1 Showing a language is not regular

After this tutorial you should be able to:

1. Show a language is not regular.

You may find the following terminology helpful. For a language L, we say that two strings x and y are *distinguished by* L if there is a string z such that only one of xz and yz are in L (and in this case we say that z *distinguishes* x and y). The method we saw in lectures to prove that a language is not regular, is to find (define) infinitely many strings $x_1, x_2, \cdots$ such that every pair of them is distinguished by L.

Problem 1. Fix $\Sigma = \{a, b\}$. Using the technique from lectures, show the following languages are not regular (for the tutorial pick one).

1. $\{a^i b^j : i > j\}$.
2. $\{a^n b^m : n$ divides m, or m divides $n\}$.
3. $\{a^{n^2} : n \geq 0\}$.
4. All strings $a^i b^j$ such that (a) i is even, or (b) $j < i$ and j is even. (hard)

2 Context free grammars

In the lecture we saw a new model of computation called 'context-free grammar' (CFG). A CFG generates strings by rewriting variables (starting with the 'start' variable) until there are none left. The language of a CFG is called a 'context-free language'. These include all the regular languages, and more. CFGs are central to describing the syntax of many programming languages. We ended the lecture with the fact that some CFGs can generate a string with more than one rightmost derivation (= more than one leftmost derivation, = more than one parse tree). Such strings are called ambiguous, and correspond, intuitively, to different "meanings" of the string.

After this tutorial you should be able to:

1. Convert an English or mathematical description of a context-free language into a CFG, and argue why your grammar is correct.
2. Describe in English or mathematics the language of a CFG, and argue why your description is correct.
3. Argue if a CFG is ambiguous or not.

Problem 2. For each of the following grammars:

Indicate the set of variables, terminals, production rules, and the start variable.

Give two strings derivable by this grammar, and two strings that are not.

Describe in one sentence the language generated by the grammar.

Is the language regular?

1.

$$S \to 0S0 \mid 1$$

2.

$$S \to X1X$$
$$X \to 0X \mid \epsilon$$

Problem 3. Consider the following grammar:

$$E \to E + T \mid E - T \mid T$$
$$T \to F \times T \mid T/T \mid F$$
$$F \to (E) \mid V \mid C$$
$$V \to a \mid b \mid c$$
$$C \to 1 \mid 2 \mid 3$$

1. Indicate the set of variables, terminals, production rules, and the start variable.
2. Give a left-most derivation of the string $a + b \times c$
3. Give a right-most derivation of the string $a + b \times c$
4. Give a parse tree for $a \times b - 2 \times c$
5. Give a parse tree for $a \times (b - 2 \times c)$

Problem 4. For each of the following languages, find a CFG that generates it (for the tutorial, pick two of these):

1. $\{bb, bbbb, bbbbbb, ...\} = \{(bb)^{n+1} \mid n \in \mathbb{Z}_0^+\}$
2. $\{a, ba, bba, bbba, ...\} = \{b^n a \mid n \in \mathbb{Z}_0^+\}$
3. $\{\varepsilon, ab, abab, ...\} = \{(ab)^n \mid n \in \mathbb{Z}_0^+\}$
4. $\{ac, abc, abbbc, ...\} = \{ab^n c \mid n \in \mathbb{Z}_0^+\}$
5. $\{a^m b^n \mid n, m \in \mathbb{Z}_0^+\}$
6. $\{a^m b^n \mid n, m \in \mathbb{Z}_0^+, m > 0\}$
7. $\{a^m b^n \mid n, m \in \mathbb{Z}_0^+, m > 0, n > 0\}$
8. $\{a^n b^n : n > 0\}$

Problem 5.

1. Write a CFG for the language of balanced parentheses. So, e.g., the following strings are in the language: $()(())$ and $((()))$ and ϵ, while the following are not: $)()$ and $((()))$. Explain each rule in at most two sentences.
2. Show this language is not regular.

