RMIT UNIVERSITY

Computing Theory
COSC 1107/1105
Final Exercise

1 Assessment details

1. Consider the grammar derivations below.

   $S \Rightarrow 00S1 \Rightarrow 0000S11 \Rightarrow 00000S111 \Rightarrow 00000\#A\#111 \Rightarrow 00000\#B\#111 \Rightarrow 00000\#C\#111 \Rightarrow 00000\#cDd\#111 \Rightarrow 00000\#c2d\#111$

   $S \Rightarrow \#A\# \Rightarrow \#aAb\# \Rightarrow \#aaBbb\# \Rightarrow \#aacCdbb\# \Rightarrow \#aaccCddbb\# \Rightarrow \#aacccDdddbb\# \Rightarrow \#aaccc3ddbb\#$

   $S \Rightarrow 0S1 \Rightarrow 0\#A\#1 \Rightarrow 0\#aaAbb\#1 \Rightarrow 0\#aaBbb\#1 \Rightarrow 0\#aaCbb\#1 \Rightarrow 0\#aacCddbb\#1 \Rightarrow 0\#aaccDdddbb\#1 \Rightarrow 0\#aacc2ddbb\#1$

   (a) From the above derivations, construct rules that must exist in any context-free grammar G for which these derivations are correct. (2 marks)

   Answer:

   $S \rightarrow 00S1 \mid 0S1 \mid \#A\#$
   $A \rightarrow aAb \mid B \mid aaAbb \mid aBb$
   $B \rightarrow cCd \mid C$
   $C \rightarrow cCd \mid cCdd \mid cDd$
   $D \rightarrow 2 \mid 3$

   It is also possible to have some equivalent grammars without redundant rules, such as the one below.

   $S \rightarrow 00S1 \mid 0S1 \mid \#A\#$
   $A \rightarrow aAb \mid C$
   $C \rightarrow cCd \mid cCdd \mid cDd$
   $D \rightarrow 2 \mid 3$

   (b) Assuming that these are all the rules in G, give $L(G)$ in set notation. (3 marks)

   Answer: $L = \{0^i\#a^jc^kwd^lb^j\#1^m \mid 2m \geq i \geq m, k \leq l \leq 2k, k, l \geq 1, i, j, m \geq 0, w \in \{2, 3\}\}$

   (c) Is the grammar G context-sensitive? Explain your answer. (2 marks)

   Answer: Yes. This grammar is context-free, so it is also context-sensitive.

   (d) What happens to $L(G)$ if the rule $S \rightarrow SS$ is added to G? Explain your answer (but there is no need to give the new language in set notation). (3 marks)

   Answer: This now allows a sequence of words from the previous language, such as $00\#c3d\#100\#cc3dd\#11$. But it also allows intermediate strings like $00SSS11$ and hence $000S10S100S111$. This is almost the same as a sequence of words from the previous language except that the outermost 0's and 1's may not balance, eg $000S10S100S111$ is $000S1$ then $0S1$ then $00S111$. So it is not very simple to unravel this.

(e) Construct a context-free grammar for the language L_2 below. (4 marks)

$L_2 = \{a^i\ 0^j\ w\ c^l\ e\ d^l\ w^R\ 1^k\ b^{2i} \mid w \in \{2,3\}^*, k \geq j, i, j, k \geq 0, l \geq 1\}$

Answer: One such grammar is below.

$S \to aSbb \mid A$
$A \to 0A1 \mid A1 \mid B$
$B \to 2B2 \mid 3B3 \mid cCd$
$C \to cCd \mid e$

(f) Explain why there is not necessarily a deterministic pushdown automaton for L_2. (2 marks)

Answer: There is always a nondeterministic pushdown automaton for any context-free language. But as it is not always possible to convert a nondeterministic pushdown automaton into a deterministic one, there is not necessarily a deterministic pushdown automaton for a given context-free language.

(2+3+2+3+4+2 = 16 marks)

2. The following discussion was discovered in some ancient archives. There are between 6 and 12 incorrect statements in the paragraph below. Identify all incorrect statements and justify each of your answers.

Note: A single sentence counts as one statement.

"The Chomsky Hierarchy is a way of classifying languages and the classes of grammars that generate them.

The distinct classes of grammar are labelled as Type 0 to Type 4 inclusive.

Each class of grammar is distinct, meaning that no grammar appears in more than one class.

For each class of grammar there is a corresponding class of automata, and these automata can be either deterministic or non-deterministic.

For every nondeterministic automaton in any of these classes there is an equivalent deterministic automaton which recognises the same language.

It is also the case that for every possible input for every automaton in these classes, the computation on that input always terminates, although this may take a very long time.

This means that we can divide the decidable problems into two classes – the tractable ones, which can be decided in at most polynomial time on a deterministic Turing machine, and the intractable ones which take at least exponential time on a deterministic Turing machine.

Some well-known examples of intractable problems include the vertex cover problem, the Hamiltonian circuit problem, the Travelling Salesperson problem and the Halting problem for Turing

It is remarkable that many of these intractable problems are related by being NP-complete, which means they are all of a similar level of complexity.

The problems in NP are those which can be solved in polynomial space on a nondeterministic Turing machine.

To be NP-complete, a problem has be in NP, but also be at least as hard as every other problem in NP.

