COMPSCI 3331 - Final review questions

Fall 2022

These questions are provided for review purposes and are not guaranteed to be the same or similar to the questions on the final. Some questions may be easier than the questions on the final and some may be harder.

Solutions to as many questions as possible will be written as I am able. There is no guarantee that any solutions will be available for any particular question.

1. Let G be the CFG defined by the following set of productions.

$$\begin{array}{rcl} S & \rightarrow & bbAaA \mid SSaa \mid aa \mid ABC \\ A & \rightarrow & Ab \mid Ac \mid CC \\ B & \rightarrow & BA \mid bb \mid Dd \\ C & \rightarrow & DA \mid \varepsilon \\ D & \rightarrow & a \end{array}$$

Give an equivalent grammar to G that has no ε-productions.

2. Let G be the CFG defined by the following set of productions.

$$\begin{array}{rcl} S & \rightarrow & SbaB \mid aa \mid ABC \\ A & \rightarrow & Ba \mid DaDd \\ B & \rightarrow & BA \mid ca \mid Dd \\ C & \rightarrow & DA \mid \varepsilon \\ D & \rightarrow & a \end{array}$$

Convert the grammar to CNF.

3. Let $G = (V, \Sigma, P, S)$ be a CFG in CNF. Give an $O(n^3)$ algorithm for the following problem:

- Input: A word w and a nonterminal $A \in V$.
- Output: the value $n_A = \max\{|u| \ : \ u \text{ is a suffix of } w \text{ and } A \Rightarrow^* u\}$.

That is, n_A is the length of the longest suffix of w that is generated by A in the grammar.

Solution: Let n be the length of w.

```
Construct the CYK table for the word $w$.
for $i = 1; i \leq n; i++$ do
    if $A \in T[i,n]$ then
        return i
    end if
end for
return -1   ▷ No suffix of $w$ can be generated
by $A$
```

The idea is that $T[i,n]$ gives the nonterminals that generate a suffix of the word. By looping through in this order, we find the longest suffix of w that can be generated by A.

4. Construct PDAs for the following languages:

(a) $L = \{a^n b^m xy \ : \ x,y \in \{0,1,2\}, x \equiv n (\text{mod} 3) \text{ and } y \equiv m (\text{mod} 3)\}$.

(b) $L = \{w\#x \ : \ w,x \in \{a,b\}^*, |w|_a = |x|_a, |w|_b \equiv |x|_b (\text{mod} 3)\}$.

Be sure to indicate what the starting stack symbol is for your PDA and how your PDA accepts words.

Solution ideas covered in review session.

5. Let C be a fixed integer. Extend the language from Assignment 3 as follows:

$$L_C = \{x\#1^n \ : n \geq 0, x \in \{a,b\}^* \text{ and } \ n - C \leq |x|_a \leq n + C\}$$

Give a context-free grammar for L_C. The productions in your grammar will depend on the value of C. Describe them using a uniform notation (e.g., by using consistently named variables or consistently defined productions, for instance).

The grammar $G_C = (V, \Sigma, P_C, S)$ generates L_C:

- $V = \{S, T, A\}$.
- $\Sigma = \{a, b, \#, 1\}$.
- P_C is the following set of productions:

$$\begin{aligned} S &\to aS1 \\ S &\to bS \\ S &\to T1^i \quad \forall 1 \leq i \leq C \\ S &\to \# \end{aligned}$$

$$\begin{array}{rcll} S & \to & A^iT & \forall 1 \le i \le C \\ A & \to & bA & \\ A & \to & Ab & \\ A & \to & a & \\ T & \to & aT1 & \\ T & \to & bT & \\ T & \to & \# & \end{array}$$

6. Consider the following modified language from Assignment 3:

$$L = \{u\#v \ : \ u, v \in \{0,1\}^* \text{ and } \mathrm{bin}(v^R) = \mathrm{bin}(u) + 2\}$$

Give a PDA that accepts L.

7. A CFG G is in Griebach Normal Form (GNF) if every production has the form

$$A \to aB_1B_2 \cdots B_n$$

for some letter a and nonterminals $B_1, B_2, \cdots, B_n$ (where $n \ge 0$).

Any grammar (that does not derive ε) can be converted to GNF. Given this fact, show that for any CFG L that does not include ε, you can construct a PDA M that accepts L in the following additional conditions:

- The PDA accepts by empty stack.
- The PDA M does not have any ε-transitions. That is, there are no rules of the form $\delta(q, \varepsilon, \gamma) = \{(q', \beta), \ldots\}$ for any stack symbol γ.

Solution: The major idea is:

- Convert a grammar to GNF.
- Convert the GNF grammar to a PDA using the usual algorithm.
- This algorithm produces transitions of two forms:
  - $\varepsilon, A/aB_1B_2 \cdots B_n$
  - $a, a/\varepsilon$
- But now we see that every successful computation of such a PDA uses a rule of the first form, followed immediately by a rule of the second form.
- So they can be replaced by rules like this in the PDA.

  - $a, A/B_1B_2\cdots B_n$

- This is not an ε rule as defined in the question.

