Monash University
Faculty of Information Technology
2^{nd} Semester 2023

FIT2014 Exercises 6 Context-Free Grammars and the Pumping Lemmas

Attempt all questions in the main section before your class.

ASSESSED PREPARATION: Question 3.
You must provide a serious attempt at this entire question at the start of your class.

1.
(a) Find a Context-Free Grammar for the language

$$\{\mathbf{a}^n\mathbf{b}^i\mathbf{c}^{2n} : i, n \in \mathbb{N}\}.$$

(b) Prove, by induction on the string length, that every string of the form $\mathbf{a}^n\mathbf{b}^i\mathbf{c}^{2n}$ can be generated by your grammar.
(c) Find a Pushdown Automaton that recognises this language.

2. Let LOL be the language generated by the following Context-Free Grammar.

$$\begin{aligned} S &\rightarrow P && (1)\\ P &\rightarrow PP && (2)\\ P &\rightarrow \mathtt{ha}Q && (3)\\ Q &\rightarrow Q\mathtt{a} && (4)\\ Q &\rightarrow \varepsilon && (5) \end{aligned}$$

(a) **Warm-up:**

(i). What is the shortest string in LOL? Give a derivation for it.

(ii). Is the above grammar regular?

(iii). Is LOL a regular language? If so, give a regular expression for it. If not, prove it.

(iv). Is the above grammar in Chomsky Normal Form? If so, why; if not, state why not.

(b) Here is an attempted proof that LOL is infinite. Comment on what is correct and incorrect in this proof.

1. Let x be some string in LOL.
2. The string x must have a derivation, using the above grammar.
3. The only rule in the grammar that does not have a nonterminal symbol on its right-hand side is the last one, (5).
4. So any derivation of x must finish with an application of (5).
5. Therefore the string just before x, in the derivation of x, must have a Q.
6. Therefore the derivation of x has the form

   $$S \Longrightarrow \cdots \Longrightarrow x_{\text{left}} Q x_{\text{right}} \overset{(5)}{\Longrightarrow} x_{\text{left}} x_{\text{right}} \quad = \; x,$$

   where x_{left} denotes the portion of x before Q, and x_{right} denotes the portion of x after Q.
7. So, instead of applying rule (5) to this last occurrence of Q, we could apply rule (4) followed by rule (5), giving

   $$S \Longrightarrow \cdots \Longrightarrow x_{\text{left}} Q x_{\text{right}} \overset{(4)}{\Longrightarrow} x_{\text{left}} Q \mathtt{a} x_{\text{right}} \overset{(5)}{\Longrightarrow} x_{\text{left}} \mathtt{a} x_{\text{right}}.$$

   This new string is also in LOL and is one letter longer than x. Call it y.
8. We can apply this argument again to y, to get a string in LOL that is one letter longer than y, and then we can get strings that are even longer than that, and so on, indefinitely.
9. So we can generate infinitely many strings. Therefore LOL is infinite.

(c) Prove by induction that, for all $n \geq 2$, the language LOL contains a string of length at least n.

(d) Here is an attempted *proof by contradiction* that LOL is infinite. It uses, in the middle, the proof given in (c). Comment on what is good and bad about this proof.

1. Assume, by way of contradiction, that LOL is finite.
2. $\langle$ *Insert here a correct proof, based on (c), that LOL has arbitrarily long strings.* $\rangle$
3. This contradicts our assumption that LOL is finite.
4. Therefore LOL is infinite.

(e) Construct a correct proof by contradiction that LOL is infinite, using the following approach:

1. Start by assuming LOL is finite, as in (c) step 1.
2. Show that LOL contains a nonempty string. (No need to use the assumed finiteness of LOL just yet.)
3. Among all the nonempty strings in LOL, choose x carefully . . . How should you choose it?
4. Show how to construct a string in LOL that is longer than x.
5. . . . . . .
6. Therefore LOL is infinite.

3.

Let `i` be a terminal symbol which we think of as representing an integer.

Let PLUS-MINUS be the language over the five-symbol alphabet {`i`,`+`,`-`,`(`,`)`} defined by the following rules:

- `i` belongs to PLUS-MINUS.
- If x belongs to PLUS-MINUS then so does `(`x`)`.
- If x and y belong to PLUS-MINUS, then so do x`+`y and x`-`y.

(a) Write a Context-Free Grammar for PLUS-MINUS.

(b) Using your grammar, give a derivation for the string `(i+(i-i))-i`.

Now let `d` be a terminal symbol that can stand for any date.

You can think of `i` and `d` as two tokens used by a hypothetical lexical analyser, where the lexemes for the token `i` would be actual integers and the lexemes for the token `d` would be actual dates (such as 24/9/1966). But you don't need to do anything with lexical analysers in this question.

