School of Computing and Information Systems
COMP30026 Models of Computation
Week 8: Regular Expressions, Context-Free Grammars, and the Pumping Lemma for Regular Languages

Homework problems

P8.1 Give regular expressions for the following languages over the alphabet $\Sigma = \{0, 1\}$.

(i) $\{w \mid w$ has length at least 3 and its third symbol is $\mathtt{0}\}$

(ii) $\{w \mid$ every odd position of w is a $\mathtt{1}\}$

(iii) $\{\epsilon, \mathtt{0}\}$

(iv) The empty set

Bonus: Draw minimal NFAs for these languages, and those from **??**

Solution:

(i) $(0 \cup 1)(0 \cup 1)0(0 \cup 1)^*$

(ii) $(1(0 \cup 1))^*(\epsilon \cup 1)$

(iii) $\epsilon \cup 0$

(iv) $\emptyset$

P8.2 String s is a *suffix* of string t iff there exists some string u (possibly empty) such that $t = us$. For any language L we can define the set of suffixes of strings in L:

$$\mathit{suffix}(L) = \{x \mid x \text{ is a suffix of some } y \in L\}$$

Let A be any regular language. Show that $\mathit{suffix}(A)$ is a regular language. *Hint:* think about how a DFA for A can be transformed to recognise $\mathit{suffix}(A)$.

Solution: If A is regular then $\mathit{suffix}(A)$ is regular. Namely, let $D = (Q, \Sigma, \delta, q_0, F)$ be a DFA for A. Assume every state in Q is *reachable* from q_0. Then we can turn D into an NFA N for $\mathit{suffix}(A)$ by adding a new state q_{-1} which becomes the NFA's start state. For each state $q \in Q$, we add an epsilon transition from q_{-1} to q.

That is, we define N to be $(Q \cup \{q_{-1}\}, \Sigma, \delta', q_{-1}, F)$, with transition function

$$\delta'(q, x) = \begin{cases} \{\delta(q, x)\} & \text{for } q \in Q \text{ and } x \in \Sigma \\ Q & \text{for } q = q_{-1} \text{ and } x = \epsilon \\ \emptyset & \text{for } q \neq q_{-1} \text{ and } x = \epsilon \end{cases}$$

The restriction we assumed, that all of D's states are reachable, is not a severe one. It is easy to identify unreachable states and eliminate them (which of course does not change the language of the DFA). To see why we need to eliminate unreachable states before generating N in the suggested way, consider what happens to this DFA for $\{\epsilon\}$: $(\{q_0, q_1, q_2\}, \{\mathtt{a}\}, \delta, q_0, \{q_0\})$, where $\delta(q_0, \mathtt{a}) = \delta(q_1, \mathtt{a}) = q_1$ and $\delta(q_2, \mathtt{a}) = q_0$.

P8.3 In general it is difficult, given a regular expression, to find a regular expression for its complement. However, it can be done, and you have been given all the necessary tricks and algorithms. This question asks you to go through the required steps for a particular example.

Consider the regular language $(\mathtt{ba}^*\mathtt{a})^*$. Assuming the alphabet is $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, we want to find a regular expression for its complement, that is, for

$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ is not in } (\mathtt{ba}^*\mathtt{a})^*\}$$

To complete this task, go through the following steps.

(a) Construct an NFA for $(\mathtt{ba}^*\mathtt{a})^*$. Two states suffice.

(b) Turn the NFA into a DFA using the subset construction method.

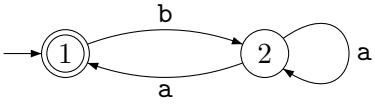
(c) Do the "complement trick" to get a DFA D for L.

(d) Reflect on the result: Wouldn't it have been better/easier to apply the "complement trick" directly to the NFA?

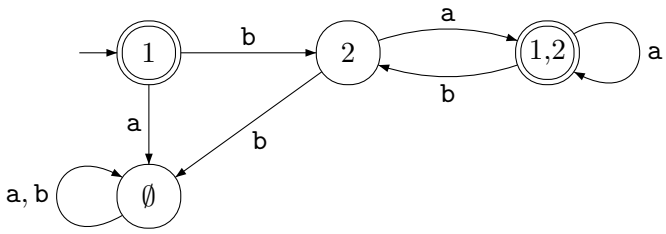
(e) Turn DFA D into a regular expression for L using the NFA-to-regular-expression conversion process shown in the lecture on regular expressions.

