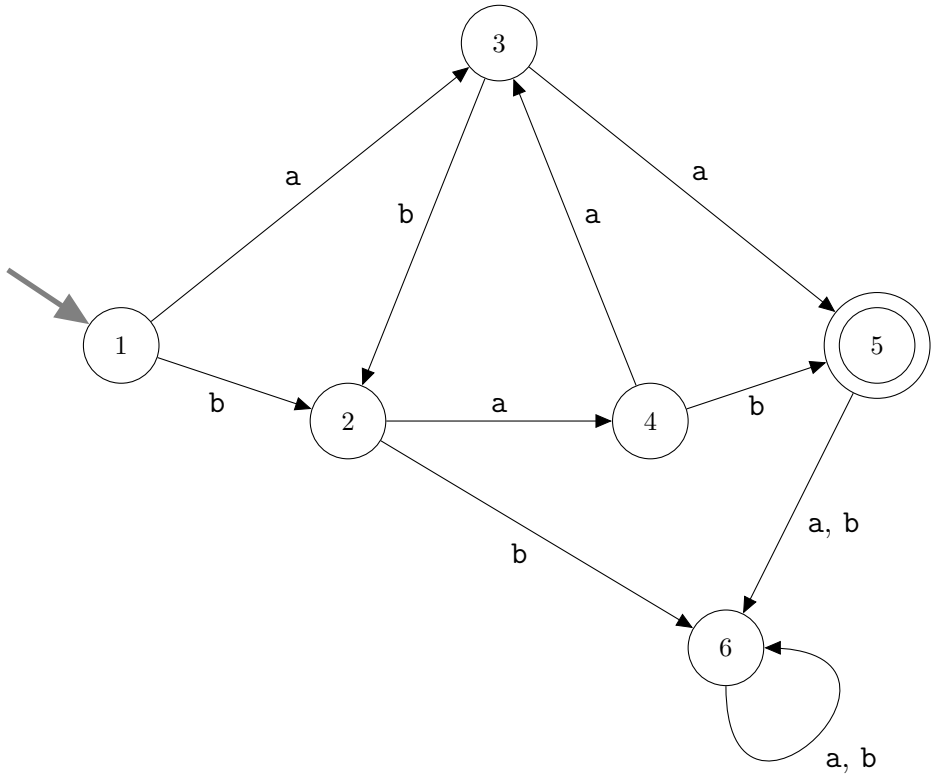
Monash University
Faculty of Information Technology
2^{nd} Semester 2023

FIT2014
Exercise Sheet 5
Pumping Lemma, and Context Free Languages

Although you may not need to do all the many exercises in this Exercise Sheet, it is still important that you attempt all the main questions and a selection of the Supplementary Exercises.

ASSESSED PREPARATION: Question 5.
You must provide a serious attempt at this entire question at the start of your class.

1.
Consider the following Finite Automaton, called SHAUN:



The **SHAUN Game** is played between two players, Reg and Nona, as follows. They take turns, with Reg moving first. The game is very short: Reg moves first, then Nona moves, then Reg moves, then Nona moves, making only four moves in total. The rules for their moves are as follows.

- Reg first chooses a number N.
- Nona chooses a string w accepted by SHAUN the Finite Automaton such that $|w| > N$.
- Then Reg divides w up into substrings x, y, z such that $y \neq \varepsilon$ and $|xy| \leq N$.
- Then Nona chooses a non-negative integer i.

The result of the game is that:

- if xy^iz is accepted by SHAUN, then Reg wins;
- if xy^iz is rejected by SHAUN, then Nona wins.

For example, here are two possible games between Reg and Nona.

- **First game:**
  1. **Reg:** chooses $N = 2$.
  2. **Nona:** chooses $w =$ `bab`.
  3. **Reg:** chooses $x = \varepsilon$, $y =$ `ba`, $z =$ `b`.
  4. **Nona:** chooses $i = 3$.