A **date expression** is an expression using symbols from the six-symbol alphabet {`d`,`i`,`+`,`-`,`(`,`)`}, formed according to the following rules:

- `d` is a date expression
- An integer expression may be added or subtracted after any date expression, giving another date expression (but the date expression must come *before* the integer expression). (This represents adding/subtracting some number of days to/from a date to get another date.)
- One date expression may be subtracted from another date expression, giving an integer expression. (This represents finding the number of days between two dates.)
- Parentheses may be put around any date expression, giving another date expression (representing the same date).
- Integer expressions are constructed using the same rules as for PLUS-MINUS along with the extra rule that the difference between two date expressions is an integer expression.

Examples of valid date expressions: `d`, `d+i`, `d-(i+i)`, `d+d-d`, `((d+i)-(d-d))`, `d-((d-i)-(d+i))`.
Some <u>invalid</u> date expressions: ε, `i`, `i+d`, `i-d`, `d+d`, `d-d`, `((d+i)-(d-i))`.

(c) Write a Context-Free Grammar for the language of date expressions.

(d) Is the language of date expressions regular? Justify your answer.

π**.**

(a)

Suppose a PDA has the property that every transition *either* pushes *or* pops (not both).[1] Suppose that, if the PDA is in state q with an empty stack, then reads string w, and is then at state r

[1]This is a property of the modified PDA we construct when constructing a CFG from a PDA. See Lecture 14, slides 12–15.

with an empty stack. Let P be any directed path through the PDA from q to r that w can take for this to happen.

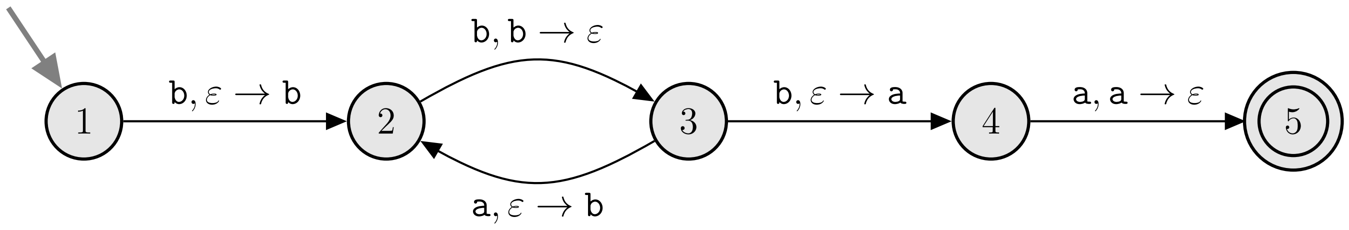
What can you say about the length of P?

(b)

How can we use this observation in deciding which symbols A_{qr} are needed for the process of converting the PDA to a CFG for the same language?

(c)

Consider the following PDA.



Using your observations in (a) and (b), which symbols A_{qr} do you actually need when applying the PDA-to-CFG algorithm to this PDA?

(d)

Convert this PDA to a CFG for the same language.

4. Recall the **Pumping Games** of Exercise Sheet 5. An expansion pack has now been released, called **Double Pumping Games**.

You can play a Double Pumping Game for *any* language. The players are Con and Noni. First, the players are given a language L (which may or may not be context-free). Con moves first, then Noni moves, then Con moves, then Noni moves. The rules for their moves are as follows.

- Con first chooses a number $k \in \mathbb{N}$.
- Then Noni chooses a string $w \in L$ with $|w| > 2^{k-1}$.
- Then Con divides w up into substrings u, v, x, y, z such that $vy \neq \varepsilon$ and $|vxy| \leq 2^k$.
- Then Noni chooses a non-negative integer i.

The result of the game is that:

- if $uv^ixy^iz \in L$, then Con wins;
- if $uv^ixy^iz \notin L$, then Noni wins.

(a) Play the game for (i) PALINDROMES, and (ii) $\{\mathtt{a}^n\mathtt{b}^n\mathtt{a}^n : n \geq 0\}$.

(b) Using quantifiers, write down the assertion that Con has a winning strategy.

(c) Using quantifiers, write down the assertion that Noni has a winning strategy.

(d) What does the Pumping Lemma for CFLs tell us about circumstances in which winning strategies exist?

5. Let G be the following CFG.

$$\begin{aligned} S &\rightarrow AB & (6)\\ A &\rightarrow AC & (7)\\ B &\rightarrow CD & (8)\\ A &\rightarrow \mathtt{a} & (9)\\ C &\rightarrow \mathtt{b} & (10)\\ D &\rightarrow \mathtt{a} & (11) \end{aligned}$$

Use the Cocke-Younger-Kasami (CYK) algorithm to determine whether or not the string `abbba` is generated by G.

