Programming Languages and Computation

Week 2: Context Free Grammars

We start with some revision of sets. If you are unfamiliar with any of the notation, please ask one of the TAs.

* 1. Write $\mathbb{N}$ for the set of natural numbers $0, 1, 2 \ldots$, and write Σ for the alphabet $\{0, 1\}$. List the elements of the following sets in any order.

(a) $\{1, 2, 3\} \cup \{2, 4, 6\}$

(b) $\{1, 2, 3\} \cap \{2, 4, 6\}$

(c) $\{1, 2, 3\} \times \{2, 4, 6\}$

(d) $\{1, 2, 3\} \setminus \{2, 4, 6\}$

(e) $\{2m \mid m \in \mathbb{N},\ 0 \leq m \leq 5\}$

(f) $\{uu \mid u \in \Sigma^*, |u| = 2\}$

(g) $\{u0v \mid u \in \Sigma^*,\ v \in \Sigma^*,\ |uv| = 2\}$

(h) $\{uvw \mid u \in \Sigma,\ v \in \Sigma,\ w \in \Sigma,\ w \text{ is the xor of } u \text{ and } v\}$

Solution

(a) $1, 2, 3, 4, 6$

(b) 2

(c) $(1, 2)$, $(1, 4)$, $(1, 6)$, $(2, 2)$, $(2, 4)$, $(2, 6)$, $(3, 2)$, $(3, 4)$, $(3, 6)$

(d) $1, 3$

(e) $0, 2, 4, 6, 8, 10$

(f) 0000, 0101, 1010, 1111

(g) 001, 100, 101, 000, 010, 011, 110

(h) 000, 011, 101, 110

Recall the grammar from the notes for Boolean expressions.

* 2. Give derivations for the following strings which are in the language of Boolean expressions.

> (a) true && false
>
> (b) true || false && true
>
> (c) true && (false || true)

Solution

There are different possible derivations depending on which non-terminals are chosen to be replaced. I will generally replace leftmost non-terminals at each step.

(a)

$$\begin{aligned}
B &\to F \\
&\to L \mathbin{\&\&} F \\
&\to \text{true} \mathbin{\&\&} F \\
&\to \text{true} \mathbin{\&\&} L \\
&\to \text{true} \mathbin{\&\&} \text{false}
\end{aligned}$$

(b)

$$\begin{aligned}
B &\to F \mathbin{||} B \\
&\to L \mathbin{||} B \\
&\to \text{true} \mathbin{||} B \\
&\to \text{true} \mathbin{||} F \\
&\to \text{true} \mathbin{||} L \mathbin{\&\&} F \\
&\to \text{true} \mathbin{||} \text{false} \mathbin{\&\&} F \\
&\to \text{true} \mathbin{||} \text{false} \mathbin{\&\&} L \\
&\to \text{true} \mathbin{||} \text{false} \mathbin{\&\&} \text{true}
\end{aligned}$$

(c)

$$\begin{aligned}
B &\to F \\
&\to L \mathbin{\&\&} F \\
&\to \text{true} \mathbin{\&\&} F \\
&\to \text{true} \mathbin{\&\&} L \\
&\to \text{true} \mathbin{\&\&} (B) \\
&\to \text{true} \mathbin{\&\&} (F \mathbin{||} B) \\
&\to \text{true} \mathbin{\&\&} (L \mathbin{||} B) \\
&\to \text{true} \mathbin{\&\&} (\text{false} \mathbin{||} B) \\
&\to \text{true} \mathbin{\&\&} (\text{false} \mathbin{||} F) \\
&\to \text{true} \mathbin{\&\&} (\text{false} \mathbin{||} L) \\
&\to \text{true} \mathbin{\&\&} (\text{false} \mathbin{||} \text{true})
\end{aligned}$$

* 3. Give three distinct derivations for the string true || false.

