Name: [redacted]

Student ID: [redacted]

CS181 Winter 2012 - Midterm
Wednesday, February 8, 2012

- You will have 110 minutes to take this exam.
- You are allowed to use any theorem shown in class or in the textbook, as long as you clearly cite it.
- Place your name and UID on every page of your solutions. **Please use separate pages for each question.**
- There are three questions and an extra credit question, bearing a total of 125 points and 40 extra credit points. You will receive significant partial credit for writing down clearly expressed correct intuition, even if your answer is incorrect.
- For each part (except for the extra credit), 20% of the points will be given if your answer is "I don't know". However, if instead of writing "I don't know" you write things that do not make any sense, no points will be given.
- **Skim the entire exam before you begin.** Avoid spending too much time on a single part, so you would be able reach the other parts.

Honor Code Agreement: I understand this exam is open-book and open-notes, but any materials not used in this course are strictly prohibited. I also understand that this exam is to be taken individually without any outside help (except possibly from the professor or the TA) within the time limits set forth. I agree to adhere to the course honor code and if I am unsure of any rules of the honor code, I will ask for clarification from the professor or the TA.

Signature: [redacted]

Question	Points	
1	38	40
2	29	60
3	0	25
EC	5	40
Total	72	125+40

Name: [redacted]

Student ID: [redacted]

1. **Super GNFAs.** Recall the definition of a GNFA (page 73 of the textbook). Define a *super GNFA* as a further generalization of NFA which has a context-free grammar describing each transition (instead of a regular expression).

   Let $G = (V, \Sigma, R, S)$ be any context-free grammar over Σ. Let $\mathcal{G}$ be the set of all context-free grammars over Σ. Define a *super generalized nondeterministic finite automaton* as a 5-tuple, $(Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$, where $Q, \Sigma, q_{\text{start}}, q_{\text{accept}}$ are the same as a GNFA and,

   - $\delta : (Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\}) \to \mathcal{G}$ is the transition function.

   For each transition a GNFA may take this transition upon consuming some input word generated by the grammar describing this transition. Formally,

   A word $w \in \Sigma^*$ is said to be accepted by a super GNFA if and only if there exists some sequence of states $(q_0, q_1, q_2, \ldots, q_t)$ and some decomposition of w into t strings $w_1 \ldots w_t$ where each $w_i \in \Sigma^*$ such that,

   - $q_0 = q_{\text{start}}, q_t = q_{\text{accept}}$
   - for all $1 \leq i \leq t$, $w_i \in L(\delta(q_{i-1}, q_i))$

   The language accepted by any super GNFA is the set of all words over Σ^* that are accepted by it. We only need a precise explanation (and not a formal proof) for both parts below.

   (a) (**10 pts.**) Recall that context-free grammars are closed under union and concatenation. Show that context-free grammars are also closed under the star operator.

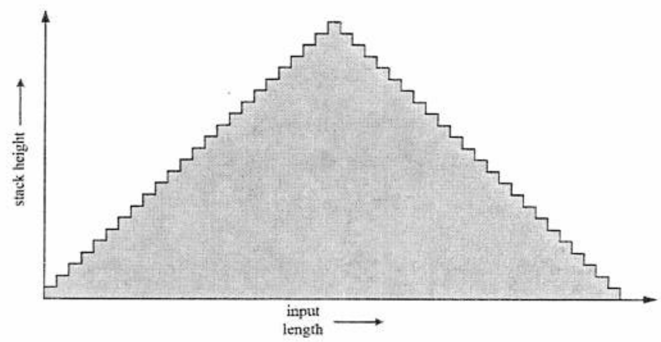
   (b) (**30 pts.**) Show that every super GNFA is equivalent to a super GNFA with only two states, $\{q_{\text{start}}, q_{\text{accept}}\}$. Thus super GNFA are equivalent to CFGs.

Name: [redacted]

Student ID: [redacted]

2. **Hungry PDA.** A *Hungry PDA* is a PDA in which:

- Every transition consumes one input character (note that this means that a hungry PDA cannot have a transition between states that ignores the input).
- Every transition always pushes or pops a symbol from the stack
- Symbols are popped from the stack only after all pushes have occurred. Thus, the stack height diagram always looks as shown below:



- The stack *must* be empty in order for the hungry PDA to accept.

An *appetizing language* is one that has a hungry PDA that accepts it.