This property is shown by a process known as reduction, in which a problem is shown to be NP-complete by showing that it can be reduced to some already known NP-complete problem.

Starting with a theorem from the early 1970's, which showed that 3SAT is NP-complete, a large library of NP-complete problems and their relationships has been built up over the years.

Known NP-complete problems include the knapsack problem, the bin packing problem, integer factorisation and the 3-colouring problem.

It is strongly suspected that all NP-complete problems are intractable, i.e. cannot be solved in polynomial time. In order to show that P $\neq$ NP, it would be sufficient to show that one NP-complete problem is certainly intractable. NP-complete problems also have the intriguing property that a purported solution can be checked in polynomial time, which makes them potentially useful for zero-knowledge proofs. " (12 marks)

Answer: The 8 errors are explained below.

1 There are only 4 classes of grammar, which are labelled Type 0 to Type 3 inclusive.

2 The hierarchy does not have distinct classes; Type 3 grammars are also Type 2 (but not vice-versa), Type 2 are also Type 1, and Type 1 are also Type 0.

3 Not every nondeterministic automaton has a deterministic equivalent; in particular, a non-deterministic pushdown automaton does not necessarily have a deterministic equivalent.

4 Not every computation on every input always terminates; specifically for Turing machines, this is not true.

5 The Halting problem for Turing machines is undecidable, and hence it is not intractable.

6 The problems in NP are those which can be solved in polynomial time on a nondeterministic Turing machine.

7 A problem is shown to be NP-hard (a key aspect of showing it is NP-complete) by showing that some existing NP-complete problem can be reduced to it.

8 It is not known whether integer factorisation is NP-complete or not.

3. (a) Prove that Empty Language problem for Turing machines is undecidable by reducing the Halting problem to it. Make sure all the details of the reduction are clearly specified. You may use a diagram to assist with this if you wish. (4 marks)

Answer: Suppose there does exist a Turing machine *Empty* which solves the Empty Language problem, ie that *Empty* is able to decide if the language of a Turing machine is empty.

Let us build a Turing machine *Halt* as follows:

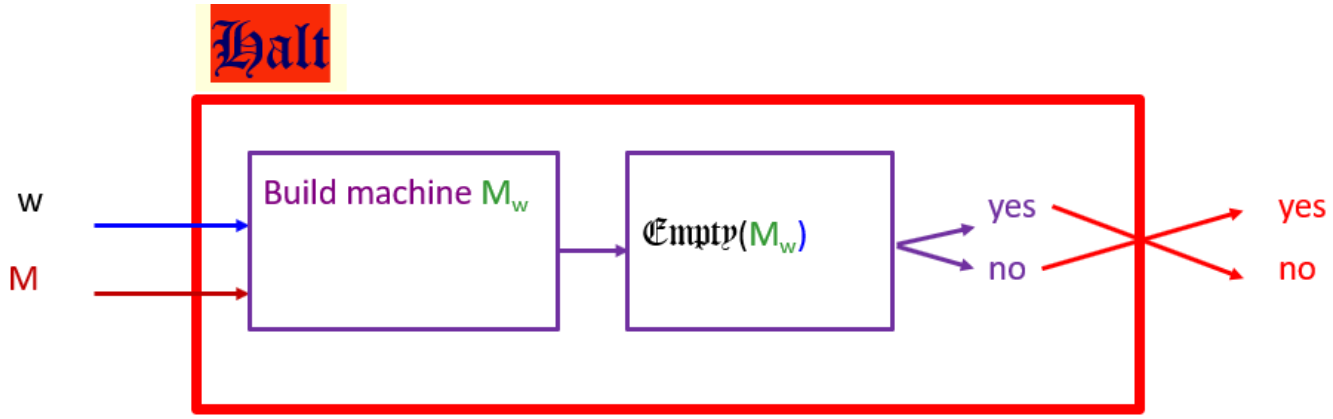
- First machine *Halt* takes M and builds a machine M^w that deletes its tape, writes w, and then simulates M on w. If M halts on w then M^w accepts.
- Then *Halt* runs *Empty* on M^w and accepts if *Empty* rejects (and rejects if *Empty* accepts).

We next show that *Halt* solves the Halting Problem.

- Suppose that M halts on input w. Then M^w will always accept, so *Empty* will reject M^w; which means that *Halt* accepts M^w.
- Suppose that M does not halt on w. Then M^w will never halt, and so the language of M^w is empty. So *Empty* will accept M^w and so *Halt* rejects M^w.

This means that *Halt* solves the Halting Problem, a contradiction. Hence, *Empty* cannot exist and the problem is undecidable. **QED**