  Outcome: **Nona wins**, because $xy^iz = xy^3z =$ `bababab` is *rejected* by SHAUN.
- **Second game:**
  1. **Reg:** chooses $N = 5$
  2. **Nona:** chooses $w =$ `abaaa`.
  3. **Reg:** chooses $x =$ `a`, $y =$ `baa`, $z =$ `a`.
  4. **Nona:** chooses $i = 2$.

  Outcome: **Reg wins**, because $xy^iz = xy^2z =$ `abaabaaa` is *accepted* by SHAUN.

(a) Play this game *twice* with one of your fellow FIT2014 students (or a friend or family member, if you prefer), using different moves to those shown in the example games above, and different moves in each game. For the second game, you must reverse the roles you had in the first game. So each player plays one game as Reg, and one as Nona. Record each of the games, showing for each game:

- the name of each player, and which role they each played (Reg/Nona),
- the sequence of moves by each player,
- the outcome of the game.

(b) Using quantifiers, write down the assertion that Reg has a winning strategy. (I.e., he can choose a first move such that, no matter what Nona does in reply, Reg can choose his second move so that, for any last move by Nona, the outcome is that Reg wins.)

When doing this, assume you have a predicate $\mathsf{ShaunAccepts}(w)$ which takes a string w as its sole argument and is True if SHAUN accepts w and False if SHAUN rejects w.

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

(d) One of the players has a winning strategy. Determine who this is, and describe the winning strategy.

2.

Let SHAWN be the language represented by the regular expression $(\mathtt{a} \cup \varepsilon)(\mathtt{baa})^*(\mathtt{a} \cup (\mathtt{bab}))$.
The **SHAWN Game** is played just like the SHAUN Game, except that

- in Nona's first move, she is required to choose a string w of length $> N$ that *belongs to the language* SHAWN (instead of being accepted by the Finite Automaton SHAUN);
- the condition for Reg to win is that the final string xy^iz *belongs to the language* SHAWN (instead of being accepted by the Finite Automaton SHAUN);
- the condition for Nona to win is that the final string xy^iz *does not* belong to the language SHAWN (instead of being rejected by the Finite Automaton SHAUN).

One of the players has a winning strategy. Determine who this is, and describe the winning strategy.

3.

The HALF-AND-HALF *language* is defined by

$$\text{HALF-AND-HALF} \;=\; \{\mathtt{a}^n\mathtt{b}^n \;:\; n \in \mathbb{N}\}.$$

The **HALF-AND-HALF Game** is also played between Reg and Nona, exactly as for the SHAWN game except that the language HALF-AND-HALF is used instead. So, the string w must be chosen to belong to HALF-AND-HALF, and the outcome is determined by whether or not the string xy^iz belongs to HALF-AND-HALF. The rules for the moves are as follows.

- Reg first chooses N
- Nona chooses a string $w \in$ HALF-AND-HALF such that $|w| > N$.
- Then Reg divides w up into substrings x, y, z such that $y \neq \varepsilon$ and $|xy| \leq N$.
- Then Nona chooses a non-negative integer i.

The result of the game is that:

- if $xy^iz \in$ HALF-AND-HALF, then Reg wins;
- if $xy^iz \notin$ HALF-AND-HALF, then Nona wins.

(a) Play this game twice, for practice.
(b) One of the players has a winning strategy. Determine who this is, and describe the winning strategy.

4. The games SHAWN and HALF-AND-HALF are instances of a huge class of games called **Pumping Games**. You can play a Pumping Game for *any* language. As usual, the players are Reg and Nona. First, the players are given a language L (which may or may not be regular). Reg moves first, then Nona moves, then Reg moves, then Nona moves. The rules for their moves are as follows.

- Reg first chooses a number N
- Nona chooses a string $w \in L$ with $|w| > N$.
- Then Reg divides w up into substrings x, y, z such that $y \neq \varepsilon$ and $|xy| \leq N$.
- Then Nona chooses a non-negative integer i.