Solution

(a)

$$\begin{aligned} B &\to F \mathbin{\|} B \\ &\to L \mathbin{\|} B \\ &\to \textsf{true} \mathbin{\|} B \\ &\to \textsf{true} \mathbin{\|} F \\ &\to \textsf{true} \mathbin{\|} L \\ &\to \textsf{true} \mathbin{\|} \textsf{false} \end{aligned}$$

(b)

$$\begin{aligned} B &\to F \mathbin{\|} B \\ &\to F \mathbin{\|} F \\ &\to L \mathbin{\|} F \\ &\to \textsf{true} \mathbin{\|} F \\ &\to \textsf{true} \mathbin{\|} L \\ &\to \textsf{true} \mathbin{\|} \textsf{false} \end{aligned}$$

(c)

$$\begin{aligned} B &\to F \mathbin{\|} B \\ &\to F \mathbin{\|} F \\ &\to F \mathbin{\|} L \\ &\to F \mathbin{\|} \textsf{false} \\ &\to L \mathbin{\|} \textsf{false} \\ &\to \textsf{true} \mathbin{\|} \textsf{false} \end{aligned}$$

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

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

(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: $S \to^* \epsilon$.

Solution

(a) R, S, T, X

(b) a, b

(c) ab, ba, aab

(d) aa, bb, ϵ

(e) false

(f) true

(g) false

(h) true

(i) true

(j) false

(k) true

(l) true

(m) false

* 5. Recall the syntax of the While programming language from the reference notes. Which of the following are valid While 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) $\mathit{foo} \leftarrow x + 2$

(b) $\mathit{foo} \leftarrow x + 2;\ \mathsf{skip}$

(c) $\mathsf{while}\ x\ \mathsf{do\ skip}$

(d) $\mathsf{if}\ x \leq 3 \parallel !\, x = y\ \mathsf{then}\ x \leftarrow x + y\ \mathsf{else\ skip}$

(e) $x \leftarrow 2;\ y \leftarrow 0;$

(f) 6

(g) $\{\mathit{bar} \leftarrow \mathit{foo}\};\ \mathit{foo} \leftarrow \mathit{bar}$

(h) $\mathsf{while\ true\ do}\ \{y \leftarrow y + 1;\ z \leftarrow z + 1\}; x \leftarrow 1$

Solution

(a) yes

(b) yes

(c) no

(d) yes

(e) no

(f) no

(g) yes

(h) yes

** 6. 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)

Solution

(a) This is one way to describe it, I will write terminal symbols in `terminal` type to distinguish them:

$$\begin{array}{rcl} S & \longrightarrow & LM \\ M & \longrightarrow & LM \mid DM \mid \epsilon \\ L & \longrightarrow & \mathtt{a} \mid \mathtt{b} \mid \cdots \mid \mathtt{z} \mid \mathtt{\_} \mid \mathtt{A} \mid \mathtt{B} \mid \cdots \mid \mathtt{Z} \\ D & \longrightarrow & \mathtt{0} \mid \mathtt{1} \mid \cdots \mid \mathtt{9} \end{array}$$

(b) One straightforward way is as follows (here I use `terminal` type for the terminal symbols to distinguish them from nonterminals):

$$\begin{array}{rcl} S & \longrightarrow & D \mid H \mid B \\ D & \longrightarrow & ND \mid N \\ N & \longrightarrow & \mathtt{0} \mid \mathtt{1} \mid \cdots \mid \mathtt{9} \\ H & \longrightarrow & \mathtt{0x}G \\ G & \longrightarrow & AG \mid A \\ A & \longrightarrow & N \mid \mathtt{a} \mid \mathtt{b} \mid \cdots \mid \mathtt{e} \mid \mathtt{A} \mid \mathtt{B} \mid \cdots \mid \mathtt{E} \\ B & \longrightarrow & \mathtt{0b}Z \\ Z & \longrightarrow & YZ \mid Y \\ Y & \longrightarrow & \mathtt{0} \mid \mathtt{1} \end{array}$$

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

> final static synchronized

Solution

$$\begin{array}{rcl} S & \longrightarrow & ABC \mid ACB \mid BAC \mid BCA \mid CAB \mid CBA \\ A & \longrightarrow & \textsf{final} \mid \epsilon \\ B & \longrightarrow & \textsf{static} \mid \epsilon \\ C & \longrightarrow & \textsf{synchronized} \mid \epsilon \end{array}$$

** 8. 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 $\{a^n b^n \mid n \geq 0\}$ over $\{a, b\}$. Hint: express "not of shape $a^n b^n$ 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 $\{a, b, \#\}$.