Solution:

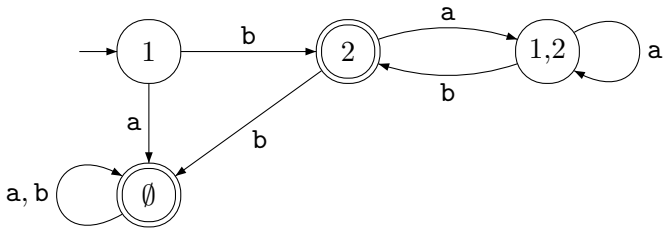
(a) Here is an NFA for $(\mathtt{ba}^*\mathtt{a})^*$:



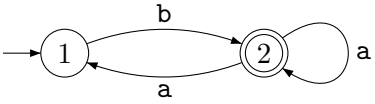
(b) Here is an equivalent DFA, obtained using the subset construction:



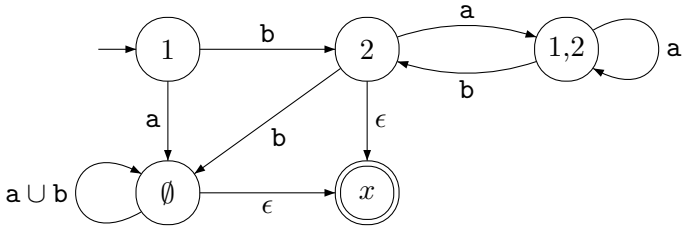
(c) It is easy to get a DFA for the complement:



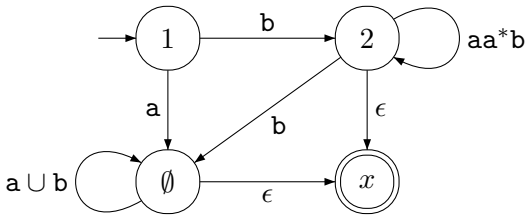
(d) It would be problematic to do the "complement trick" on the NFA, as it is only guaranteed to work on DFAs. We would get the following, which accepts, for example, baa:



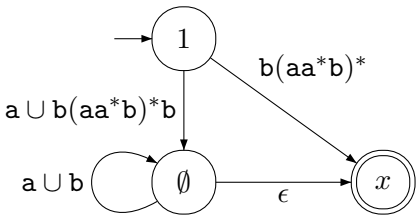
(e) Starting from the DFA that we found in (c), we make sure that we have just one accept state:



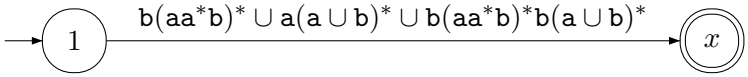
Let us first remove the state labeled 1,2:



Now we can remove state 2:



Finally, eliminating the state labelled $\emptyset$, we are left with:



The resulting regular expression can be read from that diagram.

P8.4 A *palindrome* is a string that reads the same forwards and backwards. Use the pumping lemma for regular languages and/or closure results to prove that the following languages are not regular:

(a) $B = \{\mathtt{a}^i\mathtt{b}\mathtt{a}^j \mid i > j \geq 0\}$

(b) $C = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ is not a palindrome}\}$

(c) $D = \{www \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$

Solution:

(a) We use the pumping lemma to show that B is not regular. Assume that it were. Let p be the pumping length, and consider $\mathtt{a}^{p+1}\mathtt{b}\mathtt{a}^p$. This string is in B and of length $2p+2$. By the pumping lemma, there are strings x, y, and z such that $\mathtt{a}^{p+1}\mathtt{b}\mathtt{a}^p = xyz$, with $y \neq \epsilon$, $|xy| \leq p$, and $xy^iz \in B$ for all $i \in \mathbb{N}$. By the first two conditions, y must be a non-empty string consisting of as only. But then, pumping down, we get xz, in which the number of as on the left no longer is strictly larger than the number of as on the right. Hence we have a contradiction, and we conclude that B is not regular.

(b) We want to show that $C = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ is not a palindrome}\}$ is not regular. But since regular languages are closed under complement, it will suffice to show that $C^c = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ is a palindrome}\}$ is not regular. Assume that C^c *is* regular, and let p be the pumping length. Consider $\mathtt{a}^p\mathtt{b}\mathtt{a}^p \in C^c$. Since $|s| \geq p$, by the pumping lemma, s can be written $s = xyz$, so that $y \neq \epsilon$, $|xy| \leq p$, and $xy^iz \in C^c$ for all $i \geq 0$. But since $|xy| \leq p$, y must contain as only. Hence $xz \notin C^c$. We have a contradiction, and we conclude that C^c is not regular. Now, if C was regular, C^c would be regular too. Hence C cannot be regular.