The result of the game is that:

- if $xy^iz \in L$, then Reg wins;
- if $xy^iz \notin L$, then Nona wins.

(a) Using quantifiers, write down the assertion that Reg has a winning strategy.
(b) Using quantifiers, write down the assertion that Nona has a winning strategy.
(c) What does the Pumping Lemma tell us about circumstances in which winning strategies exist?

5. Let VERY-EVEN be the language, over {0,1}, consisting of all binary representations of positive integers that (i) are multiples of 2^n and (ii) have at most $3n$ bits altogether, where $n \in \mathbb{N}$.

Note that a binary number is a multiple of 2^n if and only if its last n bits are all zero. We assume that our positive binary numbers have no leading zeros, so that their leftmost bit is always 1.

Examples of strings in this language:

- With $n = 1$, our number must be even and have at most three bits, so we have the strings 10, 100 and 110, which are binary representations of 2, 4 and 6 respectively.
- With $n = 2$, our number must be a multiple of $2^2 = 4$ and have at most six bits, so we have the strings

  100, 1000, 1100, 10000, 10100, ..., 111000, 111100,

  being binary representations of all multiples of 4 between 4 and 60 inclusive:

  4, 8, 12, 16, 20, ..., 56, 60.

Examples of strings <u>not</u> in this language:

- 1, because in this case there are $n = 0$ zeros on the right, so the length bound is $3n = 3 \times 0 = 0$, so this string is too long!
- 1000100: it has two zeros on the right, signifying that it is a multiple of $2^2 = 4$ (but not of 8). But it has length 7 which violates the length bound of $3n = 6$.

We will first use the Pumping Lemma for Regular Languages to prove, step by step, that VERY-EVEN is not regular. Then we will show that it is context-free and give a derivation using it.

(a) First, state your initial assumption about VERY-EVEN.

(b) From that assumption, you know that there exists a certain number which we usually call N. Where does N come from? What assumption can we make about it?

(c) Now design a string w of length $\geq N$ whose structure depends on N in some way.

- The intention is that the structure of w will help in the later steps of the proof. You may not know *yet* how to choose w; that's ok, you can try some different possibilities and see how they work and come back and revisit your choice of w later, if necessary.
- Make sure that w depends on N. The design of w should be such that, for any *specific* value of N, you get one *specific* string w. So you are really describing an infinite *family* of strings, parameterised by N.
- It is not necessary for your design to encompass *all possible* strings in the language. In fact, some strings will help this proof but others may not help. So you only need to design a *certain type* of string in the language that helps the proof.

(d) Now look at all possible placements of a nonempty substring y within w such that y lies within the first N bits of w. Describe these possible placements. Can you classify them into a small number of cases?

(e) Show how, for each such placement, you can find some $i \geq 0$ such that xy^iz is not in VERY-EVEN.

If your w doesn't help you do this, think about why not, and go back to step (c) and try a different w.

(f) What do you conclude, and why?

(g) Let's be evil and reconsider step (c) from a sinister angle. Suppose your enemy is trying to answer this question and they are struggling to design their string w. Which string w could you suggest to them so that the rest of their proof will not work? In order for your deception to be credible, your evil w still needs to be in the language, must still depend on N, and must have length $\geq N$.

(h) Find a Context-Free Grammar for VERY-EVEN.

(i) Using your CFG, give a derivation of the string `100100000` (the binary representation of 288).

6. Some programs (e.g., the editor **emacs**), allow a regular expression to contain $\backslash n$, where n is a digit. This must be matched by another copy of whatever substring matched the subexpression in the n-th pair of parentheses in the regular expression.

For example, suppose we have the expression **a(b*)c\1a**. This matches the string **abbbbcbbbba**, since the first **bbbb** matches what's within the first parentheses (i.e., **b***), so that any occurrence of **\1** must also be matched by that exact string, **bbbb**. The expression is not matched by the string **abbbbcbba**, since the second string of **b**'s is different to the first and therefore cannot match **\1**.

