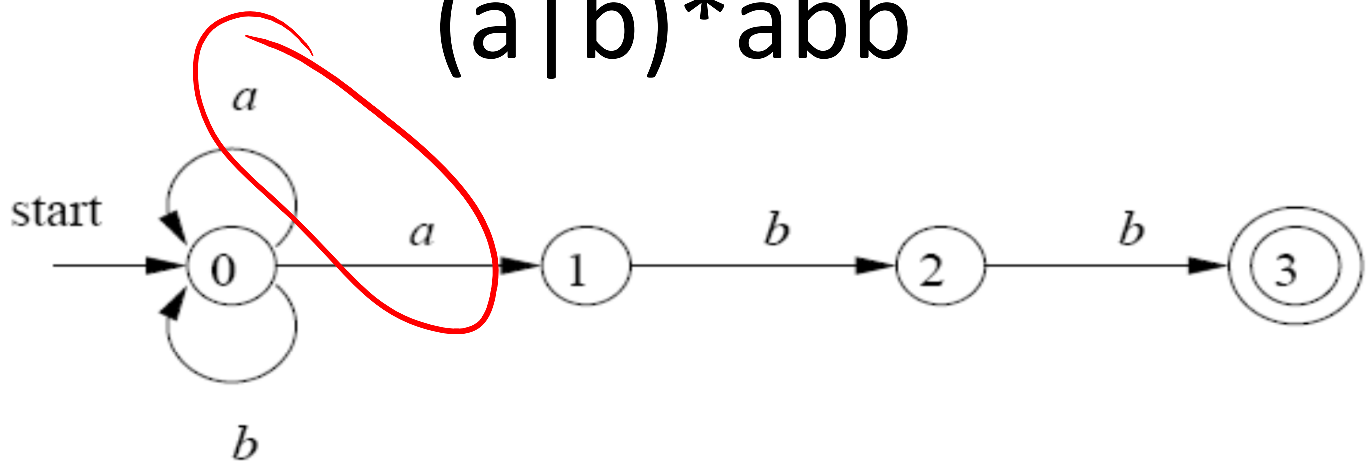


Figure 3.24: A nondeterministic finite automaton

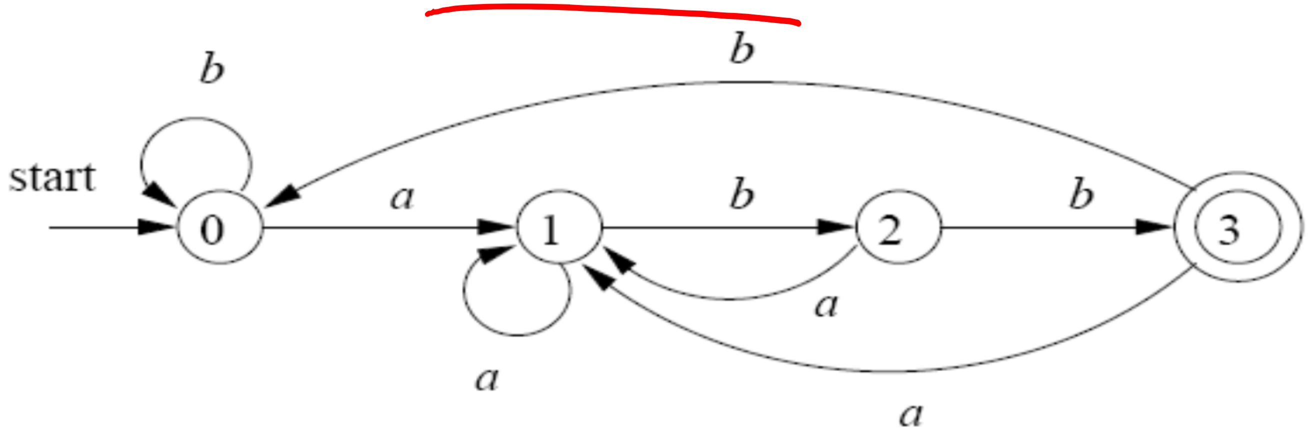


Figure 3.28: DFA accepting $(\mathbf{a}|\mathbf{b})^*\mathbf{abb}$

Transition Tables

- For NFA, each entry is a set of states:

STATE	a	b
0	{0,1}	{0}
1		{2}
2		{3}
3	-	-

- For DFA, each entry is a unique state:

STATE	a	b
0	1	0
1	1	2
2	1	3
3	1	0

Oregon State University

NFA to DFA

OPERATION	DESCRIPTION
$\epsilon\text{-}closure(s)$	Set of NFA states reachable from NFA state s on ϵ-transitions alone.
$\epsilon\text{-}closure(T)$	Set of NFA states reachable from some NFA state s in set T on ϵ-transitions alone; $= \cup_{s \text{ in } T}\ \epsilon\text{-}closure(s)$.
$move(T, a)$	Set of NFA states to which there is a transition on input symbol a from some state s in T.

while (there is an unmarked state T in $Dstates$) {
    mark T;
    for (each input symbol a) {
        $U = \epsilon\text{-}closure(move(T, a))$;
        if (U is not in $Dstates$)
            add U as an unmarked state to $Dstates$;
        $Dtran[T, a] = U$;
    }
}

Quiz #3

- **Tokenize** the following C statement:

```
float limitedSquare(x) float x; {
  /*returns x-squared, but never more than 100*/
  return (x<=-10.0||x>=10.0)?100:x*x;
}
```

- Given Σ = {a,b}, provide regular expressions for languages below:
  - all strings beginning and ending in *a*
  - all strings of *a's* and *b's* of even length
  - all strings with an odd number of *a's*
  - string of zero or more *a's* followed by same number of *b's*
- Give the FSA for each (DFA or NFA)…