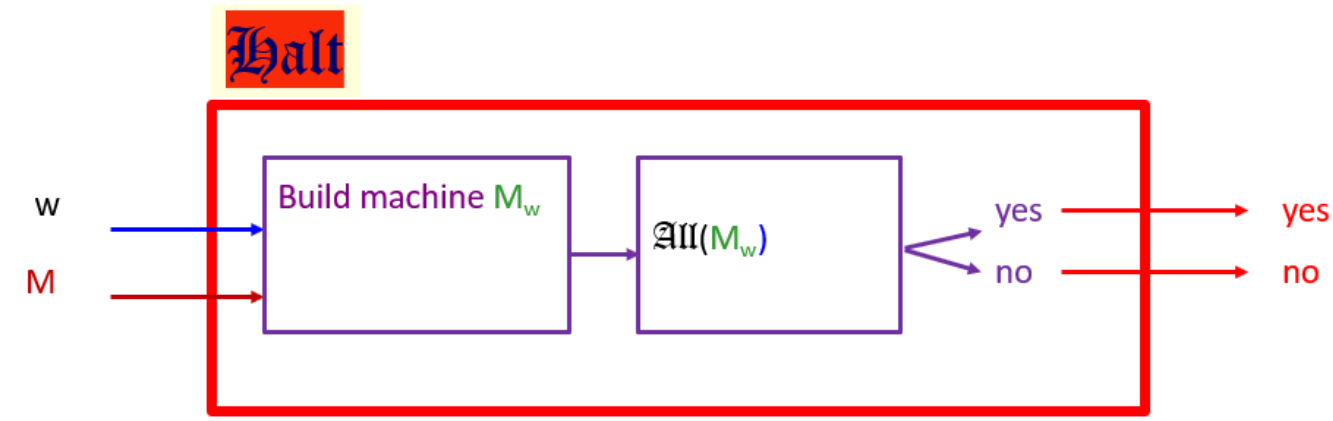
The reduction looks like this (note that this predates the Reductions notes, and hence has slightly different notation).



(b) A similar problem is the All Inputs problem for Turing machines. How similar is the proof for the undecidability of this problem to the above proof for the empty language problem? Explain your answer. (4 marks)

Answer: The proof is very similar, with a slight modification in the reduction. This time we have a machine *All*, and the *Halt* machine accepts M^w if the machine *All* accepts M^w and rejects M^w if *All* rejects M^w.

The reduction looks like this (note that this predates the Reductions notes, and hence has slightly different notation).



(c) The sun is around 4.5 billion years old, and is expected to use up its hydrogen reserves (leading to it becoming a red giant) in about 5 billion years. The Tower of Hanoi is a game that is known to require $2^n - 1$ moves to complete for a tower of height n. The 4-player Platypus game is known to require $n(n+1)(n+2)(n+3)/24$ matches for a tournament with n entries. Assuming that moves in the Tower of Hanoi take 0.1 seconds each and $n = 64$, and 4-player Platypus matches take 0.001 seconds each and $n = 500,000$, calculate which

process is more complete by the time the sun uses up its hydrogen reserves. In other words, at that time, which will be greater – the percentage of the Tower of Hanoi moves that are completed out of the total when $n = 64$ or the percentage of the matches for a tournament of n machines that are completed when $n = 500,000$? Explain your answer and include all relevant calculations. (4 marks)

Answer: The total time available is $5 \times 10^9 \times 365.25 \times 24 \times 60 \times 60$ seconds, or 157,788,000,000,000,000 seconds.

For the Tower of Hanoi, this is 1,577,880,000,000,000,000 moves out of a total of $2^{64} - 1 = 18,446,744,073,709,600,000$ or about 8.55%.

For the 4-player Platypus game this is 157,788,000,000,000,000,000 moves out of a total of $(500,000 \times 500,001 \times 500,002 \times 500,003)/24 = 2,604,197,916,781,250,000,000$ or about 6.05%.

(d) How does your answer change if the moves for the Tower of Hanoi now take 0.01 seconds and those for the Platypus game now take 0.01 seconds? Explain your answer. (2 marks)

Answer: The Tower of Hanoi now makes $15,778,800,000,000,000,000$ moves out of a total of $18,446,744,073,709,600,000$ or about 85.5%.

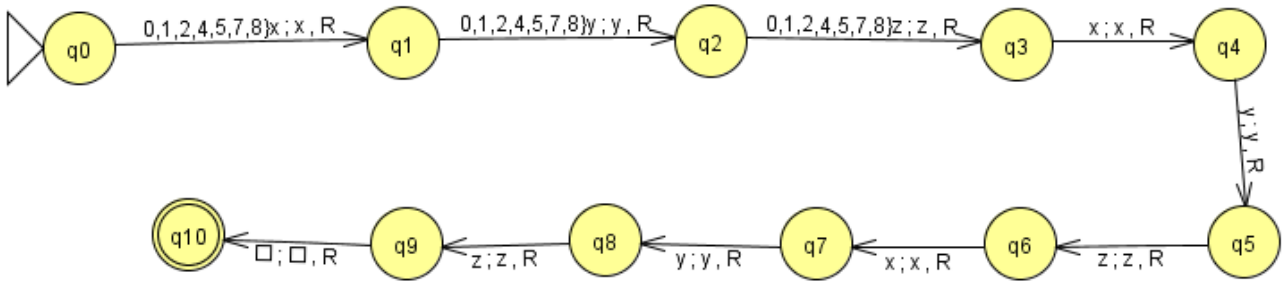
For the 4-player Platypus game this is now also $15,778,800,000,000,000,000$ moves out of a total of $(500,000 \times 500,001 \times 500,002 \times 500,003)/24 = 2,604,197,916,781,250,000,000$ or about 0.61%.

(4+4+4+2 = 14 marks)

4. "Everything in triplicate" (or *omnes in triplicata* in Latin :-)) is the motto of the University of Warthogs, which means that student numbers are of the form www where $|w| = 3$ and $w \in \{0,1,2,4,5,7,8\}^*$ (the digits 3, 6 and 9 are not used in such numbers, as these are considered sacred digits).)

