CS3200 - Scanning & Parsing

Announcements

flex hw - due next Sat.
Next Step: Parsing, which is the

- Menhaden in the

- Flex

- DFAs + NFAs

- Scenarios and regular expressions

Last week: Ch2 of book
\[ (\text{O01}) (\text{O11}) \]

an odd length

\[ \text{Ex: } \exists m \text{ a string with } 0 \text{ and has } \]

\[ (\text{O01}) \]

ends with a 0

\[ \text{Ex: Give the regular expression for } \]


Are you \equiv 0 \mod 3? 

Ex: 3 symbol alphabet: \{0, 1, 2\}
Given: A number of 15 is accepted if not 0 or 5.

Ex: Strings of 0s and 1s.
NFAs : DFAs w/ ambiguity

Thm : NFAs ≡ DFAs are equivalent; both accept reg. languages.
Converting NFA to DFA (9.5.7)
However, NFAs are easy to convert from regex and DFA. Unfortunately, there's not a great way to compare NFAs and DFAs.

But code can be used to emulate a DFA. Usually easier to specify a regular expression. Generally scanners...
(Base case)

Simple NFA:

Given a regular expression, we can construct an NFA.

(Consistency in NFA (p. 57, 36))
Q1:

Q2:

Q3:

Concatenation:

3 operators
and Kleene closure (A).

(d) Kleene closure
Result:

(see p. 57-58)

Next: Convert to DFA.
Can easily find all the equivalence classes and minimize.

Not minimal.

This DFA is a bit redundant.

Note:
Process to minimize (p. 59)
Given DFA, generate case statements.

Now:

\[
\begin{aligned}
B: & \\
A: & \text{case } \text{char} = \text{c} \\
& \text{case } \text{state} = \text{c} \\
& \text{curr-char} = c \\
& \text{state} = A \\
& \text{repeat} \\
& \text{read char} \\
& \text{state} \text{ is:} \\
\end{aligned}
\]
This is what they do to you?

express in
language

An example is the generation of a DFA.

So, the result is...
Ex: On [ ]

expression, do not have.

expression, which is regular.

Sometimes, this needs a type

Ex: [ ] 0's and 1's.

Ex: [ ] unique on equal number

Certain languages are not regular.

Limitations of Regular Expressions
Regular expressions can’t quite do this.

\[ \text{expr} \to \text{id} \text{ | number | expr op expr} \]

E.g.

\[ \text{expr} \to \text{expr op expr} \]

\[ \text{expr} \to \text{id} \text{ | number} \]

Need to "nest" expressions.

```
why do we need this?
```
A set of productions - 
A set of symbols S (a non-terminal)
A set of non-terminals N
A set of terminals T

Disjoint in terms of productions (called Backus-Naur form or BNF)

Context-free languages
Is cool el?:

- Yes: S = QT, OQT
- No:

\[ S = QT \]

\[ S \rightarrow QT \]
5 ≤ 50 ≤ 10 ≤ 100 ≤ 150 ≤ 150 ≤ 0 ≤ 010 ≤ 0 ≤ 0

Q: Is this enough?

Ex: 3126 is not a palindrome number.

Ex: 31213 is equal 31213 number.
Expression grammars: Simple calculator
Example: Show how rules can generalize 3 + 4 ≠ 5.
Parse Tree

\[
\frac{5 \times 3}{4 + 5}
\]
result. Expressions and then compute

Example:

Consider a simple calculator.

Example:

Formal. The computational tool is

Your: These are also tools to do this for

Pause:
Flux: a function for a Calculator?

```c
/*
 * Flux: a function for a Calculator
 *
 * Recognize tokens for the calculator and print them out
 */

enum yylexertype
{
    EOF = 264,
    ABS = 263,
    DIV = 262,
    MUL = 261,
    SUB = 260,
    ADD = 259,
    NUMBER = 258,
};

main(argc, argv)
{
    /*
     * Identify whitespace
     */
    return(yylex = (yylval = 0)? return NUMBER;
    return ABS,
    return DIV,
    return MUL,
    return SUB,
    return ADD;
    return;

    while(tok = yytext())
    {
        printf("%d\n", yylval);
        if(tok == NUMBER)
            printf("%d", yylval);
        printf("%d", tok);
    }
    int tok;
}
```
In Action:

```
264
258 = 45
263
259
258 = 34
262

Mystery character a
\ a / 34 + 45
$ a.out
$ cc latex.c -lft
$ latex this-4.1
```
Bison accepts these Bourne shell statements
Running:

Building:

c - 0 @ tbl-5.tab. lex, y, c - tbl

tbl-5.1 tbl-5.1 tbl-5.1 tbl-5.1 tbl-5.1 tbl-5.1 tbl-5.1 tbl-5.1 tbl-5.1

tax tbl-5.1 tbl-5.1 tbl-5.1 tbl-5.1 tbl-5.1 tbl-5.1 tbl-5.1 tbl-5.1 tbl-5.1

3 = 20 / 4 - 2
20 - 4 / 2
10 = 4 * 3 + 4
14 = 4 * 3 + 4
$ ./tbl-5

# part of the makefile
But only works on a particular grammar.

Essentially, this is a CFC.

Back to what Bison is.