Practice problems

Problem 6. (Exam 2022) Fix $\Sigma = \{0, 1\}$. Provide a context-free grammar for the language

$$\{u0v1w : \text{len}(v) = \text{len}(u) + \text{len}(w)\}$$

For instance, taking $u = 01, w = 11, v = 1100$ we get that $u0v1w = 0101100111$ is in the language.

Give a short explanation/justification of your answer.

Problem 7. Some programming languages define the `if` statement in similar ways to the following grammar:

$$\begin{aligned} \textit{Conditional} &\to \text{ if } \textit{Condition} \text{ then } \textit{Statement} \\ \textit{Conditional} &\to \text{ if } \textit{Condition} \text{ then } \textit{Statement} \text{ else } \textit{Statement} \\ \textit{Statement} &\to \textit{Conditional} \mid S_1 \mid S_2 \\ \textit{Condition} &\to C_1 \mid C_2 \end{aligned}$$

1. Show that this grammar is ambiguous.
2. Show that the string you provided can be interpreted in two different ways, i.e., resulting in programs with different behaviours. You may want to write the programs corresponding to each derivation (hint: you can read these off from the parse trees).
3. Write a CFG that captures `if` statements but is not ambiguous.

Problem 8. Consider the CFG G':

$$S \to Sab \mid aSb \mid abS \mid Sba \mid bSa \mid baS \mid \epsilon$$

1. Show that if $x \in L(G')$ then x has the same number of as as bs.
2. Show that there is a string with the same number of as as bs that is not in $L(G')$.

The moral of this problem is that it is easy to design a CFG that misses some intended strings (in this case, the intended language is "same number of as as bs").

Problem 9. I asked GPT-4 to write recursive code for the language of well-formed parentheses.

It gave me this:

```python
def is_well_formed(s):
    # Base case: An empty string is well-formed
    if not s:
        return True
    # If string starts with '(' and ends with ')'
    if s[0] == '(' and s[-1] == ')':
        return is_well_formed(s[1:-1])
    return False
```

Is this correct?

Problem 10. Consider the following context-free grammar G:

$$S \to ST$$
$$S \to a$$
$$T \to BS$$
$$B \to +$$

For each of the following strings, say whether it is generated by the grammar, and if so, give a derivation, and if not give a reason:

1. $+a+a$
2. $a+a$
3. $a+a+a$
4. $a+a++$

Problem 11. Describe the language generated by the following context-free grammar:

$$S \to X1Y$$
$$X \to \epsilon \mid X0$$
$$Y \to \epsilon \mid 1Y \mid Y0$$

Briefly explain why your answer is correct.

Is the grammar ambiguous?

Problem 12. Consider the following context-free grammar:

$$S \to 0S \mid 0S1S \mid \epsilon$$

For each of the following words, say whether or not it is generated by the grammar: 001, 0101, 0110. The only justification that is needed is a derivation for those words that are generated by the grammar.

Show that the grammar is ambiguous.

Problem 13. (Practice Exam) Fix $\Sigma = \{0, 1, \#\}$. Provide a context-free grammar for the language of strings of the form $u\#v$ where $u, v \in \{0,1\}^*$ and $|u| = |v|$ and the reverse of u is not equal to v.

For instance, $01\#11, 011\#011$ is in the language, while $\#, 0\#0, 01\#10$ are not.

Give a short explanation/justification of your answer.

Problem 14. A CFG is called *right-regular* if every rule is of the form $A \to a$ or $A \to aB$ where A, B are arbitrary nonterminals and a is an arbitrary terminal or ϵ. A CFG is called *left-regular* if every rule is of the form $A \to a$ or $A \to Ba$ where A, B are arbitrary nonterminals and a is an arbitrary terminal or ϵ. A CFG is *regular* if it is left-regular or right-regular.