Solution

(a)

$$\begin{array}{rcl} S & \longrightarrow & XSX \mid X \\ X & \longrightarrow & a \mid b \end{array}$$

(b)

$$\begin{array}{rcl} S & \longrightarrow & XSX \mid aab \\ X & \longrightarrow & a \mid b \mid \epsilon \end{array}$$

(c) Here we dedicate a nonterminal T to describe the inductive characterisation given in the hint and then use S to force an extra a.

$$\begin{array}{rcl} S & \longrightarrow & TaT \\ T & \longrightarrow & a \mid aTb \mid bTa \mid TT \mid \epsilon \end{array}$$

(d) When you are asked to give the complement, or otherwise to describe something according to a negated condition, it is best to translate it immediately into one or more positive conditions because CFG have no direct way to express negation. In this case, not of shape $a^n b^n$ means either:

- The word has all the a before any b but there are strictly more a than b (nonterminal R),
- or, the word has all the a before any b, but there are more b than a (nonterminal T),
- or, the word has a b occurring before an a (non-terminal U).

$$\begin{array}{rcl} S & \longrightarrow & R \mid T \mid U \\ R & \longrightarrow & AaRb \mid a \\ A & \longrightarrow & aA \mid \epsilon \\ T & \longrightarrow & aTbB \mid b \\ B & \longrightarrow & bB \mid \epsilon \\ U & \longrightarrow & XbXaX \\ X & \longrightarrow & aX \mid bX \mid \epsilon \end{array}$$

(e) Strings in this language have the following shape: $v\#w_1v^Rw_2$ where w_1 and w_2 are completely arbitrary. We need to match v to v^R letter by letter, so this is best done by a single nonterminal (T). The "middle" of the string generated by this non-terminal (between the matching pieces of v and its reverse) will contain the # and, to the right of it, any arbitrary substring w_1. Finally, the whole string has an arbitrary suffix w_2. We can describe this as follows:

$$\begin{array}{rcl} S & \longrightarrow & TX \\ X & \longrightarrow & aX \mid bX \mid \epsilon \\ T & \longrightarrow & aTa \mid bTb \mid \#X \end{array}$$

*** 9. 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 $L(G) \setminus \{\epsilon\}$.

Solution

The following grammar describes $L(G) \setminus \{\epsilon\}$ without using unit or ϵ-productions:

$$S \longrightarrow aSbb \mid bSaa \mid abb \mid baa$$

We can obtain it in the following way (which more-or-less follows a general approach to eliminating mutually recursive definitions in programming). First, since $S \longrightarrow T$ and $T \longrightarrow S$, we can replace the production $T \longrightarrow bTaa$ by $T \longrightarrow bSaa$ without changing the language of the grammar. The new grammar, call it G', is:

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

The reasoning for why this does not change the language is as follows.

- The new grammar (with this production replaced) cannot derive any strings that were not already derivable in the old grammar, because we could already derive $T \to bTaa \to bSaa$ in the old grammar.
- The new grammar (with this production replaced) can derive all the string that were derivable in the old grammar because we can derive $T \to bSaa \to bTaa$ in the new grammar.

Therefore, the new grammar and the old grammar derive the same strings. Now, we have eliminated the directly recursive part of the T-rules, we can look to eliminate T altogether from G'. We can replace the production $S \longrightarrow T$ in G' by the three productions $S \longrightarrow bSaa \mid S \mid \epsilon$. This does not change the language of the grammar because if we use $S \longrightarrow T$, then we must then replace T by something, and these are the three possibilities (so we are just removing a step from the derivation). This gives equivalent grammar G'':

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

But note that $S \longrightarrow S$ is a useless production which contributes nothing to the language described, so we can remove it. Then note that it is impossible to introduce a T nonterminal when starting from S, so all the T rules are useless in G''. Hence, we obtain the equivalent grammar G''':

$$S \longrightarrow aSbb \mid bSaa \mid \epsilon$$

Now we still want to remove the ϵ-production. In general this will change the language, because $S \to \epsilon$ is a valid derivation. However, we want to describe the language $L(G) \setminus \{\epsilon\}$ anyway, so this derivation should be impossible. We just observe that, in a derivation of a string in $L(G''') \setminus \{\epsilon\}$, the production $S \longrightarrow \epsilon$ will only be used after one of the other two productions. Thus we can combine $S \to aSbb \to abb$ and $S \to bSaa \to baa$ into two single rules and get rid of the ϵ-production:

$$S \longrightarrow aSbb \mid bSaa \mid abb \mid baa$$

*** 10. Give an English description of the language of the grammar in Q4.

Solution

The language of strings over $\{a, b\}$ that are not palindromes.