Prove or disprove: every language described by an extended regular expression of this type is regular.

7. Let CENTRAL-ONE be the language of binary strings of odd length whose middle bit is 1.
(a) Prove or disprove: CENTRAL-ONE is regular.
(b) Prove or disprove: CENTRAL-ONE is context-free.

8. Consider the following Context Free Grammar for arithmetic expressions:

$$\begin{aligned} S &\rightarrow E \\ E &\rightarrow E + T \mid E - T \mid T \\ T &\rightarrow T * F \mid T/F \mid F \\ F &\rightarrow \textbf{int} \mid (E) \end{aligned}$$

where **int** stands for any integer. Find parse trees for each of the following arithmetic expressions.

i) $4 + 6 - 8 + 9$

ii) $(4 + 10)/(8 - 6)$

iii) $(2 - 3) * (4 - 8)/(3 - 8) * (2 - 4)$

9. In this question, you will write a Context Free Grammar for a very small subset of English.

For many years, children in Victorian schools learned to read from *The Victorian Readers* (Ministry of Education, Victoria, Australia, 1928; many reprints since). Some of you may have grandparents (or even parents!) who used these books in school.

The *First Book* of this series (there were eight altogether) contains simple sentences, with illustrations. Among these sentences are:

I can hop.
I can run.
I can stop.
I am big.
I am six.
I can dig.
I can run and hop and dig.
Tom can hop and dig.
Tom is big.
Tom and I can run.

(a) Write a simple CFG in BNF which can generate these sentences and a variety of others.
(b) Using your grammar, give a derivation and a parse tree for the sentence

Tom and I can dig and hop and run.

10. Prove the following:

> If a string in a context-free language has a derivation of length n, then it has a leftmost derivation of length n.

The simplest approach is to think about parse trees ...
For a more complicated proof, by induction, see Q20.

Supplementary exercises

11.

Recall that the EVEN-EVEN *language* consists of all strings over alphabet $\{\mathtt{a}, \mathtt{b}\}$ that contain an even number of as and an even number of bs.

The **EVEN-EVEN Game** is played between two players, Reg and Nona, as follows. They take turns, with Reg moving first. The game is very short: Reg moves first, then Nona moves, then Reg moves, then Nona moves, making only four moves in total. The rules for their moves are as follows.

- Reg first chooses a number N.
- Nona chooses a string $w \in$ EVEN-EVEN such that $|w| > N$.
- Then Reg divides w up into substrings x, y, z such that $y \neq \varepsilon$ and $|xy| \leq N$.
- Then Nona chooses a non-negative integer i.

The result of the game is that:

- if $xy^iz \in$ EVEN-EVEN, then Reg wins;
- if $xy^iz \notin$ EVEN-EVEN, then Nona wins.

Here are two possible games between Reg and Nona.

- **First game:**
  1. **Reg:** first chooses $N = 4$.
  2. **Nona:** chooses $w = \mathtt{ababbaab}$.
  3. **Reg:** chooses $x = \varepsilon$, $y = \mathtt{abab}$, $z = \mathtt{baab}$.
  4. **Nona:** chooses $i = 2$.

  Outcome: **Reg wins**, because $xy^iz = xy^2z = \mathtt{abababbbaab} \in$ EVEN-EVEN.
- **Second game:**
  1. **Reg:** first chooses $N = 4$.
  2. **Nona:** chooses $w = \mathtt{ababbaab}$.
  3. **Reg:** chooses $x = \mathtt{ab}$, $y = \mathtt{ab}$, $z = \mathtt{baab}$.
  4. **Nona:** chooses $i = 0$.

  Outcome: **Nona wins**, because $xy^iz = xy^0z = \mathtt{abbaab} \notin$ EVEN-EVEN.

(a) Play this game twice, for practice.
(b) Using quantifiers, write down the assertion that Reg has a winning strategy.