1. Write a right-regular grammar generating the language $a \mid b^*c$.
2. Write a left-regular grammar generating the language $a \mid b^*c$.
3. Find a grammar that only has rules of the form $A \to a$ or $A \to aB$ or $A \to Ba$, and that generates a language that is not the language of any RE (very hard, but we will see how to do this later).
4. Show that the regular grammars generated exactly the languages of regular expressions (very hard, but we will see how to do this later).

Problem 15. Give a context-free grammar which generates just the valid identifiers. A valid identifier begins with a letter or underscore, contains only letters, underscores, or digits, and is ended by a blank.

Problem 16. Give a CFG that generates the language of propositional formulas over the atoms p, q. Make sure to state what the variables and terminals are.

Problem 17. Provide a context-free grammar for the following language L: the set of well-bracketed strings that use two different types of brackets, i.e., $()$ and $\{\}$. For instance, $(\{\})()$ is in the language, as is $(())$, as is $()\{\}$, but $(\}$ is not. To justify your answer you should briefly explain a) why your grammar only generates strings in L, and b) why your grammar generates all strings in L.

Problem 18. Consider the following grammars.

1. $S \to a \mid SbS$
2. $S \to aS \mid Sa \mid a$
3. $S \to S[S]S \mid \varepsilon$
4. $S \to 0S1 \mid 01$

5. $S \to +SS \mid -SS \mid a$
6. $S \to SS+ \mid SS* \mid a$
7. $S \to S(S)S \mid \epsilon$
8. $S \to aSbS \mid bSaS \mid \epsilon$
9. $S \to a \mid S+S \mid SS \mid S* \mid (S)$

For each:

- Describe in English the language generated, and justify your answer.
- Say if the language is ambiguous, and justify your answer.
- If ambiguous, try find an equivalent unambiguous grammar.

Problem 19. Construct unambiguous CFGs for each of the following languages. In each case show that your grammar is correct.

1. Arithmetic expressions in postfix notation.
2. Left-associative lists of identifiers separated by commas.
3. Right-associative lists of identifies separated by commas.
4. Arithmetic expressions of integers and identifiers with the four binary operations $+, -, *, /$.
5. Add unary plus and minus to the arithmetic operations of the previous language.

Problem 20. Find a syntactic condition on the structure of a grammar that guarantees that the language generated by the grammar is infinite (i.e., contains infinitely many strings).

Problem 21. Here is an application of grammars to Compilers.

Syntax-directed translation is done by attaching rules or programs fragments to productions in a grammar.

Consider the CFG generating the set of binary strings:

$$\begin{aligned} S &\to 0 \\ S &\to 1 \\ S &\to S0 \\ S &\to S1 \end{aligned}$$

Attach to each rule the following pseudocode:

$$\begin{aligned} S \to\ & 0 && S.val = 0 \\ S \to\ & 1 && S.val = 1 \\ S \to\ & S0 && S.val = S.val \times 2 \\ S \to\ & S1 && S.val = S.val \times 2 + 1 \end{aligned}$$

Write an algorithm that will take a parse-tree for a string as input, do a bottom-up traversal of the tree (i.e., visit a node only after visiting all its children), and when visiting an internal node of the tree, executes the corresponding command. Once the traversal is complete, what is the content of $S.val$?

Problem 22. Construct syntax-guided translations between notations of arithmetic expressions:

1. From infix notation to prefix notation.
2. From infix notation to prefix notation.
3. From postfix notation to infix notation.

Problem 23. In lectures we said that the following CFG G generates the set L of strings with the same number of as as bs.

$$S \to \epsilon \mid aSbS \mid bSaS$$

Why is this true?

Problem 24. Show that the set of strings over the alphabet $\{(,)\}$ accepted by the function *check* is the set of strings generated by the grammar $S \to SS \mid (S) \mid \varepsilon$.

Figure 1: Pseudocode

```
def check(myStr):
    counter = 0
    for i in myStr:
        if i == "(":
            counter += 1
        elif i == ")":
            if counter > 0:
                counter -= 1
            else:
                return "Reject"
    if counter == 0:
        return "Accept"
    else:
        return "Reject"
```