CSE 401 - Compilers
Section 2

1/24/2013
12:30 - MEB 238
1:30 - EE 037

Regex Exercise

strings of 0s and 1s such that every sequence of two 1s must be preceded by at least two consecutive 0s and followed by at least three

Anyone think about this more?
Have a regex?
Have a DFA/NFA?
Think it's impossible?

Regex Exercise

strings of 0s and 1s such that every sequence of two 1s must be preceded by at least two consecutive 0s and followed by at least three

A Key Observation: The validity of the next character depends on at most the four preceding characters

Regex Exercise

strings of 0s and 1s such that every sequence of two 1s must be preceded by at least two consecutive 0s and followed by at least three

A Key Observation: The validity of the next character depends on at most the four preceding characters

Suggests that we can build a DFA

- States encode last characters seen

Regex Exercise

strings of 0s and 1s such that every sequence of two 1s must be preceded by at least two consecutive 0s and followed by at least three

seen	can see
^	0,1
^0	0,1
00	0,1
10	0,1
001	0,1

seen	can see
^1	0
^01	0
101	0
0011	0
110	0
1100	0

Regex Exercise

strings of 0s and 1s such that every sequence of two 1s must be preceded by at least two consecutive 0s and followed by at least three

Regular languages: RE <-> NFA <-> DFA

We've seen RE -> NFA -> DFA
DFA -> NFA is trivial
NFA -> RE can be done algorithmically via...

Generalized Nondeterministic Finite Automaton (GNFA)

An NFA but:

- One start state
- One accept state
- REs instead of single characters on its edges

NFA -> GNFA:

- Add super-start and super-accept states

GNFA -> RE:

- Remove states one at a time, fixing edges

Regex Exercise

strings of 0s and 1s such that every sequence of two 1s must be preceded by at least two consecutive 0s and followed by at least three

((1?0)*(1?(00110)*00)?)*1?

Believe me? Questions?

Project Clarifications

Longest match examples:

- "true;" -> TRUE SEMICOLON
- "truethat;" -> ID(truethat) SEMICOLON
- "verytrue;" -> ID(verytrue) SEMICOLON
- "true that;" -> TRUE ID(that) SEMICOLON

JFLEX tries to all match REs at once

Another case:

- "45true" -> INT(45) TRUE

Project Questions?

Parser Ambiguities

```
expr   ::= expr + expr | expr - expr |
           expr * expr | expr / expr |
           int
int    ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
```

1. Find an ambiguous parse
2. Fix the grammar
3. Support parenthesis

Parser Ambiguities

```
expr   ::= expr + term | expr - term | term
term   ::= term * factor | term / factor | factor
factor ::= int | ( expr )
int    ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
```

Definition Review

Sentential Form: α in S =>* α

`S =>* ( term * factor ) =>* (2 * 3)`

Handle: A position in α and a production that we can "undo"

`term ::= term * factor` at position 4

Shift-Reduce Exercise

```
expr   ::= expr + term | expr - term | term
term   ::= term * factor | term / factor | factor
factor ::= int | ( expr )
int    ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
```

Stack	Input	Action (shift or reduce)
$	1 + 2 * 3$	shift
...	...	...
$S	$	accept

Regular or Context-Free?

1. $L = \{0^n 1^n \mid n >= 0\}$
2. $L = \{0^n 1^m \mid n >= 0, m > n\}$
3. $L = \{w \mid \#_0(w) == \#_1(w)\}$
4. $L = \{w \mid \#_{01}(w) == \#_{10}(w)\}$
5. Balanced parenthesis?

Generating regex / DFA / grammar?

Regular or Context-Free?

1. $L = \{0^n 1^n \mid n >= 0\}$ CF
2. $L = \{0^n 1^m \mid n >= 0, m > n\}$ CF
3. $L = \{w \mid \#_0(w) == \#_1(w)\}$ CF
4. $L = \{w \mid \#_{01}(w) == \#_{10}(w)\}$ R!
5. Balanced parenthesis? CF

Questions?

Go get a job!