8. Prove that the following languages are not context-free:

   (a) $L = \{a^p \ : \ p \text{ is a prime number}\}$.

   (b) $L = \{a^n b^{n^3} \ : \ n \geq 0\}$.

   (c) $L = \{w \in \{a,b\}^* \ : \ |w|_b = 2^{|w|_a}\}$.

Solution

(a) Let n be the constant defined by the pumping lemma. As there are infinitely many primes, let p be a prime number greater than n. Then $x = a^p \in L$.

By the pumping lemma, we can write $x = uvwxy$ where $|vwx| \leq n$ and $|vx| > 0$. As all letters in x are a, let i, j, k, ℓ be the lengths of u, v, x and w, so that $u = a^i, v = a^j$, $w = a^k$, $x = a^\ell$ and $y = a^{p-i-j-k-\ell}$. Note that $j+k+\ell \leq n$ and $j+\ell > 0$.

Consider now $uv^{p+1}wx^{p+1}y = a^{p+jp+\ell p}$. But as $p+jp+\ell p = p(\ell+j+1)$, and $\ell+j+1 \geq 2$, the length of $uv^{p+1}wx^{p+1}y$ is not prime. Thus, $uv^{p+1}wx^{p+1}y$ is not in L and L is not context free.

You'll note that this solution is very similar to the version you saw in Assignment 2 for showing that the same language is not regular. This is because there is only one letter a in all words, so there isn't the added complexity of the cases in this example.

(b) Let n be the constant defined by the pumping lemma and let $z = a^n b^{n^3}$. Decompose z as $z = uvwxy$ where $|vwx| \leq n$ and $|vx| > 0$. There are four cases:

- either v or x have more than one letter, i.e., they cross the boundary between a's and b's. In this case, by considering uv^2wx^2y, we can form a word that has a b before an a and this word is not in L.

- both u and v consist entirely of a's. In this case, let $|vx| = i$ for some $i > 0$. Then $uv^2wx^y = a^{n+i}b^{n^3}$. But as $i \neq 0$, $(i+n)^3 \neq n^3$. So $uv^2wx^2y \notin L$.

- both u and v consist entirely of b's. In this case, again let $|vx| = i$ for some $i > 0$. Then $uv^2wx^y = a^n b^{n^3+i}$. But as $i \neq 0$, $(n)^3 \neq i+n^3$. So $uv^2wx^2y \notin L$.

- u consists of a's and v consists of b's. Then write $z = uvwxy$ where

$$\begin{aligned}
u &= a^i \\
v &= a^j \\
w &= a^{n-i-j}b^k \\
x &= b^\ell \\
y &= b^{n^3-k-\ell}
\end{aligned}$$

for some $i, j, k, \ell \geq 0$ with $j+k+\ell \leq n$ and $j+\ell > 0$. We now consider uv^2wx^y and show that this word is not in L. Consider

$$uv^2wx^2y = a^{n+j}b^{n^3+\ell}$$

To show that this word is not in L, we must show that $(n+j)^3 \neq n^3+\ell$. We have two subcases:

– If $j = 0$, then we must have $\ell \neq 0$, since $j+\ell > 0$. But in this case,

$$a^{n+j}b^{n^3+\ell} = a^n b^{n^3+\ell}$$

and certainly this word is not in L.

– If $j \neq 0$, then consider $n^3+\ell$. As $\ell \leq j+|w|+\ell \leq n$, we have

$$\begin{aligned} n^3+\ell \leq n^3+n \\ &< n^3+3n^2+3n+1 \quad (\text{as } 3n^2+2n+1>0) \\ &= (n+1)^3 \leq (n+j)^3 \end{aligned}$$

Thus, $n^3+\ell < (n+j)^3$, and thus the number of a's cubed is not equal to the number of b's and the word is not in L.

(c) Idea covered in review session.

9. For each of the languages in the previous question, give an informal description of a multi-tape TM that recognizes the language.

(a) $L = \{a^p \ : \ p \text{ is a prime number}\}$.

(b) $L = \{a^n b^{n^3} \ : \ n \geq 0\}$.

(c) $L = \{w \in \{a,b\}^* \ : \ |w|_b = 2^{|w|_a}\}$.

Solution

(a) A 3-tape TM can recognize the language:

describe the usage of tapes.

- Tape 1: input tape.
- Tape 2: stores the current divisor of the input that we are testing.
- Tape 3: current multiple of the current divisor.

starting.

To start, the TM writes two a's on Tape 2. Then it repeates the following steps:

- Copy the number of a's on Tape 2 to Tape 3. Keep copying this number until the number of a's on Tape 3 is greater than or equal to the number of a's on Tape 1.
- If the number of a's on Tape 3 is equal to the number of a's on Tape 1, then stop and say no.
- Otherwise, erase all the a's on Tape 3, and add one a on Tape 2.
- if the number of a's on Tape 2 is equal to the number of a's on Tape 1, then end and say yes.

- Otherwise, go back to the beginning of this list of steps.

(b) idea missing.

