# Syntax

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2016

### 1<sup>st</sup> Midterm

Sep. 30 (Friday) 6:30PM – 7:45PM

010 Sparks Building

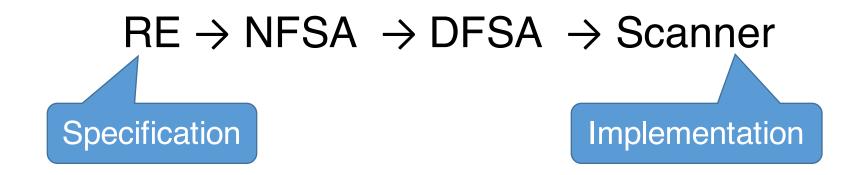
HW3 due next Wednesday NOON (**no late submission**)

1st Mideterm covers materials in HW1 to HW3

Next Wednesday: review of assignments and practice problems

Next Friday: evening exam (no lecture)

## Regular Expressions to Scanner



All strings with n 0's followed by n 1's,  $n \ge 1$ ?

Not a regular language!

## Context-Free Grammar (CFG)

#### A CFG consists of

- A set of terminals T (basic alphabet)
- A set of non-terminals N
- A start symbol S (a non-terminal)
- A set of production rules

$$A ::= \omega$$
nonterminal and nonterminal

## Context-Free Grammar (CFG)

#### **Terminals**

- Building block of CFG
- E.g., letters, digits, or tokens from scanner

#### Non-terminals

- Symbols defined by production rules
- E.g., if-stmt = IF LPAREN expr RPAREN stmt

terminal

## CFG Example

A pure  $\lambda$ -term is defined inductively as follows:

- Any variable x is a  $\lambda$ -term
- If e is a  $\lambda$ -term, so is  $\lambda x$ . e (abstraction)
- If  $e_1$ ,  $e_2$  are  $\lambda$ -terms, so is  $e_1e_2$  (application)

# CFG Example

In RE:  $var \rightarrow (a | b | ... | Z)^+$ 

## CFG Example

```
expr ::= id | number | - expr | (expr) | expr op expr op ::= + | - | * | / ...
```

#### Terminals Can be Tokens

```
expr := id \mid number \mid - expr \mid (expr) \mid expr op expr
```

where id number op are tokens recognized by the scanner

### **Build CFG**

All strings with n 0's followed by n 1's,  $n \ge 1$ ?

$$S ::= 0S1 \mid 01$$

### **Build CFG**

All strings with n 0's and n 1's,  $n \ge 0$ ?

$$S ::= 0S1S \mid 1S0S \mid \epsilon$$

## Context-Free Language

L(G): the language (set of strings) defined by G

#### **Definition:**

L(G) is the set of all terminal strings derivable from the start symbol of G

### Derivation

A sequence from start symbol, each step a nonterminal is replace by RHS of production

#### A derivation of "xyz"

```
var ⇒ letter var
```

 $\Rightarrow$  letter letter var  $\Rightarrow$  letter letter

 $\Rightarrow$  x letter letter  $\Rightarrow$  x y letter  $\Rightarrow$  x y z

A multi-step reduction

var ⇒\* xyz

#### RE and CFG

#### CFG is strictly more expressive than RE

- 1. For any RE R, there is CFG C, such that L(R) = L(C)
- 2. Exists CFG C, such that for all RE R,  $L(R) \neq L(C)$

- 1. Proof by induction.
- 2. A language with n 0's followed by n 1's,  $n \ge 1$  S  $\rightarrow$  0S1 | 01

## RE

## **CFG**

$\epsilon$	$S ::= \epsilon$
a	S := a
$R_1 R_2$	$S ::= R_1 R_2$
$R_1 I R_2$	$S ::= R_1 \mid R_2$
R*	$S ::= S R \mid \epsilon$

where  $R_1$ ,  $R_2$  and R are the equivalent nonterminal for  $R_1$ ,  $R_2$  and R in RE respectively