The University of Melbourne
School of Computing and Information Systems
COMP30026 Models of Computation

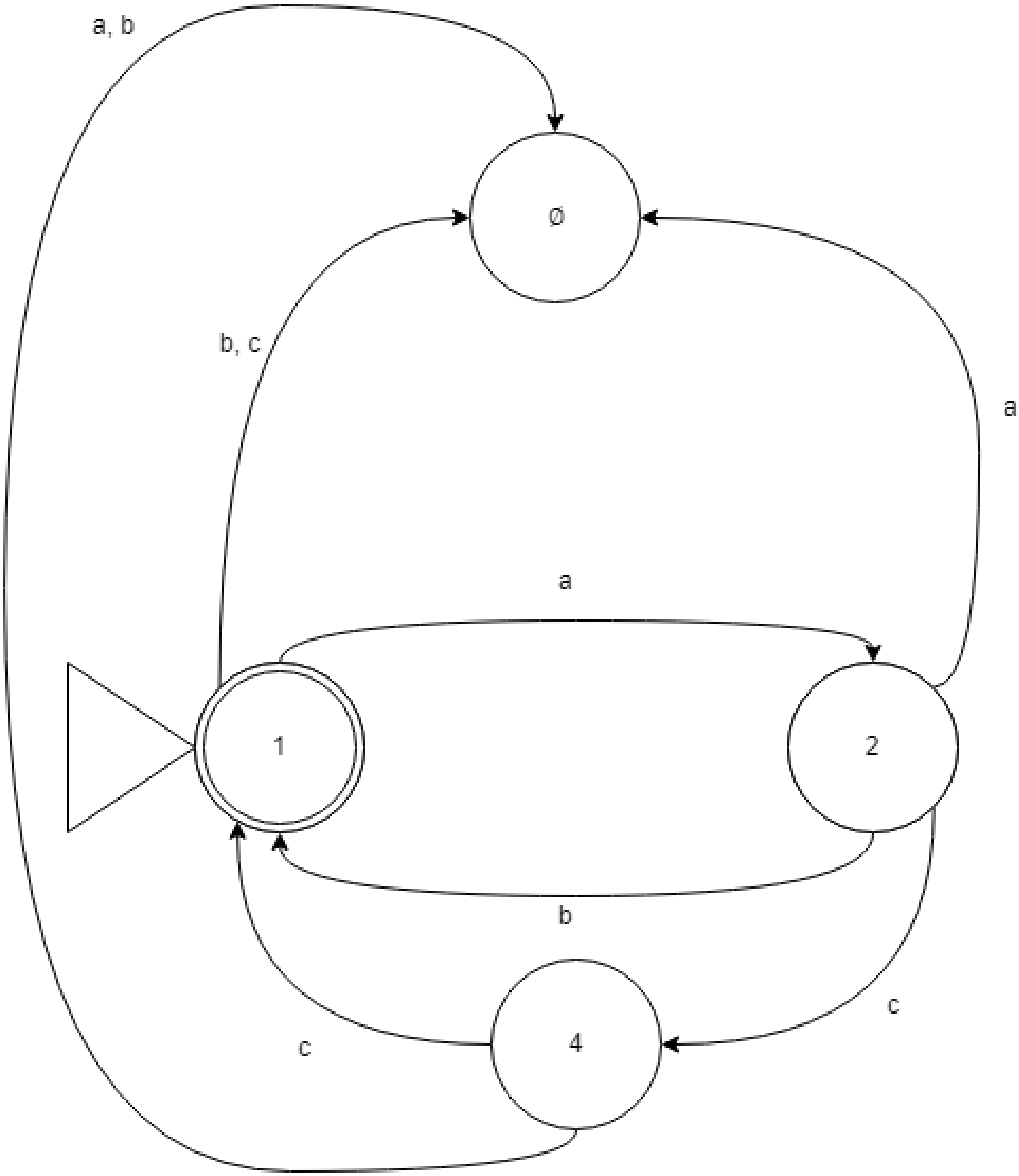
Assignment 2, 2018

Samuel Xu
Student No. 835273

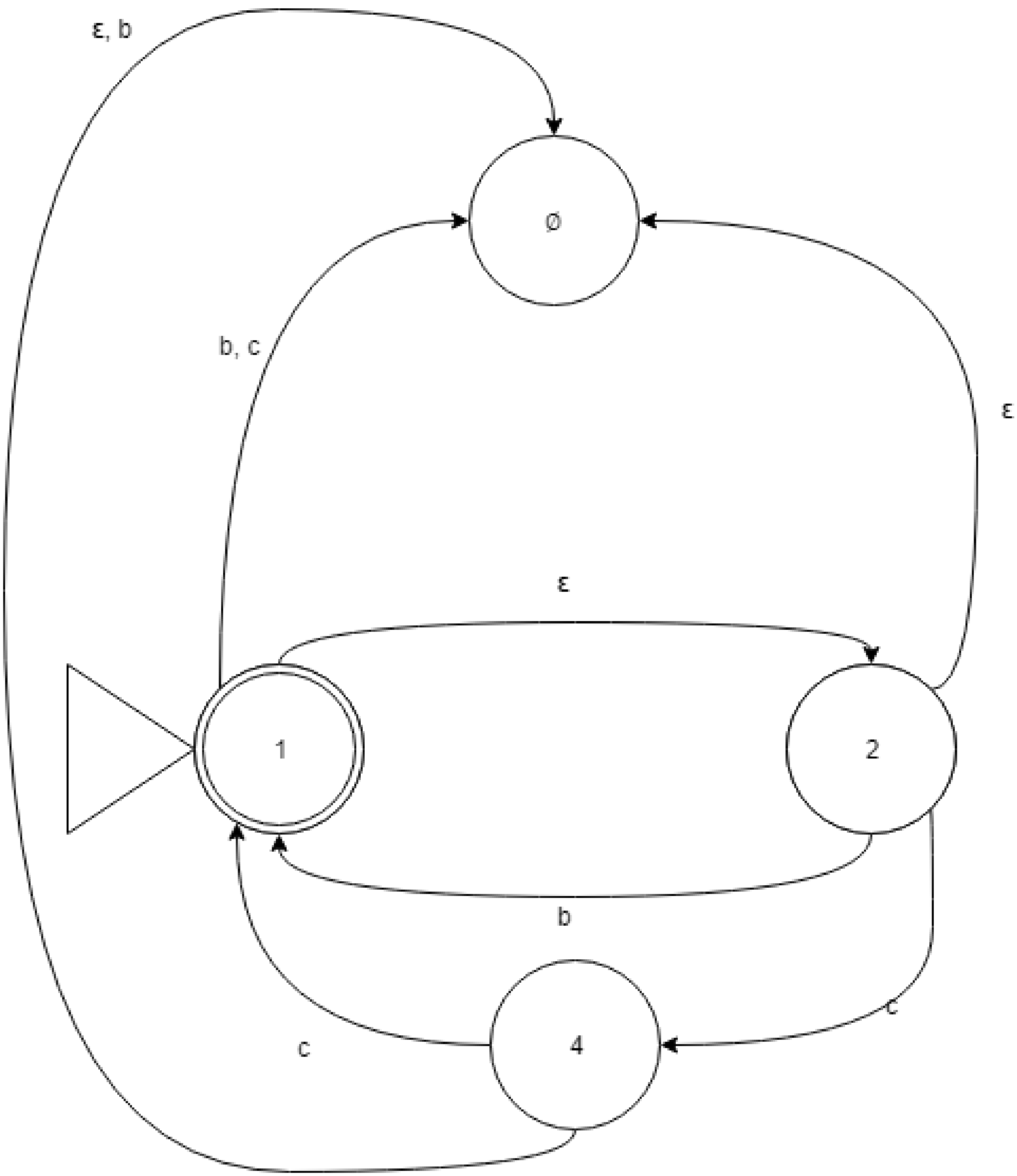
Challenge 4 Answer Part A

To show that $drop(a, R)$ is a regular language, let R be a regular language, with a finite non-empty alphabet Σ. Then, consider the $DFA(Q, \Sigma, \delta, q_0, F)$ for regular language R. When we apply the $drop(a, R)$ to the DFA, we simply replace every transition containing a with a ϵ arc. This creates an NFA, which means that $drop(a, R)$ must be regular (as any language is regular when it is able to be defined by an automata).

