

Worth: 10%

Due: By 1pm on Wednesday 28 January

- [30] 1. For each language below, either give a context-free grammar (CFG) for the language (and justify briefly that your CFG correctly generates all strings in the language, and **only** strings in the language), or state that there is *no* CFG for the language (no justification required). Write all of your answers in “basic” BNF, *i.e.*, *without* using any of the extensions shown in class.
- All strings over $\{0, 1\}$ whose *length* is a multiple of 5.
 - All strings over $\{0, 1\}$ in which all 0’s occur in pairs and are eventually followed by a 1.
 - All strings over $\{a, b, c\}$ of the form $a^\ell b^m c^n$, where $\ell = m + n$.
 - All strings over $\{a, b, c\}$ of the form $a^\ell b^m c^n$, where $m = \ell + n$.
 - All strings over $\{a, b, c\}$ that contain the same number of *a*’s and *c*’s—in any order, and independently of the number of *b*’s. (HINT: Suppose you count +1 for each ‘*a*’ and –1 for each ‘*c*’, moving left-to-right one character at a time. What can you say about the value of the count, and how does this help you define a pattern that the string must follow?)
 - All strings over $\{0, 1\}$ that do **not** contain 1100 as a substring. (**Tricky!**)
- [60] 2. Consider the following CFGs.

 G_1 :
$$\begin{aligned} \langle \text{expr} \rangle &::= \langle \text{expr} \rangle \text{ or } \langle \text{expr} \rangle \mid \\ &\quad \langle \text{expr} \rangle \text{ and } \langle \text{expr} \rangle \mid \\ &\quad \text{not } \langle \text{expr} \rangle \mid (\langle \text{expr} \rangle) \mid \\ &\quad \langle \text{value} \rangle == \langle \text{value} \rangle \mid \\ &\quad \langle \text{value} \rangle <= \langle \text{value} \rangle \mid \\ &\quad \langle \text{value} \rangle >= \langle \text{value} \rangle \\ \langle \text{value} \rangle &::= \langle \text{var} \rangle \mid \langle \text{num} \rangle \\ \langle \text{var} \rangle &::= \langle \text{alpha} \rangle \mid \langle \text{alpha} \rangle \langle \text{var} \rangle \\ \langle \text{alpha} \rangle &::= a \mid b \mid c \mid \dots \mid x \mid y \mid z \\ \langle \text{num} \rangle &::= \langle \text{digit} \rangle \mid \langle \text{digit} \rangle \langle \text{num} \rangle \\ \langle \text{digit} \rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{aligned}$$
 G_2 :
$$\begin{aligned} \langle \text{expr} \rangle &::= \langle \text{expr} \rangle \text{ or } \langle \text{expr} \rangle \mid \langle \text{conj} \rangle \\ \langle \text{conj} \rangle &::= \langle \text{conj} \rangle \text{ and } \langle \text{conj} \rangle \mid \langle \text{pred} \rangle \\ \langle \text{pred} \rangle &::= \text{not } \langle \text{pred} \rangle \mid (\langle \text{expr} \rangle) \mid \langle \text{comp} \rangle \\ \langle \text{comp} \rangle &::= \langle \text{value} \rangle == \langle \text{value} \rangle \mid \\ &\quad \langle \text{value} \rangle <= \langle \text{value} \rangle \mid \\ &\quad \langle \text{value} \rangle >= \langle \text{value} \rangle \\ \langle \text{value} \rangle &::= \langle \text{var} \rangle \mid \langle \text{num} \rangle \\ \langle \text{var} \rangle &::= \langle \text{alpha} \rangle \mid \langle \text{alpha} \rangle \langle \text{var} \rangle \\ \langle \text{alpha} \rangle &::= a \mid b \mid c \mid \dots \mid x \mid y \mid z \\ \langle \text{num} \rangle &::= \langle \text{digit} \rangle \mid \langle \text{digit} \rangle \langle \text{num} \rangle \\ \langle \text{digit} \rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{aligned}$$

- Give a parse tree in G_1 for the string “not $x == 4$ or $x <= 2$ ”. Does this string have more than one parse tree in G_1 ? Justify.
- Give a parse tree in G_2 for the string “not $x == 4$ or $x <= 2$ ”. Does this string have more than one parse tree in G_2 ? Justify.
- Give a specific example of a string with *more than one* parse tree in G_2 —also give at least two distinct parse trees for your string.
- Give a specific example of a string *not* produced by G_1 , and explain why this is the case.
- Give a specific example of a string *not* produced by G_2 , and explain why this is the case.
- Prove or disprove: $L(G_2) \subseteq L(G_1)$ (*i.e.*, each string produced by G_2 can be produced by G_1).
- Prove or disprove: $L(G_1) \subseteq L(G_2)$.

- (h) Give an *unambiguous* grammar G_3 for the language $L(G_2)$. State explicitly the decisions you made concerning associativity in order to disambiguate the language. Explain why your string from question 2c has a unique parse tree in G_3 .

[10] 3. ML warm-up question!

The functions are simple enough that you could easily look up the answer, but you are encouraged to develop your own code, and learn how to write and run it on CDF (or whatever platform you would like to use—but remember that for Assignment 2, your code must run on CDF).

We will mark your code by hand, but you are encouraged to try and run it first to make sure it is correct! However, simply submit your written solution (no need for your source code). Part of your mark will be based on using ML conventions properly—in other words, you will be penalized if you misuse ML features to try to write your solution in what is essentially Python or Java or C or any other imperative language.

- (a) Write a function `nozero` that takes a list of integers and that returns the list with every 0 removed.
- (b) Write a function `nox` that takes an integer x and a list of integers and that returns the list with every instance of x removed.
- (c) **Bonus!** Write a function `rem` that takes an integer x and that returns another function, that removes every instance of x from a list of integers.