(a) Construct a Turing machine to recognise such numbers. Note that any string of length 8 or less should be rejected, as should any string of length 10 or more. In addition, any string of length 9 containing a 3, 6 or 9 should be rejected. (3 marks)

Answer: Note that the solution below won't strictly work in JFLAP (although the documentation says it should :-(). But this will be fine for this exercise.



(b) Which of the following machines could also be used to recognise such numbers? Briefly justify each of your answers. (2 marks)

- A non-deterministic PDA
- A deterministic PDA
- A non-deterministic finite state automaton (NFA)
- A deterministic finite state automaton (DFA)
- A linear-bounded automaton

Answer: This language is finite, so an NFA or DFA could be used to recognise it. Therefore, any of the above could be used to recognise it.

(c) The Chancellor of the University of Warthogs, Pumbaa the Magnificent, soon realises that limiting the number of students in his university to a few hundred is not a good idea. So he then decrees that student numbers can now be of the form www or ww^Rw or www^R or ww^Rw^R where $|w| = 3$ and $w \in \{0, 1, 2, 4, 5, 7, 8\}^*$

Which of the following machines can be used to recognise the new student numbers? Briefly justify each of your answers. You do not have to explicitly construct any machines in your answer. (3 marks)

- A deterministic Turing machine
- A non-deterministic PDA
- A non-deterministic finite state automaton (NFA)
- A deterministic finite state automaton (DFA)

Answer: This language is also finite, although a little more complex than before. But as above, it can be recognised by an NFA or DFA, and so any of the above could be used to recognise it.

(d) Having finally received some better advice, Pumbaa further decrees that from now on, all student numbers will be of the form ww^Rw where $|w| \geq 3$ and $w \in \{0, 1, 2, 4, 5, 7, 8\}^*$.

Given these changes, which of the following machines can be used to recognise this latest definition of valid student numbers? Briefly justify each of your answers. You do not have to explicitly construct any machines in your answer. (3 marks)

- A deterministic Turing machine
- A non-deterministic PDA
- A non-deterministic finite state automaton (NFA)
- A deterministic finite state automaton (DFA)

Answer: As the length of the strings is now unbounded, we can only use a deterministic Turing machine out of these classes of automata. This language is not context-free, so all the others cannot recognise it.

(e) After a particularly bad nightmare, Pumbaa (who changes his mind like the wind) now insists that in addition to the conditions in the previous question, all numbers must have the property that the w in ww^Rw cannot repeat any digits. In other words, the only valid student numbers are ww^Rw where $|w| \geq 3$ and $w \in \{0, 1, 2, 4, 5, 7, 8\}^*$ and there are no repeated digits in w. For example, 124224211242 is no longer a valid student number as the string 1242 has a repeated 2. Does this change your answer to the previous question? If so, why and how? If not, why not? (3 marks)

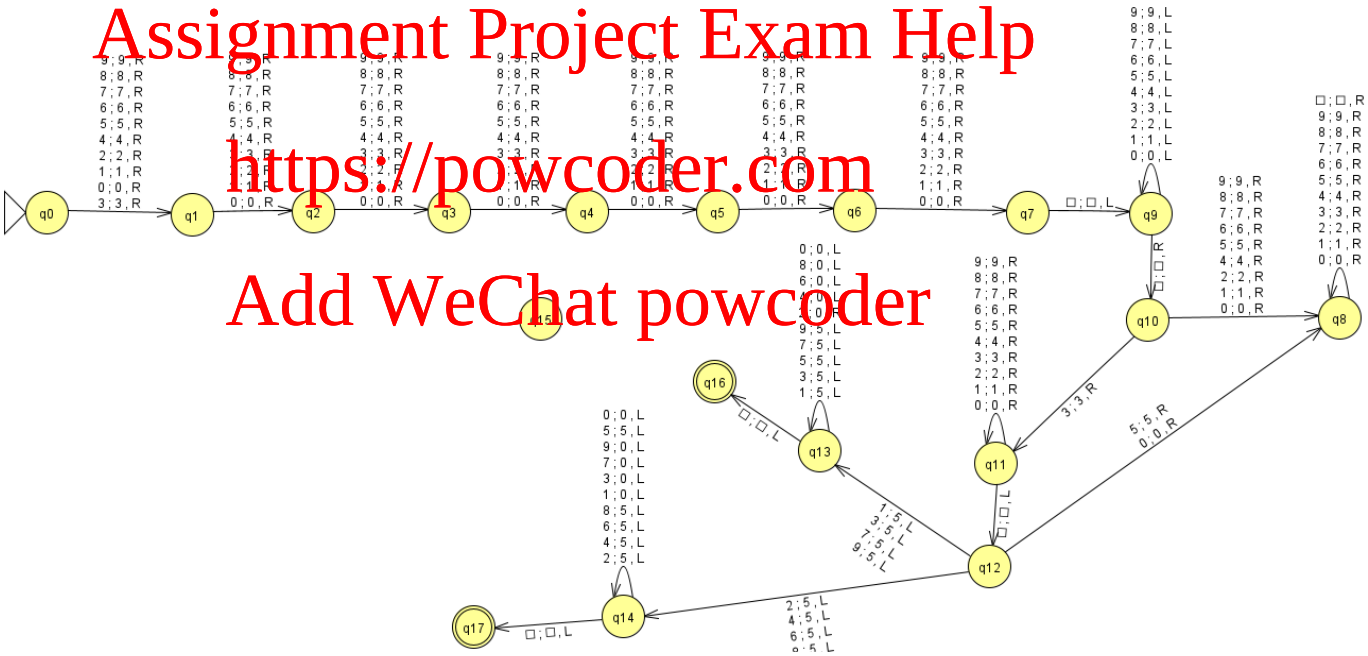
Answer: The key thing to recognise here is that w can have not more than 7 digits without repeating any. So again this is a finite language, and so an NFA or DFA could be used to recognise it. Therefore, any of the above could be used to recognise it.