Example: Consider the language R' {acc, abab, ababab, accacc}, with the DFA as shown below:



After removing all a arcs, we have the NFA for drop(b, R') {acc, accacc}:



Challenge 4 Answer Part B

Let there be a finite alphabet for language L, where $\Sigma = \{a, b, c\}$
Let L be the regular language $L = \{ab^ic^i | i > 0\}$
We therefore defined $drop(a, L)$ as $\{b^ic^i | i > 0\}$, which is a context free *non-regular* language as defined in lecture 18.
$drop(b, L)$, however, will result in the language $\{ac^i | i > 0\}$ which *is* a regular language.

Challenge 5 Answer

Let $DFA(Q, \Sigma, \delta, q_0, F)$ and $PDA(Q', \Sigma', \Gamma, \delta', q_0', F')$

The goal is to translate our DFA to a 3-state PDA. This can be accomplished by creating a PDA similar to that of a naive search or graph traversal.
This process is possible as there are only two cases to account for:

a. The current state is *not* an accept state

b. The current state is an accept state

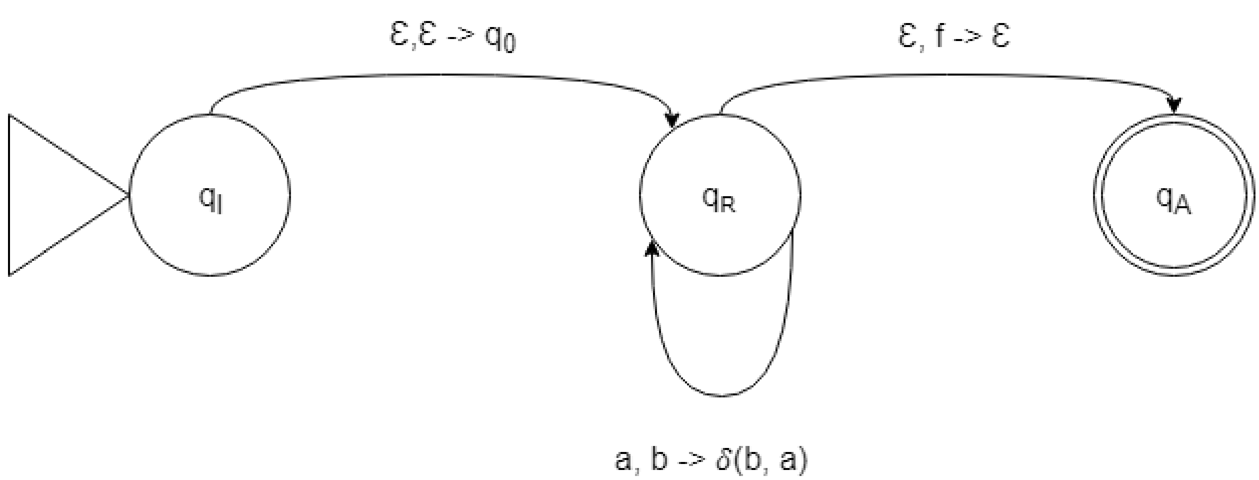
This results in a translation from any DFA to a 3-state PDA; the first state of the PDA being the *initialisation state* (q_I), followed by the *searching state* (q_R) and the *accept state* (q_A).

