UNCA CSCI 431
Exam 2 Fall 2019
3 December 2019 – 3:15 pm to 4:55 pm

You may use your notes, printouts, scratch paper, and your textbook. You may not use any calculators, electronic devices, or help from any other source or person.

Anyone needing a break during the exam must leave their exam with the instructor.

This exam must be turned in before 4:55 PM.

Name:_____

There are five equally-weighted questions.

Problems 2 and 5 spread over two pages to allow more room for artwork.

Extra space for long-winded answers of exam problems.

During the exam, it was decided that the exam grade would be computing by dropping the question with the lowest score. The remaining four will be equally weighted.

Problem 1: Regular Expressions, Grammars, and Derivations

Problem 1A: Regular expressions

Write a regular expression that represents **all** strings of 0 and 1 with an even number of 1's. You can use either the textbook or **grep** syntax in your answer.

Note that 001100 and 11 and ε are **in**, but 000100 and 1 are **out**.

`(0*10*10*)*` ***or*** **`0*(10*1)*0*`** ***or*** **`(0*10*1)*0*`**

Problem 1B: Context free grammar

Write a context free grammar that represents **all** strings of 0 and 1 with an odd number of 1's.

Note that 001100 and 11 and ε are **in**, but 000100 and 1 are **out**.

This answer was submitted by Cole Peterson.

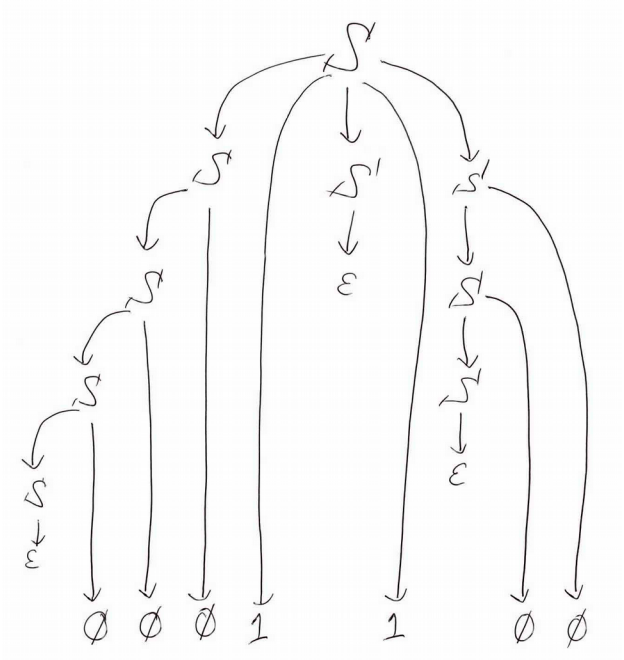
`S→ S1S1S | S0 | ε`

Problem 1C: String derivation

Illustrate the derivation of the string 0001100 using the grammar you created in Problem 1B.

This answer is dependent on your answer for Problem 1B.

`S→S1S1S→S01ε1S0→S0011S00→S00011ε00→ε0001100→0001100`



Problem 2: CFG transformations

Use the following CFG in both subproblems

- The alphabet for the language is **Σ** = {**0**, **1**}.
- The start variable for the language is **T**.
- Here are the rules:
  - **T → 0M0**
  - **M → 1T | T1 | ε**

Problem 2A: CFG → CNF
Convert the Context Free Grammar to Chomsky Normal Form.

You should stick with the procedure described in Example 2.10 of the textbook.

Add new start state S_0

S_0 → T
T → 0M0
M → 1T | T1 | ε

Remove ε productions

S_0 → T
T → 0M0 | 00
M → 1T | T1

Remove unit rules (S_0 → T)