(c) Here is a four tape TM to recognize L.

- Tape 1: input tape
- Tape 2: count of a's
- Tape 3: count of b's
- Tape 4: scratch tape - current power of 2.

The algorithm starts by copying the number of a's and b's from the input tape to Tapes 2 and 3. The input tape is scanned from left to right and each time an a is read, one a is written on Tape 2, and similarly for b's on Tape 3.

After completing this, we write $2^{|w|_a}$ on Tape 4, based on the count on Tape 2.

- Verify that there is at least one a on Tape 2. If not, then we check if the input word was b, which is the only word that has zero a's. Otherwise, we proceed to the next step.
- Write two a's on Tape 4, and erase one a from Tape 2.
- For each a on Tape 2, do the following steps:
  - For each a on Tape 4, write another a on Tape 4. Do this by crossing off a's on Tape 4 and then writing a b at the end of the tape. After all the a's are crossed off, then we can convert all of the symbols on the tape (crossed off a's and b's) back to a's.
  - Then we mark off one of the a's on Tape 2.
- When all of the a's are marked off on Tape 2, there are 2^n a's on Tape 4 when there are $n-1$ a's on Tape 2.
- Compare the number of a's on Tape 4 to the number of b's on Tape 3. If they are equal, answer yes, otherwise, answer no

have TM that accept it
but it could be in
inf loop.

10. Show that the following language is $\underline{\text{r.e.}}$:

$$L = \{e(M_1)\#e(M_2) \ : \ L(M_1) \cap L(M_2) \neq \emptyset\}$$

Solution idea covered in review.

11. Show that the following problem is undecidable by reduction: Given a TM M, is $L(M)$ a finite language?

$$L = \{e(M_1)\#e(M_2) \ : \ L(M_1) \cap L(M_2) \neq \emptyset\}$$

Solution idea covered in review.

12. Show that the following language is decidable: $L_{ND} = \{e(M) : M \text{ is a nondeterministic TM }\}$.

Solution: This problem is decidable, unlike most other problems we have seen about TMs, because **this question asks about the TM, not about** $L(M)$.

To show that this is decidable, consider the encoding of a TM. It contains encodings of transitions in some order. To determine if the machine is nondeterministic, we design a two tape TM. Tape 1 is the input tape and Tape 2 is the tape that holds the current state,letter pair:

- To begin, we look at the first transition of the encoding. We copy the state and letter to Tape 2. (This is a block of 0s followed by a 1, followed by another block of 0s - it represents the state that we are in and the letter we are reading in the transition that is being encoded.)
- Moving right from that transition, we look at all other transitions. If they encode the same state-letter combination, we stop and answer yes - the TM is nondeterministic. We match this by simply comparing the number of 0s on Tape 2 with the number of 0s on the current transition on Tape 1.
- If we get to the right end of the encoding of the TM, we then move left and find the transition that we are currently looking at in the encoding. (it's the only one with the current state-letter combination.)
- We then move one transition to the right in the encoding and copy this state-letter combination to Tape 2.
- We repeat the above steps - move to the right to try to find a matching transition.
- If we get to the end of the encoding, rreturn no.

This algorithm always stops and either say yes or no, so the problem is decidable.

13. Show that the following problem is either decidable or undecidable: Given a CFG G is $L(G)$ infinite? (Hint: review the proof of the pumping lemma for CFLs.)

Solution: By the pumping lemma for CFLs, we know that if a nonterminal repeats itself in a derivation, then we have infinitely many words in the language. On the other hand, if there is no way for a nonterminal to repeat itself in any derivation, then the language is finite. We can only generate finitely many words because each derivation can only have each nonterminal in it at most once.

To determine if a nonterminal can repeat in a derivation, we use a modification of the 'reachable' algorithm from the conversion algorithm for CNF. First, we assume that the grammar is in CNF. In particular, all nonterminals in the grammar are useful and there are no unit productions.

Now, for a nonterminal A, let $r(A)$ be the nonterminals in the grammar that can be reached from A.

- initially, if $A \to \alpha$ is some production and B is one of the nonterminals in α, then put B in $r(A)$.

- Repeat this step until the set $r(A)$ does not change (i.e., a closure algorithm): if $B \in r(A)$ and $B \to \beta$ is a production, put all of the nonterminals in β into $r(A)$.

After doing this for all nonterminals, if there is a nonterminal $A \in r(A)$, then there is a derivation $A \Rightarrow^* \alpha A \beta$ for some $\alpha, \beta \in (V \cup \Sigma)^*$. Because there are no unit productions, we must have that $\alpha\beta \neq \varepsilon$. Also, since none of the symbols in α, β are useless, and A is not useless, then we know that α, A, β can all derive words, say v, w, x and that $S \to^* \gamma A \delta$ for some $\gamma, \delta \in (V \cup \Sigma)^*$. Again, γ, δ do not contain any useless symbols, so they can derive words u, y. Then similar to the proof of the pumping lemma, $L(G)$ is infinite as $uv^iwx^iy \in L(G)$ for all $i \geq 0$.