

CS 81, Logic and Computability
Problem Set 8: Context-Free Grammars

This problem set uses the Stanford CS 103 CFG developer:
<https://web.stanford.edu/class/archive/cs/cs103/cs103.1156/tools/cfg/>.

Challenge 1: Grammar Youngling [60 Points]

Using the online CFG tool, construct an **unambiguous** CFG that produces all the strings of **just zeroes** that have an even number of zeroes, and produces no other strings.

Then, test your CFG on the following strings:

```
1
00

0000
000000
00000000
000
0
00000
1
00100
```

Notice, the second item we test for is epsilon, the empty string. We test for that by including a blank line in our test box.

Include your grammar and a screenshot of the test results. The screenshot should look like Figure 3. **For full points** your grammar should only have unambiguous derivations for the strings in that language. Our sample solution uses two rules.

Challenge 2: Grammar Padawan [35 Points]

Let $L_1 = \{w\#x \mid w, x \in \{0, 1\}^* \wedge (|w| = |x|) \wedge (w \neq x^R)\}$. Note that the input alphabet is $\Sigma = \{0, 1, \#\}$ and recall that x^R denotes the string x in reverse order.

For example, the following strings are in L_1 : 000#001, 1100#0001, 00#01. However, the following strings are not in L_1 : 00#000, 001#100.

Construct a context-free grammar for L_1 using the online tool and **include a screenshot of the results for tests** of the following strings:

```
00#00
00#01
00#000
1100#001
001#100
0
000#001
1#0
0010#0
001##101
```

Please include your grammar and your screenshot of the test results. Ambiguous grammars are fine for this problem.

Challenge 3: Grammar Jedi [5 Points]

This is a practice challenge, in case you want a more challenging problem to work on. It is only worth a small number of points, so don't sweat it if you can't figure it out.

Let $P(w)$ be a predicate on the domain of strings in $\{0,1\}^*$ where the interpretation of $P(w)$ is that the number of 0's in w is different from the number of 1's in w . Consider the language $L_2 = \{w \mid w \in \{0,1\}^* \wedge P(w)\}$. For example, the following strings are in L_2 : 000, 001, 100, 10110.

Construct a context-free grammar for L_2 using the online tool, and **include a screenshot of the results for tests** of the following strings:

```
000
001
100
10110
001100
01
000001
10
11001001
1010101010
```

Please include your grammar and your screenshot of the test results. Ambiguous grammars are fine for this problem.

Answers

Answer 1:

Verify

This is the CFG you have input above:

Start symbol: **S**

$S \rightarrow \epsilon \mid 0A$

$A \rightarrow 0S$

Some strings from the language of this grammar:

```
000000000000
00000000000000
00
0000
0000000000000000
000000000
000000
00000000000
```

Test

To test the CFG above, input test strings here, one per line. An empty line corresponds to the empty string. Results will be shown automatically.

```
0000
000000
00000000
000
0
00000
1
00100
```

Test Results for CFG

#	String	Matches	
1	"00"	Yes	See Derivation
2	" "	Yes	See Derivation
3	"0000"	Yes	See Derivation
4	"000000"	Yes	See Derivation
5	"00000000"	Yes	See Derivation
6	"000"	No	
7	"0"	No	
8	"00000"	No	
9	"1"	No	
10	"00100"	No	

Figure 1: Screenshot of online tool tests.

Answer 2:

Answer 3:

Verify

This is the CFG you have input above:

Start symbol: **S**
S $\rightarrow \epsilon \mid A \mid B$
A $\rightarrow 0C1 \mid 1C0$
B $\rightarrow 0A0 \mid 1A1 \mid 0B0 \mid 1B1$
C $\rightarrow \# \mid 0C0 \mid 1C1 \mid 1C0 \mid 0C1$

Some strings from the language of this grammar:

```
0011#0110
1100#1100
01#00
0001000#0010010
100111#1110100
ε
1111100#1100101
011111#110010
00#01
1110#0010
```

Test

To test the CFG above, input test strings here, one per line. An empty line corresponds to the empty string. Results will be shown automatically.

```
1100#001
001#100
0
000#001
1#0
0010#0
001##101
```

Test Results for CFG

#	String	Matches
1	"00#00"	No
2	"00#01"	Yes See Derivation
3	"00#000"	No
4	"1100#001"	No
5	"001#100"	No
6	"0"	No
7	"000#001"	Yes See Derivation
8	"1#0"	Yes See Derivation
9	"0010#0"	No
10	"001##101"	No

Figure 2: Screenshot of online tool tests.

Verify

This is the CFG you have input above:

Start symbol: **S**
S → **A0A** | **B1B**
A → ϵ | **0A1A** | **1A0A** | **0A**
B → ϵ | **0B1B** | **1B0B** | **1B**

Some strings from the language of this grammar:

```
1110
1110111110101
001100101000
000110001
11011101
010001000110000
0000101
10001010000010
001
010101110000010
```

Test

To test the CFG above, input test strings here, one per line. An empty line corresponds to the empty string. Results will be shown automatically.

```
100
10110
001100
01
000001
10
11001001
1010101010
```

Test Results for CFG

#	String	Matches	
1	"000"	Yes (ambiguously)	Derivation One Derivation Two
2	"001"	Yes	See Derivation
3	"100"	Yes	See Derivation
4	"10110"	Yes (ambiguously)	Derivation One Derivation Two
5	"001100"	Yes (ambiguously)	Derivation One Derivation Two
6	"01"	No	
7	"000001"	Yes (ambiguously)	Derivation One Derivation Two
8	"10"	No	
9	"11001001"	No	
10	"1010101010"	No	

Figure 3: Screenshot of online tool tests.