S_0 → 0M0 | 00
T → 0M0 | 00
M → 1T | T1

Add variables for 0 and 1

S_0 → XMX | XX
T → XMX | XX
M → YT | TY
X → 0
Y → 1

Fix targets with more than two variables

S_0 → XP | XX
T → XP | XX
M → YT | TY
P → MX
X → 0
Y → 1

Continuing with Problem 2

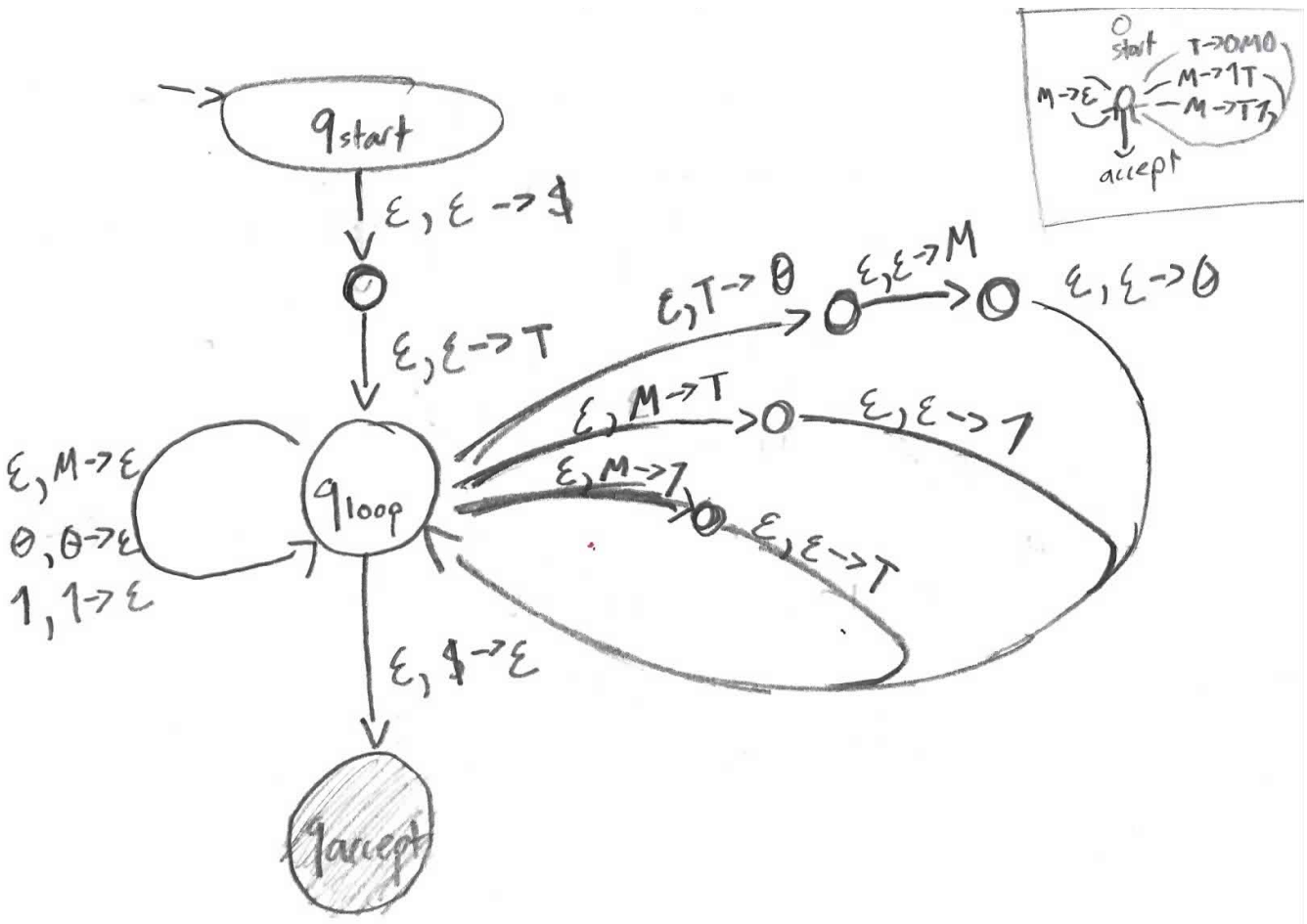
Use the following CFG in both subproblems

- The alphabet for the language is **Σ** = {**0**, **1**}.
- The start variable for the language is **T**.
- Here are the rules:
  - **T → 0M0**
  - **M → 1T | T1 | ε**

Problem 2B: CFG → PDA
Generate a PDA the Context Free Grammar shown above.

You should stick with the **flower algorithm** presented in Theorem 2.20 of the textbook.

This answer submitted by Cristopher Borgstede



Pumping Lemma (Theorem 1.70)

If A is a regular language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

- for each $i \geq 0$, $xy^iz \in A$
- $|y| > 0$
- $|xy| \leq p$

Problem 3: Disproving regularity

Show that the language $\{0^i10^i \mid i \geq 0\}$ is **not** regular.

