CSCI_4230_PL_10_10_2018.md

Midterm Talk

scheme talk

- Please use a `let` expression

Exam review

Working through Pass1

Num1a

```
ab*
```

Num1b

```
a(b|c)*
```

Precidence

1. * = Kleene star
2. concat = things next to each other
3. | = alternation; this or that

Num1c - a then 1 or more b's

```
abb*
```

or

```
ab+
```

or

```
ab*b
```

Num1d - even number of chars

```
((a|b|c)(a|b|c))*
```

or

```
char -> a|b|c
evenString -> (char char)*
```

Num1e - even number of chars

```
((a|b|c)(a|b|c))*(a|b|c)
```

or

```
char -> a|b|c
evenString -> (char char)*char
```

or

```
(a|b|c)((a|b|c)(a|b|c))*
```

or

```
char -> a|b|c
evenString -> char(char char)*
```

NUMBER 2

Num2a - regexpr that starts with a letter or a "_"

```
( [a-zA-Z] | _ ) ( [a-zA-Z] | [0-9] | _ )*
```

NUMBER 3

BNF: terminal symbols, non-terminal symbols, concantenation

- alternation in BNF is just a way of saving space
- NO PARENTHESIS

- YES Recursion is allowed

EBNF: PARENTHESIS, Kleene Star (zero or more), optionals (? or [ ])

- can be described by BNF although using diff grammar

BNF example

- you can see recursion

```
S -> a B
B -> Bb | E        // E is the empty string
```

EBNF example

```
S -> ab*
```

or

```
S->a{b}
```

example of [ ]; "optionals"

```
S -> sign digits
sign -> + | - | E        // E is the empty string
digit -> 0|1|2|3|4|5|6|7|8|9
```

or with optional

```
S -> [+|-] digit
```