First we start by defining our variables:

DFA:
Let Q be a set of DFA states.
Let Σ be a valid alphabet for our DFA.
Let δ be a valid set of transition functions for moving between states with a given input.
Let q_0 be the initial state of our DFA.
Let F be a set of valid accept states for our DFA.

PDA:
Let $Q' = \{q_I, q_R, q_A\}$
Let $\Sigma' = \Sigma_\epsilon$
Let Γ be a stack alphabet where $\Gamma = Q$
Let δ' be the three transition functions between q_I, q_R, and q_A
(following the form described in lecture 18 $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$;
$\delta : \{(q_I, \epsilon, \epsilon, \to q_R, q_0), (q_R, a, b, \to q_R, \delta(b, a)), (q_R, \epsilon, f \to q_A, \epsilon)\}$
Let q_0' be the starting state q_I
Let F' be the accept state $\{q_A\}$
Let a be a string input where $a \in \Sigma$
Let b be a state where $b \in Q$
Let f be an accept state where $f \in F$

Now we may draw a PDA:



Challenge 6 Answer Part A

To show that $L(G) \subseteq A$, we can use structural induction. First we should consider the base case, which can be defined by the rule $S \rightarrow \epsilon$. This, of course implies that $\epsilon \in L(G)$. Additionally, we can assume for the sake of this question that $L(G) = A$, from which we can also derive $\epsilon \in A$. From this we now have the base cases as defined below:

a. $\epsilon \in L(G)$

b. $\epsilon \in A$

We may now walk through each rule, disputing that "001" is *not* valid within $L(G)$.

- $S \rightarrow S0$

In this rule, this implies that the input *ends with 0*, which "001" does *not*. Then we may assume that "001" is not in the subset of this rule $S0$.

- $S \rightarrow 1S$

In this rule, this implies that the *input starts with 1*, which "001" does *not*. Then we may assume that "001" is not in the subset of this rule $1S$.

- $S \rightarrow 01S$

In this rule, this implies that the *input starts with 01*, which "001" does *not*. Then we may assume that "001" is not in the subset of this rule $01S$.

As all of the context free grammar rules *do not allow for* "001" *to exist in* S, therefore "001" *cannot be a substring in* $L(G)$.

Therefore, $L(G) \subseteq A$

Challenge 6 Answer Part B

To prove $A \subseteq L(G)$ by induction on the length of strings, we can use the predefinted values as shown in the question (u, x, v), where a substring x of string w is defined by $w = uxv$ for some $u, v \in \Sigma^*$.
Now following the method as discussed in the discussion forum[1]; the base case can remain the same, where all strings with a length of 0 (i.e. ϵ) are in A and L(G).

a. $|w| = 0$, $w \in L(G)$

b. $|w| = 0$, $w \in A$

We can then attempt induction for all cases where w is not ϵ, as described by $w = uxv$ (i.e. "001" is not a substring).
Let $|w| = k$

Consider the following cases:

[1] RE: Challenge 6; Show A is a subset/equivalent set of L(G) by induction on length of strings in A posted by Sebastian Winter: `https://app.lms.unimelb.edu.au/webapps/discussionboard/do/message?action=list_messages&course_id=_372837_1&nav=discussion_board_entry&conf_id=_761662_1&forum_id=_436208_1&message_id=_1819563_1`

- Strings w where $k < 3$

These are all valid as the substring —001— is larger than k. Therefore it is impossible to have the substring 001 within w, and $A \subseteq L(G)$ holds for all w where $k < 3$.

- Strings w where $k \geqslant 3$

This case is *recursive*. We can prove that $A \subseteq L(G)$ holds by showing that if w holds at length k, we can construct a string of length $k+1$ that also holds by using the rules defined in $L(G)$ (as shown in part a):

- $w0 \in L(G)$
- $1w \in L(G)$
- $01w \in L(G)$

From this, we can accept any string that is accepted by $L(G)$. Therefore, $A \subseteq L(G)$.

Therefore, from part a) and part b),
$L(G) \subseteq A$ and $A \subseteq L(G)$
$\therefore L(G) = A$