6.
(a) Convert the CFG in Q2 into a Chomsky Normal Form grammar for LOL.
(b) Using this Chomsky Normal Form grammar, apply the CYK algorithm to the string `hahaa`.

7. In the game of Cricket, a *bowler* "bowls" a cricket ball at a *batsman.* Each time this is done, the batsman tries to hit the ball, and might score some number of *runs*, or the bowler might *take the wicket* of the batsman (in which case the batsman has to leave, and is replaced by another batsman). A bowler bowls the ball six times, making up an *over*, and then makes way for another bowler to have their turn.

In cricket statistics, a bowler's activity is summarised by four integers: O, M, R, W, where

- O is the total number of **overs** bowled by that bowler,
- M is the number of **maiden overs** by that bowler: these are those overs in which no run was scored by batsmen,
- R is the number of **runs** scored from the bowling of that bowler,
- W is the number of **wickets** taken by that bowler.

We assume:[2]

- A batsman can score up to six runs from each ball bowled, making up to 36 runs from an over.
- At most one wicket can be taken with each ball, making up to 6 wickets per over.

We also use the fact that the number of maiden overs is clearly at most the total number of overs.

Let CricketBowlingFigures be the language of quadruples (O, M, R, W) of nonnegative integers, each in unary[3], where $O \geq M$, $O - M \leq R \leq 36\,(O - M)$, and $W \leq 6\,O$. We assume that any of these numbers can be arbitrarily large, subject to these constraints. (The statistics can be compiled over entire lifetimes, not just single matches, and some bowlers may be immortal.) For example,

- $(\overbrace{\mathtt{111\cdots11}}^{27}, \mathtt{1111}, \overbrace{\mathtt{111\cdots11}}^{72}, \mathtt{11}) \in$ CricketBowlingFigures. It represents the bowling figures (27,4,72,2). [4]
- $(\mathtt{11}, \mathtt{1}, \overbrace{\mathtt{111\cdots11}}^{36}, \mathtt{111111111111}) \in$ CricketBowlingFigures. It represents the bowling figures (2,1,36,12).

[2]Cricket experts might notice some simplifying assumptions here . . .

[3]In *unary* representation, a number n is represented as a string of n 1s. For example, in unary, the number 5 is represented by 11111, and 0 is represented by the empty string.

[4]Question for cricket tragics: what is the significance of these bowling figures, in the history of Test Cricket?

- $(,,,) \in$ CricketBowlingFigures. It represents the bowling figures (0,0,0,0) of a bowler who does not get a turn at bowling.
- $(\texttt{1}, \texttt{11}, \texttt{111}, \texttt{1111}) \notin$ CricketBowlingFigures. The bowling figures (1,2,3,4) are impossible, since $O < M$.
- $(\texttt{11}, \texttt{1}, \overbrace{\texttt{111}\cdots\texttt{11}}^{36}, \texttt{1111111111111}) \notin$ CricketBowlingFigures. The bowling figures (2,1,36,13) are impossible, since $W > 6\,O$.

(a) Is the language CricketBowlingFigures regular? Prove or disprove.

(b) Is the language CricketBowlingFigures context-free? Prove or disprove.

Supplementary exercises

8. Find Regular Grammars for the following Regular Expressions:

i) $(a \cup b)^*(aa \cup bb)$

ii) $((a \cup b)(a \cup b))^*$

iii) $(aa \cup bb)^*$

iv) $(ab)^*aa(ba)^*$

v) $(ab \cup ba)(aa \cup bb)^*(ab \cup ba)$

9. Describe PDAs for each of the Regular Grammars you found in the previous question.

10. You have seen how to construct a grammar for any regular language, using an NFA for the language. But suppose we wanted to construct a grammar for a regular language *directly from a regular expression*, and *without constructing an NFA first*. How would you do it? (The grammar need not be regular.)

11.

Let k be a fixed positive integer. A *k-limited Pushdown Automaton* is a Pushdown Automaton whose stack can store at most k symbols (regardless of the length of the input string).

(a) Explain how a k-limited Pushdown Automaton can be simulated by a NFA.

(b) Deduce that a language is recognised by a k-limited PDA if and only if it is regular.

12. In Australian Rules Football, a team gains *points* by kicking *goals* and *behinds*. A goal is worth 6 points, and a behind is worth one point. The winner is the team with the greatest number of points at the end of the game.