(3+2+3+3+3 = 14 marks)

5. (a) Construct a Turing machine M_1 which given as input a string of length 7 over $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ performs as follows. (3 marks)

- The machine only halts in an accepting state if the input string has exactly 7 digits and commences with 3. Strings of any other length are rejected. Strings of length 7 that do not commence with 3 cause the machine to loop forever.
- If the last digit is 1, 3, 7 or 9, the machine replaces each of these digits with 5, and each of the digits 2, 4, 6 and 8 with 0.
- If the last digit is 2, 4, 6 or 8, the machine replaces each of these digits with 5, and each of the digits 1, 3, 7 and 9 with 0.
- If the last digit is 0 or 5, the machine loops forever.

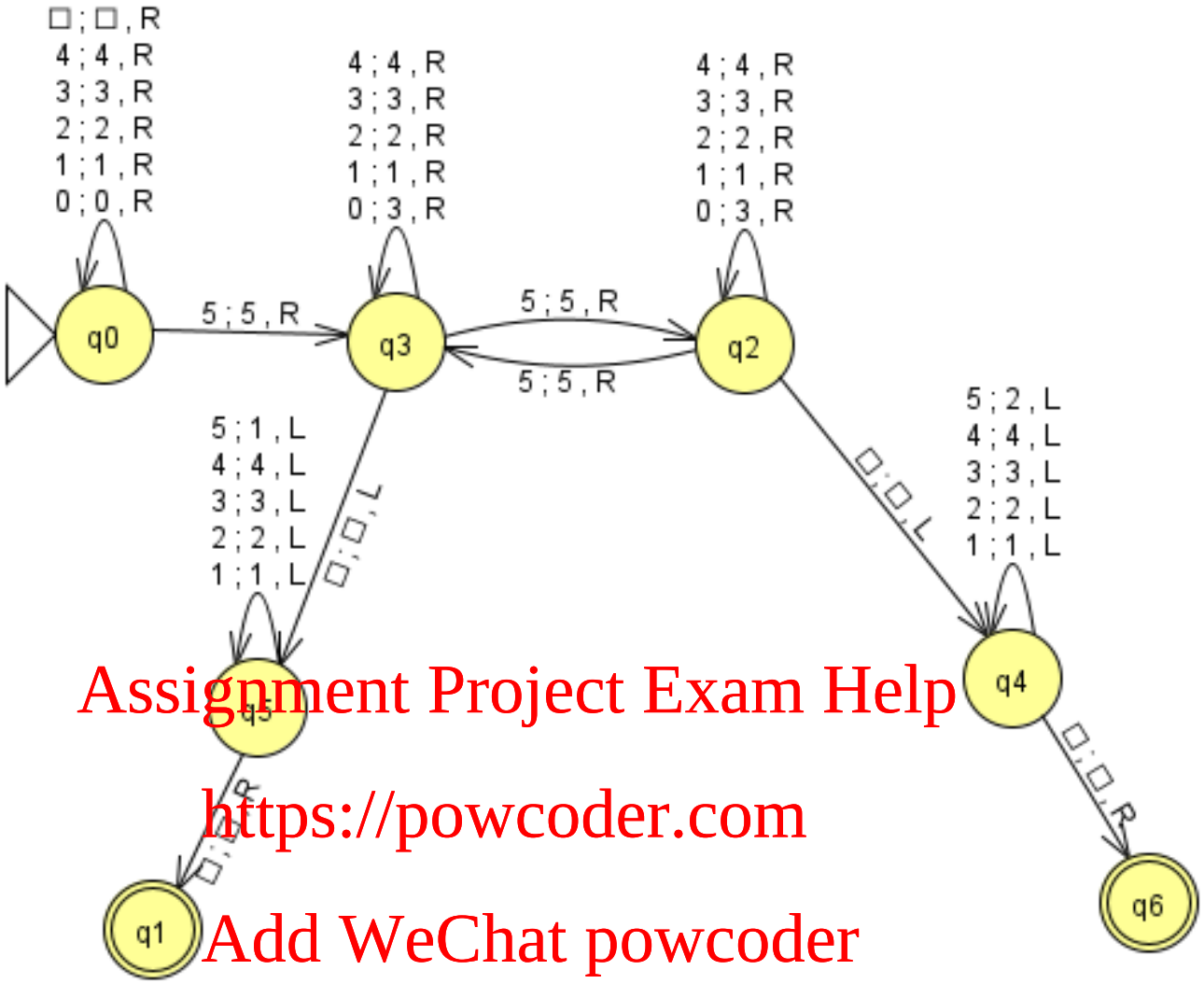
Answer: One such machine is below.