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

(d) One of the players has a winning strategy, provided $N \geq 4$. Determine who this is, and describe the winning strategy.

12. Prove or disprove:

> In a Context-Free Grammar, if the right-hand side of every production is a palindrome, then any string in the generated language is a palindrome.

13. Recall that $\overleftarrow{x}$ denotes the reverse of string x. If L is any language, define $\overleftarrow{L} = \{x : \overleftarrow{x} \in L\}$, i.e., the set of all reversals of strings in x.

Prove that, if L is a Context-Free Language, then so is $\overleftarrow{L}$.

14. Use Questions 10 and 13 to prove that, if a string in a context-free language has a derivation of length n, then it has a rightmost derivation of length n.

15. For our purposes, a *permutation* is represented as a string, over the three-symbol alphabet containing 0, 1 and "," (comma), as follows: each of the numbers $1, 2, \ldots, n$ is represented in binary, and the numbers are given in some order in a comma-separated list. Each number in $1, 2, \ldots, n$ appears exactly once, in binary, in this list. For example, the permutation 3,1,2 is represented by the string "11,1,10".

Prove that the language of permutations is not regular.

16.

(a) Prove that the difference between two squares of two consecutive positive integers increases as the numbers increase.

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

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

Definitions.

The *adjacency matrix* $A(G)$ of a graph G on n vertices is an $n \times n$ matrix whose rows and columns are indexed by the vertices of G, with the entry for row v and column w being 1 if v and w are adjacent in G, and 0 otherwise.

The *binary string representation* of $A(G)$ is obtained from $A(G)$ by just turning each row of $A(G)$ into a string of n bits, and then concatenating all these strings in row order, to form a string of n^2 bits.

17.

For this question, you may use the fact that there exist arbitrarily long sequences of positive integers that are not prime. In other words, for any N, there is a sequence of numbers x, $x+1$, $\ldots$, $x+N$ none of which is prime.[1]

Prove that the language $\{\mathbf{a}^p : p \text{ is prime}\}$ is not regular.

18. Consider the following Context Free Grammar.

$$\begin{aligned} S &\rightarrow E \\ E &\rightarrow E \cup T \mid T \\ T &\rightarrow TF \mid F \\ F &\rightarrow F^* \mid (E) \mid a \mid b \mid \epsilon \end{aligned}$$

[1] Mathematically-inclined students are encouraged to try to prove this fact for themselves.

Note: in the last line, ϵ is an actual symbol (i.e., one letter); it is not the empty string itself, but rather a symbol used in regular expressions to match the empty string.

(a) Find the **leftmost** and **rightmost** derivations of the following words generated by the above Context Free Grammar.

i) $a^*b^* \cup b^*a^*$

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

iii) $(a \cup \epsilon)(b \cup \epsilon)(a \cup \epsilon)$

(b) Prove that the language generated by this grammar is not regular.

19. You saw in Q18 that the language of regular expressions, over the eight-character alphabet $\{$a,b,ε,$\cup$,(,),$*$,$\emptyset\}$, is not, itself, regular(!), but it is context-free.

Now, prove that the language of context-free grammars is regular. Assume that the non-terminal symbols are S, $X_1, \ldots X_m$, where S is the start symbol, and that the terminal symbols are $x_1, \ldots, x_n$. So the alphabet for your regular expression is $\{S, X_1, \ldots X_m, x_1, \ldots, x_n, \varepsilon, \rightarrow\}$.

But don't get confused about this! It certainly does *not* follow that every context-free language is regular! Remind yourself of the actual relationship between regular languages and context-free languages.

20. Prove the following theorem, by induction on derivation length:

> If a string in a context-free language has a derivation of length n, then it has a leftmost derivation of length n.

It may help to do this by proving the following stronger result.

> Let σ be a string, which can contain terminal and non-terminal symbols. If a string in a context-free language has a derivation from σ of length n, then it has a leftmost derivation from σ of length n.