A team's score is given as a triple of numbers (g, b, p), where g is the number of goals, b is the number of behinds, and p is the number of points. These numbers must be nonnegative and satisfy $6g + b = p$. For example, $(1, 2, 8)$ is a valid score, since $6 \times 1 + 2 = 8$.

Let FootyScore be the language of valid football scores (g, b, p), where each number is represented in unary, and the alphabet consists of 1, the two parentheses, and the comma. For example,

$$\texttt{(1,11,11111111)} \in \text{FootyScore}.$$

In this language, there is no upper limit on the scores.

(a) Is the language FootyScore regular? Prove or disprove.

(b) Is the language FootyScore context-free? Prove or disprove.

13. Prove by induction that the CYK algorithm correctly determines whether or not a given string x is generated by a given context-free grammar G.

More specifically:

1. Prove by induction on n that, for every substring y of x, where $|y| = n$, the algorithm correctly finds all nonterminals in G that can generate y.
2. Explain briefly how, if we know all the nonterminals that can generate x, we can then determine whether x can be generated by G.

The idea of the proof is given in Lecture 16, slide 18. Your task is to avoid the gap in that sketch proof ("Continue, in this way, ...") and produce a correct proof by induction.

14. This question is a sequel to Exercise Sheet 5, Q16.

(a) Review Exercise Sheet 5, Q16, if necessary correcting your attempt at the question.

(b) Use the Pumping Lemma for CFLs to prove that the language $\{\mathbf{a}^{n^2} : n \in \mathbb{N}\}$ is not context-free.

(c) Hence prove that the language of binary string representations of adjacency matrices of graphs is not context-free.

You may assume that the intersection of a context-free language with a regular language is context-free. (Challenge: prove this.) Note that, unlike regular languages, the class of context-free languages is not closed under intersection. (Can you prove this?)

15. A finite sequence $(x_1, x_2, \ldots, x_n)$ of positive integers is *strictly increasing* if, for all i such that $1 \le i \le n-1$, we have $x_i < x_{i+1}$.

In this question, sequences are represented as binary numbers separated by `#`. For example, the sequence $(1, 2, 5, 14, 42)$, consisting of the first five Catalan numbers[5], is represented as

```
1#10#101#1110#101010
```

Let SIS be the language, over the three-letter alphabet {`0`,`1`,`#`}, consisting of all strictly increasing sequences of positive integers, with each sequence represented as described above.

Prove that SIS is not context-free.

16.

A Stock Exchange publishes daily information on the share price of a company as a 5-tuple of numbers

$$(p, c, s, p_{\max}, p_{\min})$$

where

p	=	the price per share, in cents, when trading closed at the end of the day;
c	=	the difference between p and the published price for the previous trading day;
s	=	the number of shares sold during the day;
$p_{\max}$	=	the maximum share price for this company during the most recent 52 weeks (including this week);
$p_{\min}$	=	the minimum share price for this company during the most recent 52 weeks (including this week).

[6]

The language SHAREPRICEINFO (SPI), over the seven-symbol alphabet {`0`, `1`, `-`, `+`, `(`, `)`, `,`}, consists of all numerically valid 5-tuples of binary numbers that could (hypothetically, allowing arbitrarily high prices and arbitrary trading volumes) represent information on the share price according to the above description. We require:

[5]The n-th Catalan number counts the number of members of the Dyck language that have n pairs of matching parentheses. The Catalan number sequence has links to a wide range of topics in Computer Science and "occurs in connection with so many recursive algorithms" (Donald E. Knuth, *The Art of Computer Programming, Vol. 3, Sorting and Searching*, Addison-Wesley, Reading, Ma., 1973, p. 296.)

[6]These definitions contain some simplifications for the purposes of this exercise. In practice, Australian share prices are usually quoted in dollars, with cents given after the decimal point. But increases or decreases in share prices are indeed typically given in cents, as here.

- p, $p_{\min}$ and $p_{\max}$ can be any positive integers subject to the requirement that p lies between $p_{\min}$ and $p_{\max}$ inclusive.

- c can be any integer whose value is consistent with the above definitions of all the five numbers. If it is negative, it is prefixed by a minus sign; if it is positive, it is prefixed by a plus sign; and if it is zero, no sign is given.

- s can be any nonnegative integer. Its sole constraint is that if $s = 0$ then $c = 0$ too, since if no shares are traded then the price cannot change (since it is trading in the sharemarket that determines the price).

- Binary numbers must have no leading zeros, except that the number 0 itself is represented as a single `0`.

Is the language SPI context-free? If so, give a Context-Free Grammar for it. If not, prove that it is not, using the Pumping Lemma for Context-Free Languages.