(b) Construct a Turing machine M_2 which given as input a string over $\{0, 1, 2, 3, 4, 5\}$ performs as follows. (3 marks)

- If there are no 5's in the input string, the machine loops forever.
- Otherwise
  - if there are an even number of 5's in the input string, the machine replaces each 5 with a 2.
  - If there are an odd number of 5's in the string, the machine replaces each 5 with a 1.
- All 0's in the string are replaced with a 3.

Answer: One such machine is below.



(c) Consider the machine formed by combining M_1 and M_2 as above in sequence into a new machine M_3, so that M_3 first performs as M_1 does, rewinds the tape head to the appropriate point on the tape and then performs as M_2 does. Clearly the behaviour of M_3 will depend on the behaviour of M_1 and M_2. (4 marks)

- For which inputs does M_3 halt with rejection?
- For which inputs does M_3 halt with acceptance?
- For which inputs does M_3 not halt?

In each case, justify your answer.

Answer: M_1 rejects strings of length 6 or less, or of length 8 or more. So M_3 will also reject these strings. M_1 loops forever on strings of length 7 which either do not begin with 3, or begin with 3 and end with either 0 or 5. So the only inputs which M_3 could possibly accept are those of length 7 which begin with 3 and end with 1,2,3,4,6,7,8 or 9. Now as M_1 will convert any string it accepts into a string of 0's and 5's with at least 1 5 in it (due to the 'last digit' alternatives) M_2 will not loop forever on any string which is generated by M_1. As M_2 otherwise accepts all strings and does not reject any, this means we have the following.

- M_3 halts with rejection on all strings of length 6 or less or of length 8 or more.
- M_3 halts with acceptance on all strings of length 7 which commence with 3 and end with either 1,2,3,4,6,7,8 or 9.
- M_3 loops forever on all strings of length 7 which either do not start with 3 or start with 3 and end with either 0 or 5.

(d) Consider now the machine M_0, which is the same as M_1 except that rather than deterministically replacing 1,3,7, and 9 with 5, M_0 *nondeterministically* replaces each of these digits with one of 5 or 2, and rather than deterministically replacing 2,4,6, and 8 with 5, M_0 *nondeterministically* replaces each of these digits with one of 5 or 3. For example, if there is a transition such as $\delta(q_i, 3) = (q_j, 5, R)$ in M_1, there are 2 corresponding transitions in M_0, i.e. $\delta(q_i, 3) = \{(q_j, 5, R), (q_j, 2, R)\}$ in M_0. (4 marks)

The same combination technique as above is used to combine M_0 and M_2 into M_4.

- For which inputs does M_4 halt with rejection?
- For which inputs does M_4 halt with acceptance?
- For which inputs does M_4 not halt?

In each case, justify your answer.

Answer: As there is still the execution path in M_0 in which each of the digits 1,2,3,4,6,7,8, and 9 get replaced with 5, the strings accepted by M_3 are still accepted by M_4. Therefore M_4 will have the same behaviour as M_3.

(3+3+4+4 = 14 marks)

6. (a) Prove that the language $L_1 = \{xx \mid x \in \{0,1,2\}^*\}$ is not regular by using the Pumping Lemma. Use the string 1^n2^n at an appropriate point in the proof. (5 marks)

Answer: Note: to avoid confusion, it may be simpler to write the language as $L_1 = \{xx \mid x \in \{0,1,2\}^*\}$ or something like that.

Proof: Assume L_1 is regular.

Then $\exists n \geq 1$ such that for all $w \in L_1$ and $|w| \geq n$ and $\exists x, y, z$ such that $w = xyz, |xy| \leq n$, $y \neq \lambda$ and $xy^iz \in L_1$ for all $i \geq 0$.

Consider the string $w = 1^n2^n1^n2^n$. So $w \in L_1$, $|w| \geq n$ and $|xy| \leq n$, and so we have $y = 1^j$ for some $1 \leq j \leq n$.

Now consider $i = 2$, and so xy^iz is $1^{n+j}2^n1^n2^n$.

As the Pumping Lemma requires $xy^iz \in L_1$, this is a contradiction, and so we have shown that our assumption is false, i.e. that L_1 is not regular.

(b) Write out the proof of the same result which uses the string 2^n10 and $i = 3$ instead. Which steps are different? Which steps remain the same? (4 marks)

Answer:

The steps that are different are in box $\boxed{\text{like this}}$.

Proof: Assume L_1 is regular.

Then $\exists n \geq 1$ such that for all $w \in L_1$ and $|w| \geq n$ and $\exists x, y, z$ such that $w = xyz, |xy| \leq n$, $y \neq \lambda$ and $xy^iz \in L_1$ for all $i \geq 0$.

Consider the string $\boxed{w = 2^n102^n10}$. So $w \in L_1$, $|w| \geq n$ and $|xy| \leq n$, and so we have $\boxed{y = 2^j}$ for some $1 \leq j \leq n$.

Now consider $\boxed{i = 3}$, and so xy^iz is $\boxed{2^{n+j}102^n10}$.

As the Pumping Lemma requires $xy^iz \in L_1$, this is a contradiction, and so we have shown that our assumption is false, i.e. that L_1 is not regular.