(a) (**10 pts.**) Consider a hungry PDA M, which accepts the string "000111". Let $(q_0, q_1, q_0, q_2, q_3, q_4, q_3)$ be the sequence states of the machine when it accepted "000111". Also let the stack when the machine was at q_2 be "000". Given this information (and only this information) about M, show one other string that must be accepted by M. Provide an explanation.

(b) (**30 pts.**) Building on the intuition from the previous part, formally prove the following pumping lemma for appetizing languages:

If L is an appetizing language then there exists a pumping length $p > 0$ such that for all $s \in L$ with $|s| \geq p$, there exists a partitioning $s = uvxyz$ such that

- for each $i \geq 0$, $uv^ixy^iz \in L$,
- $|v| > 0$ and $|y| > 0$, $|v| = |y|$.

(*Hint: This problem has nothing to do with the pumping lemma for context-free languages.*)

(c) (**10 pts.**) Show how to modify your proof for part (b) so that in addition to the properties above, we also have that $|vxy| \leq p$.

(d) (**10 pts.**) Using the pumping lemma for appetizing languages above (parts (b) and (c)), formally prove that the following context-free language is not appetizing:

$$L = \{0^n 1^{3n} \mid n \geq 0\}$$

$p=3$

Name: [redacted]

Student ID: [redacted]

3. **Unique, unary, regular languages.** A language L is *unary* if it is defined over a singleton alphabet. Without loss of generality assume that $\Sigma = \{1\}$. A regular language L is *unique* if it is accepted by some DFA with exactly one accepting state. Prove *formally and rigorously* that:

   - **(25 pts.)** For every **infinite**, *unique, unary, regular* language L, $\exists a \geq 0$ and $b > 0$ such that $L = \{1^a 1^{bi} \mid i \geq 0\}$. $= \{1^* 1^* \} - \{1^0\}$

$L = \{11, 1111, 111111, \ldots\} = \{1^2 1^{2i} \mid i \geq 0\}$

Name: [redacted]

Student ID: [redacted]

4. **Extra Credit. This problem will be graded more harshly than the others.** Define a *baton passing context-free grammar* as a grammar $G = (V, \Sigma, R, S)$ in which the set of rules R may contain any context-free rules (like in CFGs), but in addition, each rule also specifies a variable which has to be expanded in the next step of the derivation.

   For example, let G be,

$$S \rightarrow A0, A$$
$$A \rightarrow A0, A$$
$$A \rightarrow 1, S$$

   The above is just a context free grammar augumented with an additional non-terminal in each rule. When a particular rule is applied in a derivation, the next step of the derivation *must* expand the non-terminal specified in this rule. For example, when the rule $S \rightarrow A0, A$ is applied in a derivation, the next step of the derivation must expand A. We use the notation (β, A) to represent the fact that β has been derived so far and A is the next non-terminal that must be expanded. Derivations begin at (S, S). For e.g., $(S, S) \Rightarrow (A0, A) \Rightarrow (A00, A) \Rightarrow (100, S)$, is a valid derivation that generates 100.

   Formally, the set of rules for a baton passing grammar is a subset of $V \times (\Sigma \cup V)^* \times V$, where each rule of the form (A, α, B) is to be interpretted as $A \rightarrow \alpha, B$. That is, A could be replaced by α and the next non-terminal that one must expand is B.

   Similar to CFGs, we formally say that a tuple (wAy, A) derives $(w\alpha y, B)$ denoted by $(wAy, A) \Rightarrow (w\alpha y, B)$ if $(A, \alpha, B) \in R$. We say that any $x \in \Sigma^*$ is in the language generated by G if and only if $(S, S) \Rightarrow^* (x, \cdot)$.

   (a) (**20 pts.**) Show that every context-free language can be generated by a baton passing grammar.

   (b) (**20 pts.**) Recall that $L = \{0^n 1^n 2^n \mid n \geq 0\}$ is not a context-free langauge. Show that there exists a baton passing grammar that generates L.

(Blank Page 1 — for your answers) Name: [redacted]

Student ID: [redacted]

① Super GNFA

Ⓐ Each CFG has a PDA that accepts L(CFG) as learned in class. ↙ Thm 2.12 p.106 ed 1

The star operator for a CFG is accomplished by creating a transition from all accepting states of the PDA to the start state on $\varepsilon, \varepsilon \to \varepsilon$.

Call this new PDA P'.