Note that 1 and 001 and 00100 are **in**, but 0000, 00011000 and 101 are **out**.

Let p be the pumping length and let s be 0^p10^p which is longer than p.

By the pumping lemma, there exists x, y, and z such that $0^p10^p = xyz$

- **for each $i \geq 0$, $xy^iz \in A$**
- **$|y| > 0$**
- **$|xy| \leq p$**

From $|y| > 0$ and $|xy| \leq p$ and $0^p10^p = xyz$, we know that

- **$xy = 0^q$ where $q \leq p$**
- **$y = 0^r$ where $r > 0$**
- **Because y can be deflated, $0^{p-r}10^p$ must be in A**
- ***However*** **$0^{p-r}10^p$ cannot be in A because p-r < p**

By contraction, $\{0^i10^i \mid i \geq 0\}$ cannot be regular.

This is a bit more formal than I would usually write it.

Problem 4: Turing Machines

In Problem 4 of the last exam, you (hopefully) proved that the following language is **not** context free:

- $\{a^i b^j c^k \mid i \geq j \text{ and } i \geq k\}$

In this exam, show that this language is decidable by "designing" a Turing machine that decides language. You may assume that the input alphabet for the language are the symbols a, b and c.

In your design, do not write a super-formal description that satisfies the formalities of Definition 3.3 as shown in Figure 3.10. Instead give a concise "informal" description similar to that shown in Example 3.11 (p 174). You could also use some drawings of Turing Machine configurations to illustrate your solution.

If you run out of room here, continue on the bottom of the page 7.

There are many ways to do this. Here is one.
By the way, I improved my answer after reading some of your solutions.

Stage One: Scan from the beginning to end making sure that, in the input string, a's precede b's and b's precede c's in the input and there are no other symbols. This does not involve changing the input tape because this activity is testing if the input matches the regular expression a*b*c*. Reject the input string, if the regular expression test fails!

Stage Two: Now see if there are at least as many a's as b's and as many a's as c's. Do this by making several passes over the input. On each pass try to "cross off" one a, b, and c. Continue these passes until either:

- **There was no a to cross off, but there was at least one b or c to cross off. This means that either there are more b's than a's or more c's than a's. The Turing machine should REJECT.**
- **There was no b *and* no c to cross off. This means that there were at least as many a's as b's *and* as many a's as c's. The Turing machine should ACCEPT.**
- **Otherwise (one a and at least one b or c to cross off), continue.**

This process is guaranteed *not* to loop.

My original answer had separate passes for checking off b's and c's. Zach Boone, David Pulse and Cole Peterson "inspired" this one pass check-off solution.

Problem 5: Counting and reducing

Problem 5a: Enumerating the powers

Show that the set of all integral powers of integers, i.e., i^n for integers i and n, is countable. Consider 0^0 to be 1, just like C, Java, JavaScript and Python. Also, 3^{-5} would be in this set.

This is a variation of the correspondence argument for $\mathbb{N}$ and $\mathbb{Q}$, shown in Figure 4.16 of the textbook. You could just insert the diagram here and replace i/n with i^n,

Alternatively, you could just mention that the elements, i/n, of $\mathbb{Q}$ can by mapped into the powers, i^n, which is a subset of $\mathbb{Q}$. You should also mumble something about how you are handing the fact the 8^3 equals 4^6 but 8/3 does not equal 4/6. But it really doesn't matter since 9/3 equals 3/1. Either way there is some double counting.

Continuing with Problem 5

Problem 5B: Problem 5.24

Show that the language $5B_{TM}$ defined below is decidable.

- $5B_{TM}$ ={ <*M*, *n*, *w*> | *n* encodes a prime number
  or *M* is a Turing machine that accepts w}

By the way, the language A_{TM} described below is undecidable.

- A_{TM} = { <*M*, *w*> | *M* is a Turing machine that accepts *w* }

This is a really silly problem.

Suppose the $5B_{TM}$ was decidable. Let's build a new Turing machine *X* that does something like the following.

- ***X* receives an input <*M*, *w*> where *M* is a Turing machine encoding.**
- ***X* "calls" $5B_{TM}$ with <*M*, 4, *w*>**
- **Because 4 isn't prime, $5B_{TM}$ will accept only if M would accept w.**
- **Therefore *X* implements A_{TM} and *is* decidable.**
- **But, A_{TM} *is* undecidable!**
- **Contradiction achieved.**