(c) There are some errors in the proof below that the language $L_2 = \{1^i2^j3^i4^i \mid i, j \geq 0\}$ is not context-free. Find and correct all such errors. The best way to do this is to write out the correct proof, highlighting the differences between the correct proof and the one below. (4 marks)

Claim: The language $L_2 = \{1^i\ 2^j\ 3^i\ 4^i \mid i, j \geq 0\}$ is not context-free.

Proof: Assume L_2 is regular. Then the Pumping Lemma applies and so for all $n \geq 1$ such that for some $w \in L$ such that $|w| \geq n$, $w = xyzuv$ where

- $|yzu| \leq n$
- $y \neq \lambda$ and $u \neq \lambda$
- $xy^izu^iv \in L$ for all $i > 0$

Choose $w = 1^n2^n3^n4^n$ and so $w \in L$ and $|w| \geq n$. So by the Pumping Lemma $w = xyzuv = 1^n2^n3^n4^n$ and $|yzu| < n$. This means that y and u can contain at most two different digits.

Now if y or u contains 12, or 23 or 34, then $xy^2zu^2v \in L$, as there would be some digits out of order.

So both y and u contain at most one type of digit. But this also means that $xy^2zu^2v \notin L$, as then there will not be equal numbers of 1's, 3's and 4's in $xyzu^3v$.

This is a contradiction, and so L_2 is not context-free.

Answer:

Proof: Assume L_2 is regular *(should be context free)* Then the Pumping Lemma applies and so for all *(should be 'there is an')* $n \geq 1$ such that for some *(should be 'for all')* $w \in L$ such that $|w| \geq n$, $w = xyzuv$ where

- $|yzu| \leq n$
- $y \neq \lambda$ and $u \neq \lambda$ *(should be $y \neq \lambda$ or $u \neq \lambda$)*
- $xy^izu^iv \in L$ for all $i > 0$ *(should be $i \geq 0$)*

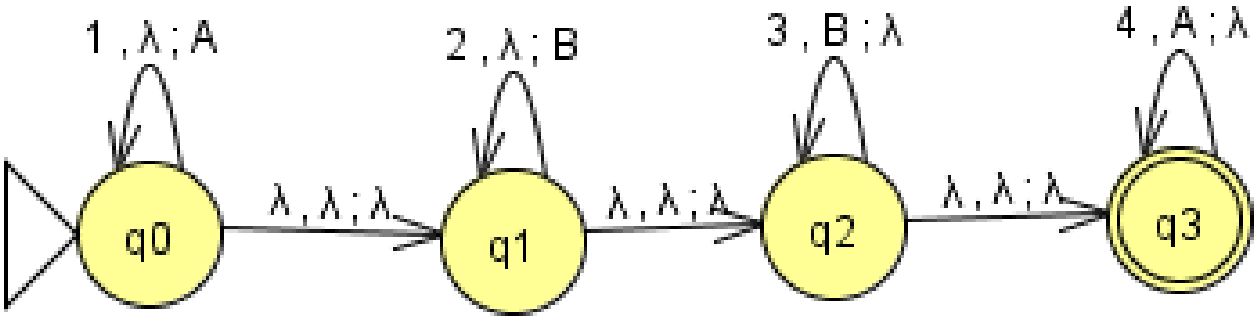
Choose $w = 1^n2^n3^n4^n$ and so $w \in L$ and $|w| \geq n$. So by the Pumping Lemma $w = xyzuv = 1^n2^n3^n4^n$ and $|yzu| < n$. *(should be $|yzu| \leq n$)* This means that y and u can contain at most two different digits.

Now if y or u contains 12, or 23 or 34, then $xy^2zu^2v \in L$, *(should be $xy^2zu^2v \notin L$)* as there would be some digits out of order.

So both y and u contain at most one type of digit. But this also means that $xy^2zu^2v \notin L$, as then there will not be equal numbers of 1's, 3's and 4's in $xyzu^3v$. *(should be xy^2zu^2v)*

This is a contradiction, and so L_2 is not context-free.

(d) Give a PDA for the language $L_3 = \{1^i2^j3^j4^i \mid i, j \geq 0\}$. Is your machine deterministic or non-deterministic? Briefly explain your answer. (2 marks)



(e) Is there a PDA for the language $L_4 = \{w \mid w \in \{1,2,3,4\}^*, n_1(w) = n_4(w), n_2(w) = n_3(w)\}$? Note that this is similar to L_3 but without any constraint on the order of the digits. Explain your answer. Note that there is no need to either construct a PDA or give a proof that there is none to answer this question; a few sentences of argument will suffice. (3 marks)

Answer: No. This language is not context-free. In order to recognise this language with a PDA, it is necessary to keep track of the number of 1's and the number of 4's simultaneously, in order to ensure the number of 1's and 4's is correct. But this cannot be done with one stack.