(c) Suppose to the contrary that D is regular, and let p be the pumping length of D. Consider the string $s = (\mathtt{a}^p\mathtt{b})^3 \in D$. Since $|s| \geq p$, by the pumping lemma, s can be written $s = xyz$, so that $y \neq \epsilon$, $|xy| \leq p$, and $xy^iz \in D$ for all $i \geq 0$. In particular, we

have $xz \in D$. Now, since $|xy| \leq p$, the string y must consist entirely of a's. Therefore $xz = \mathtt{a}^{p-|y|}\mathtt{b}(\mathtt{a}^p\mathtt{b})^2 \notin D$ since $|y| \geq 1$ and hence there are fewer a's before the first b than in the other two copies. Contradiction! Hence D is not regular.

P8.5 Give context-free grammars for the following languages. Assume the alphabet is $\Sigma = \{0, 1\}$.

(i) $\{w \mid w$ starts and ends with the same symbol$\}$

(ii) $\{w \mid$ the length of w is odd$\}$

Solution:

(a)

$$\begin{array}{rcl} S & \rightarrow & 0\ T\ 0 \mid 1\ T\ 1 \mid 0 \mid 1 \\ T & \rightarrow & 0\ T \mid 1\ T \mid \epsilon \end{array}$$

(b)

$$S \rightarrow 0 \mid 1 \mid 0\ 0\ S \mid 0\ 1\ S \mid 1\ 0\ S \mid 1\ 1\ S$$

P8.6 Construct a context-free grammar for the language $\{\mathtt{a}^i\mathtt{b}\mathtt{a}^j \mid i > j \geq 0\}$.

Solution:

$$\begin{array}{rcl} S & \rightarrow & A\ B \\ A & \rightarrow & \mathtt{a} \mid \mathtt{a}\ A \\ B & \rightarrow & \mathtt{b} \mid \mathtt{a}\ B\ \mathtt{a} \end{array}$$

P8.7 Show that the class of context-free languages is closed under the regular operations: union, concatenation, and Kleene star. *Hint:* show how context-free grammars for A and B can be manipulated to produce context-free grammars for $A \cup B$, AB, and A^*. Careful: The variables used in the grammars for A and in B could overlap.

P8.8 If we consider English words the "symbols" (or primitives) of English, we can use a context-free grammar to try to capture certain classes of sentences and phrases. For example, we can consider articles (A), nouns (N), adjectives (Q), intransitive verbs (IV), transitive verbs (TV), noun phrases (NP), verb phrases (VP), and sentences (S). List 5–10 sentences generated by this grammar:

S	$\rightarrow$	$NP\ VP$	N	$\rightarrow$	cat	IV	$\rightarrow$	hides
A	$\rightarrow$	a	N	$\rightarrow$	dog	IV	$\rightarrow$	runs
A	$\rightarrow$	the	Q	$\rightarrow$	lazy	IV	$\rightarrow$	sleeps
NP	$\rightarrow$	$A\ N$	Q	$\rightarrow$	quick	TV	$\rightarrow$	chases
NP	$\rightarrow$	$A\ Q\ N$	VP	$\rightarrow$	IV	TV	$\rightarrow$	hides
N	$\rightarrow$	bone	VP	$\rightarrow$	$TV\ NP$	TV	$\rightarrow$	likes

Are they all meaningful? Discuss "well-formed" versus "meaningful".

Solution: Here are some sentences generated from the grammar:

- A dog runs
- A dog likes a bone
- The quick dog chases the lazy cat
- A lazy bone chases a cat
- The lazy cat hides
- The lazy cat hides a bone

The grammar is concerned with the structure of well-formed sentences; it says nothing about meaning. A sentence such as "a lazy bone chases a cat" is syntactically correct—its structure makes sense; it could even be semantically correct, for example, "lazy bone" may be a derogatory characterisation of some person. But in general there is no guarantee that a well-formed sentence carries meaning.

P8.9 How would you change the grammar from the previous question so that "adverbial modifiers" such as "angrily" or "happily" can be used? For example, we would like to be able to generate sentences like "the dog barks constantly" and "the black cat sleeps quietly".