8/10 This PDA will accept any amount of w's concatenated where $w \in L$. So, the PDA accepts $w \in L(P')$. P' can then be made back into a CFG. Since the star operator applied on a CFG gave another CFG, CFG's are closed under the star operator.

— WHAT ABOUT THE EMPTY STRING? ε IS ALWAYS IN L*. YOU NEED A NEW START STATE THAT IS ALSO AN ACCEPT STATE.

Ⓑ We can collapse all intermediate transitions from q_{start} to q_{accept} into one CFG between them using the following technique:

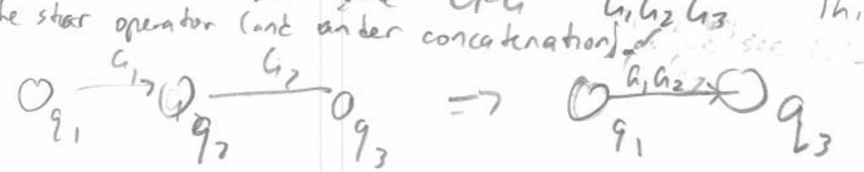
(union)

When a state can transition to multiple other states, we create a new state from its next states and create a new transition CFG to this new state. The new transition CFG is the union of the CFG's it replaced. This is valid because CFG's are closed under unions.

(star) CORRECT IDEA!

30/30

When a state q_2 has a loop as a transition, we replace the state and its loop with G_2^*, and create a transition from its neighboring states to each other on the CFG $G_1G_2^*G_3$. This is because CFG's are closed under the star operator (and under concatenation).



(concatenation)

We can remove intermediate states and replace the transitions between their neighboring states with a new CFG that is the concatenation of the old CFG's. This is because CFG's are closed under concatenation.

We repeat this process until q_{start} and q_{accept} are connected by one CFG made of the union, star, or intersection of the intermediate CFG's.

So, every GNFA is equivalent to one with just 2 states.

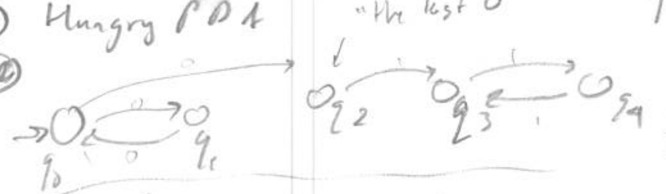
(Blank Page 2 — for your answers)

Name: [redacted]

Student ID: [redacted]

② Hungry PDA

ⓐ



This is a NFA.

How do the machine count?

insufficient explanation 3/5

correct answer 5/5.

It looks like the machine pushes 0's as it reads them (using a set of pushing states q_0 & q_1) and then pops off the stack as 1's are read (at the popping states q_3 & q_4). We know ∃ a transition (on non-empty input) from q_0 to q_2 and that q_3 is an accepting state. This machine then appears to accept on equal # of 0's & 1's, but definitely accepted an odd # of 0's & an odd # of 1's. Perhaps the machine "counted" an even # of 0's with q_0, q_1 and did similarly at q_3, q_4 for 1's. Then $L(\text{machine}) = \{0^{2k+1}1^{2k+1} \mid k \in \mathbb{N}\}$, using the given information.

So, another string is "0000011111".

PHP. 5/5

ⓑ Let our HPDA be $H = (Q, q_0, \delta, \Sigma, \Gamma, F)$.

Choose $p = |Q|$ and set s.t. $|s| \geq p$. Since $|s| > |Q|$, some states are repeated when s is passed to the PDA as input (by the PHP). If some states are repeated, there is a cycle of states. If this cycle is in the first half of processing s, then there was additional pushing that must be accounted for by having a cycle of popping in the second half. The converse is true, too. These statements are true because the stack must be empty by the time s terminates (pops and pushes must be equal in number). Since there are two portions of cycles in H, then 2 portions of s can be repeated; one in the first half and one in the second. These portions can be repeated an arbitrary amount because the cycle can be traversed an arbitrary amount of times (or not at all). So, $uv^i x y^i z \in L$ $\forall i \in \mathbb{N}$. In addition, the cycle's length must be nonzero because the word's length is greater than the amount of states. So, $|v| > 0$ and $|y| > 0$ and $|v| = |y|$ because both cycle lengths must be equal to ensure the stack is emptied when s terminates (and the stack must be emptied since $s \in L$).

Accept by empty stack 5/5.

implicit push before pop 3/5.

One input each transition 0/5.

no idea of symmetry 0/10.