(f) Give a context-free grammar for $L_5 = \{2\#1^i2^j03^j4^i\#3 \mid i, j \geq 0\}$ with at most 5 rules. The number of rules is the number of different right hand sides in the grammar; for example, the grammar below has 6 rules. (2 marks)

$S \to aA|Aa|a$
$A \to aBaa|Baa$
$B \to abc$

Answer:

$S \to 2\#A\#3$
$A \to 1A4 \mid B$
$B \to 2B3 \mid 0$

(5+4+4+2+3+2 = 20 marks)

7. Gregor the Grammar Goblin hears about your work in Computing Theory, and is of course very impressed. He particularly loves the idea of grammars generating languages, and that in principle, everyone could specify their own grammar, and hence their own language. He is also an entrepreneur, and during a conversation with you about the Chomksy hierarchy, he has an idea for a new business. This is for customers to send him an unrestricted grammar, from which he will then generate all words in the language (up to a limit of 1,000), which he will then arrange via a new artistic algorithm of his in a visually appealing manner. He will then charge the customer a very large sum of money for the artistic output. However, Gregor is not a fan of swear words, and so he wants to incorporate in the process a means to check the words generated by the grammar

against his list of offensive words. If any of the words on Gregor's list are in the language specified by the customer's grammar, Gregor will refuse to generate the artistic output, and instead fine the customer for offensive language. Naturally Gregor looks to you to develop this process, and insists that this process must work for any possible grammar that the customer may have.

(a) Explain why Gregor's idea will not work, no matter how much computing power he may use. (4 marks)

Answer:

Gregor is asking for a solution to an undecidable problem, i.e. whether a specific set of strings is in the language of an unrestricted grammar. As unrestricted grammars are equivalent in power to Turing machines, this is equivalent to asking whether a given set of strings is accepted by a Turing machine. This is an undecidable problem, and hence there is no way to provide an algorithm which will work for an given unrestricted grammar.

(b) Gregor also hears on his social media channels about universal Turing machines, thinks they are very cool, and suggests to you that a solution to his original problem may be to encode the customer's unrestricted grammar as an input to a universal Turing machine. Explain why this approach will be no help. (2 marks)

Answer:

Universal Turing machines are indeed cool, but they reflect computational properties; they cannot change them. So using a universal Turing machine will not change the fact that the problem is undecidable. In other words, a problem that cannot be solved at all cannot be solved on a universal Turing machine.

(c) Suggest a way in which Gregor can accomplish something like his original intention by an appropriate relaxation of his policy. (3 marks)

Answer:

Gregor can accomplish something like his original intention by relaxing the requirement that the check on the language of the grammar must always terminate with the correct answer. This could be done by limiting the time allowed for such a check, and being prepared to answer say 'unknown' as well as 'yes; or 'no'.

(d) Gregor is rather stubborn, though, and would like to be able to guarantee that his process of checking for swear words will work. What classes of grammar from the Chomsky hierarchy, if any, could be used for this purpose? Explain your answer. (3 marks)

Answer:

Any of the other three classes of grammar in the Chomsky hierarchy, i.e. regular, context-free and context-sensitive languages, would be suitable for this purpose, as the problem of determining whether or not a given string is in the language of the grammar is decidable.

(4+2+3+3 = 12 marks)

8. Consider the definitions below.

- A *universal Turing machine* U is a Turing machine that takes the encoding of a Turing machine M and an input string w and emulates the computation of M on w. More specifically, U takes as input *code(M)code(w)*, (where *code()* is a function which encodes Turing machines and inputs into the input language of U) and for every configuration in the computation of M on w, there is a corresponding configuration in the computation of U on *code(M)code(w)*.

- A universal Turing machine U is **steadfast** if for every M and w, whenever M terminates on w, U terminates on *code(M)code(w)*.
- A universal Turing machine U is **dedicated** if for every M and w, whenever M does not terminate on w, U does not terminate on *code(M)code(w)*.
- A universal Turing machine is **constant** if U is steadfast **and** dedicated.

(a) Is it possible to have a universal Turing machine which is steadfast but not dedicated? Is it possible to have a universal Turing machine which is dedicated but not steadfast? Explain your answers. (4 marks)

Answer:

It is possible to have a universal Turing machine which is steadfast but not dedicated. To be steadfast, the universal machine U must terminate on $code(M)code(w)$ when M terminates on w, but to be not dedicated, there must be some M and w for which M does not terminate on w but U terminates on $code(M)code(w)$. To do this, U can directly emulate M on w, but may involve a simple loop check. Then there will be an M and w for which M does not terminate on w but U terminates on $code(M)code(w)$.

It is possible to have a universal Turing machine which is dedicated but not steadfast. For example, Wolfram's universal machine imitates a terminating computation by infinitely repeating the same configuration. In other words, this is a universal Turing machine which never terminates, and hence is dedicated but not steadfast.

(b) Should it be a requirement for a universal Turing machine to be constant? Or is it reasonable for a universal Turing machine not to be constant? Explain your answer, using appropriate examples. (4 marks)

Answer:

There is no set answer here. There is an intuitive sense in which being constant means that the universal Turing machine is utterly dependent on the input machine for termination. If the intention behind the universal machine is to reproduce the input machine's behaviour in a perfectly faithful manner, then being constant seems reasonable. This is what Colmerauer's machine does, for example. On the other hand, this precludes even the simplest forms of loop checking, as this will involve the universal machine terminating when the original one does not. In addition, some like Wolfram argue that even if the original machine terminates, the universal one should not, in order to provide a consistent pattern of behaviour. The famous 2,3 machine from the competition, for example, does not terminate on any input.

(4+4 = 8 marks)