Solution: We can easily extend the grammar so that a sentence may end with an optional adverbial modifier:

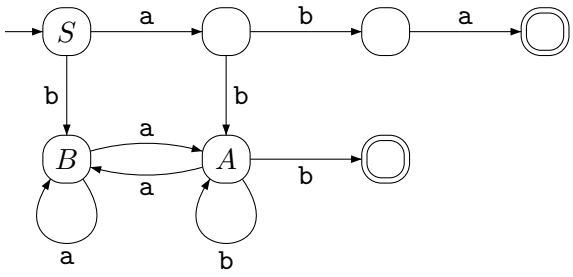
$$\begin{array}{lcl} S & \to & NP\ VP\ PP \\ & \vdots & \\ PP & \to & \epsilon \\ PP & \to & \text{quietly} \\ PP & \to & \text{all day} \\ & \vdots & \end{array}$$

P8.10 Consider this context-free grammar G with start symbol S:

$$\begin{array}{lcl} S & \to & \texttt{a b a} \mid \texttt{a b}\ A \mid \texttt{b}\ B \\ A & \to & \texttt{b} \mid \texttt{b}\ A \mid \texttt{a}\ B \\ B & \to & \texttt{a}\ A \mid \texttt{a}\ B \end{array}$$

Draw an NFA which recognises $L(G)$. *Hint:* The grammar is a regular grammar; you may want to use the labels S, A, and B for three of the NFA's states.

Solution: Here is a suitable NFA:



P8.11 Consider the context-free grammar G with rules

$$S \to \texttt{a b} \mid \texttt{a}\ S\ \texttt{b} \mid S\ S$$

Use structural induction to show that no string $w \in L(G)$ starts with `abb`.

Solution: Let w be a string in $L(G)$, that is, w is derived from S. We will use structural induction to show a stronger statement than what was required; namely we show that, for every string $w \in L(G)$, w starts with neither `b` nor `abb`. That is, if w is derived from S then it starts with neither `b` nor `abb`. (To express the property formally, we may write $\forall w' \in \{\texttt{a}, \texttt{b}\}^*(w \neq \texttt{b}w' \wedge w \neq \texttt{abb}w')$.

There is one base case: If $w =$ `ab` then w does not start `b` and it does not start with `abb`.

For the first recursive case, let $w = \texttt{a}w'\texttt{b}$, where $w' \in L(G)$. By the induction assumption, w' starts with neither `b` nor `abb`. Hence, in this case, w does not start with `b` (because it starts with `a`), and it does not start with `abb` (because w' does not start with `b`).

For the second recursive case, let $w = w'w''$, with $w', w'' \in L(G)$. By the induction assumption, w' starts with neither `b` nor `abb` (similarly for w''). Let us do case analysis on the length of w'.

- $|w'| = 0$: If $w' = \epsilon$ then $w = w''$ which starts with neither `b` nor `abb`, by assumption.
- $|w'| = 1$: In this case we must have $w' =$ `a` since, by assumption, w' does not start with `b`. But then w doesn't start with `b`, and it doesn't start with `abb` either, because w'' does not start with `b`, by assumption.
- $|w'| \geq 2$: In this case w' must start with either `aa` or `ab`. That means w does not start with `b`. And w can only start with `abb` if $w' =$ `ab` and w'' starts with `b`. But the latter is impossible, by assumption.

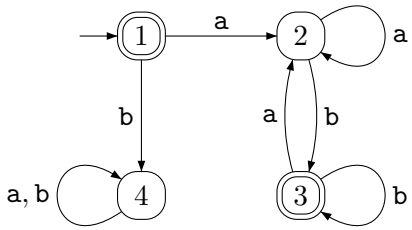
Hence in no case does w start with `abb`.

P8.12 Consider the context-free grammar $(\{S\}, \{\texttt{a}, \texttt{b}\}, R, S)$ with rules R:

$$S \;\rightarrow\; \texttt{a} \mid \texttt{b} \mid S\ S$$

Show that the grammar is ambiguous; then find an equivalent unambiguous grammar.

P8.13 Give a context-free grammar for the language recognised by this DFA:



Solution: Here is a context-free grammar that will do the job (S is the start symbol):

$$\begin{array}{lcl} S & \rightarrow & \epsilon \mid \texttt{a}\ A \\ A & \rightarrow & \texttt{a}\ A \mid \texttt{b}\ B \\ B & \rightarrow & \epsilon \mid \texttt{a}\ A \mid \texttt{b}\ B \end{array}$$