Programming Languages and Computation

Week 2: Context Free Grammars

* 1. Consider the following CFG G with start symbol R:

$$\begin{array}{rcl} R & \longrightarrow & XRX \mid S \\ S & \longrightarrow & aTb \mid bTa \\ T & \longrightarrow & XTX \mid X \mid \epsilon \\ X & \longrightarrow & a \mid b \end{array}$$

(a) What are the non-terminals of G?

(b) What are the terminals of G?

(c) Give three strings in $L(G)$.

(d) Give three strings not in $L(G)$.

(e) True or false: $T \to aba$.

(f) True or false: $T \to^* aba$.

(g) True or false: $T \to T$.

(h) True or false: $T \to^* T$.

(i) True or false: $XXX \to^* aba$.

(j) True or false: $X \to^* aba$.

(k) True or false: $T \to^* XX$.

(l) True or false: $T \to^* XXX$.

(m) True or false: $S \to^* \epsilon$.

* 2. Figure 1 contains a grammar for the Brischeme language from this week's lab sheet.
Which of the following are valid Brischeme programs (i.e. strings in the language of that grammar)? You do not have to give the derivations (but you should work through them in your head).

(a) (define x 32) (+ 4 x)

The syntax of the *Brischeme* programming language (ignoring whitespace) is given by the CFG with:

- Terminals: $0, 1, \ldots, 9$, $a, b, \ldots, z, A, B, \ldots, Z$, _, !, ?, $<$, $=$, $+$, $-$, $*$, /, (,), define, lambda, not, or, and, #t, #f.
- Nonterminals: *Prog*, *Form*, *SExpr*, *Ident*, *Literal*, *Num*, *Bool*, *Digit*, *LChar*, *IdChar*, *Primop*.
- The production rules are:

$$
\begin{array}{rcl}
\mathit{Prog} & ::= & \mathit{Form}^{*} \\
\\
\mathit{Form} & ::= & \mathit{SExpr} \\
 & | & (\ \text{define}\ \mathit{Ident}\ \mathit{SExpr}\ ) \\
\\
\mathit{SExpr} & ::= & \mathit{Literal} \\
 & | & \mathit{Ident} \\
 & | & (\ \mathit{SExpr}\ \mathit{SExpr}^{*}\ ) \\
 & | & (\ \mathit{Primop}\ \mathit{SExpr}^{*}\ ) \\
 & | & (\ \text{lambda}\ (\ \mathit{Ident}^{*}\ )\ \mathit{SExpr}\ ) \\
\\
\mathit{Ident} & ::= & \mathit{LChar}\ \mathit{IdChar}^{*} \\
\mathit{LChar} & ::= & \text{a} \mid \text{b} \mid \cdots \mid \text{z} \\
\mathit{IdChar} & ::= & a \mid b \mid \cdots \mid z \mid A \mid B \mid \cdots \mid \text{Z} \mid \text{!} \mid \text{?} \mid \text{\_} \\
\\
\mathit{Literal} & ::= & \mathit{Bool} \mid \mathit{Num} \\
\mathit{Bool} & ::= & \text{\#t} \mid \text{\#f} \\
\mathit{Num} & ::= & \mathit{Digit}\ \mathit{Digit}^{*} \\
\mathit{Digit} & ::= & 0 \mid 1 \mid \cdots \mid 9 \\
\\
\mathit{Primop} & ::= & \text{+} \mid \text{-} \mid \text{*} \mid \text{/} \mid \text{and} \mid \text{or} \mid \text{not} \mid \text{<} \mid \text{=}
\end{array}
$$

Figure 1: Brischeme (ignoring whitespace).

(b) (define x 32) (+ 4 y)

(c) (define x 32) (+ 4 5x)

(d) (define aC3! 32) (+ 4 aC3!)

(e) (define AC3! 32) (+ 4 AC3!)

(f) (+ (define x 32) x 4)

(g) (lambda x (+ x x))

(h) (define f (lambda (xy) (+ x (* 2 y))))

(i) (lambda (x b) (+ x (not b)))

(j) (lambda (x b) (if x))

(k) (lambda (x b) ((if b + -) 2 3))

(l) ((lambda () 3))

** 3. Design CFGs for the following programming language lexemes over the ASCII alphabet. You will find it convenient to use abbreviations like $\cdots$ to help present the expressions compactly.

(a) A *C program identifier* is any string of length at least 1 containing only letters ('a'–'z', lower and uppercase), digits ('0'–'9') and the underscore, and which begins with a letter or the underscore.

(b) An *integer literal* is any string taking one of the following forms:

- a non-empty sequence of digits (decimal)
- a non-empty sequence of characters from '0'–'9','a'–'e' (upper or lowercase) that are preceded by "0x" (hexadecimal)
- a non-empty sequence of bits '0' and '1' that are preceded by "0b" (binary)

** 4. Give a CFG that describes the language of all subsequences of all permutations (i.e. no repetition) of the following keywords:

final static synchronized

** 5. Give CFGs for the following languages. The later parts are more difficult than 2-star.

(a) All odd length strings over $\{a, b\}$.

(b) All strings over $\{a, b\}$ that contain aab as a substring.

(c) The set of strings over $\{a, b\}$ with more a than b. Hint: every string w with at least as many

a as b (possibly the same number of a as b) can be characterised inductively as follows. Either:

- w is just a
- or, w is of shape avb with v containing at least as many a as b
- or, w is of shape bva with v containing at least as many a as b
- or, w is of shape v_1v_2 with v_1 and v_2 each separately containing at least as many a as b
- or, w is the empty string

(d) The complement of the language $\{a^nb^n \mid n \geq 0\}$ over $\{a,b\}$. Hint: express "not of shape a^nb^n for some n" into one or more positive (i.e. without using *not* or similar) conditions.

(e) $\{v\#w \mid v,w \in \{a,b\}^*$ and the reverse of v is a substring of $w\}$, over $\{a,b,\#\}$.

*** 6. An ϵ-production is a rule of shape $X \longrightarrow \epsilon$ (for some nonterminal X). A unit production is a rule of shape $X \longrightarrow Y$ (for some non-terminals X and Y). Consider the following grammar G with start symbol S:

$$\begin{array}{rcl} S & \longrightarrow & aSbb \mid T \\ T & \longrightarrow & bTaa \mid S \mid \epsilon \end{array}$$

This grammar has two unit productions $S \longrightarrow T$ and $T \longrightarrow S$, and it has an epsilon production $T \longrightarrow \epsilon$.

Give a grammar with *no* unit productions and *no* ϵ-productions for the language $L(G) \setminus \{\epsilon\}$.

*** 7. Give an English description of the language of the grammar in Q1.