(Blank Page 3 — for your answers)

Name: [redacted]

Student ID: [redacted]

(2) (c) v must be in the 1st half and y must be in the second, as stated previously. $|s| = |uvxyz| \geq p$. If $|vxy| > p$, then u and z must always be ε. But this is not always the case. So, $|vxy| \leq p$.

0/10.

incorrect proof.

explain.

(d) Choose $w \in L$, $w = 0^p 1^{3p}$ where $p \geq n$. Split $w = 0^{a_1} 0^{x_1} 0^{x_2} 0^{x_3} 1^{x_4} 1^{x_5} 1^{x_6} 1^{a_2}$. No decomposition maintains the pumping lemma.

Case 1: $u = 0^{a_1}$, $vxy = 0^{x_1} 0^{x_2} 0^{x_3}$, $z = 0^{x_4} 0^{x_5} 0^{x_6} 1^{a_2}$, $x_1 + x_2 + x_3 \leq p$. Pumping leads to $a_1 + i x_1 + x_2 + i x_3$ 0's $\neq$ (number of 1's)/3 $\forall i$.

Case 2: $u = 0^{a_1 x_1}$, $vxy = 0^{x_2} 0^{x_3} 1^{x_4}$, $z = 0^{x_5} 0^{x_6} 1^{a_2}$, $x_2 + x_3 + x_4 \leq p$. Pumping leads to $a_1 + x_1 + x_2 i + x_3$ 0's $\neq$ (number of 1's)/3 $\forall i$

Case 3: $u = 0^{a_1 x_1 x_2}$, $vxy = 0^{x_3} 1^{x_4} 1^{x_5}$, $z = 1^{x_6 a_2}$, $x_3 + x_4 + x_5 \leq p$. Pumping leads to $a_1 + x_1 + x_2 + x_3 i$ 0's $\neq$ (number of 1's)/3 $\forall i$

Case 4: $u = 0^{a_1 x_1 x_2 x_3}$, $vxy = 1^{x_4} 1^{x_5} 1^{x_6}$, $z = 1^{a_2}$, $x_4 + x_5 + x_6 \leq p$. Pumping leads to $x_4 i + x_5 + x_6 i + a_2$ 1's $\neq$ (number of 0's)·3 $\forall i$.

Cases 1-4 lead to ~~too~~ many 0's to uphold the 1:3 ratio. Case 4 leads to too many 1's.

So, L is not appetizing

Correct quantifier 5/5

partially correct Case analysis 3/5.

(Blank Page 4 — for your answers)

Name: [redacted]

Student ID: [redacted]

(3) Unary L.

Let L_0 be the original language (infinite, unique, unary, regular language in the problem statement).

We want to show $L_0 = L - \{1^a 1^{b} \mid i \geq 0\} = \{1^{a+bi}, a \geq 0, b \geq 0, i \geq 0\}$

The regular expression for L_0 is $R = (1^a)(1^b)^*$, which, by the problem statement, satisfies the original infinite, unary, regular language L_0.

X Suppose a and b do not exist. Then not only does R not exist, but now L can't be written as a regular expression, because any regular expression for L must use concatenation a-times and b-times to detect a string in L. Since no regular expression exists for L, then one doesn't exist for L_0. So L_0 can not be regular. This is a contradiction. So, a and b must exist and $L = \{1^a 1^{bi} \mid i \geq 0\}$ must exist, too.

COMPLETELY INCORRECT PROOF/APPROACH

0/25

(Blank Page 5 — for your answers) Name: [redacted]

Student ID: [redacted]

(4) Extra Credit

(a) Every CFG can be derived from Chomsky Normal Form by Thm 2.9 p.99 ed.1.

A Ch CFG can be made from a baton passing CFG by changing its rules to:

$A \to BC \implies$ ($A \to BC, B$; $A \to BC, C$) and ($A \to a \implies A \to a, S$) where S is the start symbol.

We also add $S \to \varepsilon S, S$ and ($S \to \varepsilon, S$) so $(S, S) \Rightarrow^*$ just S.

So, since every CFG can be generated by a Ch CFG, and every Chomsky CFG can be generated by a b.p. CFG, every CFG can be generated by a b.p. CFG.

half idea. $\frac{5}{20}$

(b) There must exist a baton passing CFG for L since derivations begin at (S, S); the left hand S can generate $0^n 1^n$ and the righthand S can generate 2^n. X