CIT 596 Homework 5

Steven Tomcavage
stomcava@seas.upenn.edu

March 23, 2011

1 Exercise 2.31

Let B be the language of all palindromes over $\{0, 1\}$ containing an equal number of 0s and 1s. Show that B is not context free.

Proof.

1. Given $\Sigma = \{0, 1\}$.
2. Assume that $B = \{ab | a, b \in \Sigma^\star$ and $b = a^R$ and the count of 0s in $w =$ the count of 1s in $w\}$ is context free.
3. Let p be the pumping length of B.
4. Let $s = w^p w^p = uvxyz$, where $|vy| > 0$ and $|vxy| \leq p$.
5. In the case where $u = \epsilon$, $|z| = p^2 - |vxy|$. Pumping s gives $v^i x y^i z$. Since $|vxy| \leq p$, then $|x| < p$ and $|z| > p$, and the number of 0s and 1s in a and b are not equal.
6. In the case where $z = \epsilon$, pumping s gives a string which is not valid because of similar arguments in the previous case.
7. Therefore, s cannot be pumped, which means that B is not context free. □

2 Exercise 2.36

Give an example of a language that is not context free but that acts like a CFL in the pumping lemma. Prove that your example works.

Consider the language $F = \{a^i b^j c^k \mid i, j, k \geq 0$ and if $i = 1$ then $j = k\}$. It is not a CFL because it is not a regular language (see Exercise 1.54). But it can be pumped like a CFL in the case where $i > 1$.

Proof.

1. Let $s = a^i b^j c^k = uvxyz$.
2. If $u = a^i$ and $z = c^k$, then s can be pumped, giving $uv^i xy^i z$, where $v^i xy^i$ is an expanding series of bs. □

3 Exercise 1.54

Consider the language $F = \{a^i b^j c^k \mid i, j, k \geq 0$ and if $i = 1$ then $j = k\}$.

3.1 Part a

Show that F is not regular.

Proof.

1. Assume that F is a regular language. Then there must be some pumping length p.
2. Consider the case where $s = a^1 b^p c^p = xyz$.
3. The character a^1 must be contained by x. Since $|xy| \leq p$, then $|y| < |b^p|$, so z contains some portion of b follwed by c^p. Pumping s gives a string where $|a| = 1$, but $|b| \neq |c|$, which is not a valid string in F.
4. Therefore, F is not a regular language. □

3.2 Part b

Show that F acts like a regular language in the pumping lemma. In other words, give a pumping length p and show that F satisfies the three conditions of the pumping lemma for this value of p.

Proof.

1. Let $i > 1$ so the constraint that $j = k$ is removed.
2. Let $p = 4$, so the string $w = aabc = xyz$.
3. If $x = \epsilon$, then $aabc = yz$, so $y = aa$. Pumping this gives $y^2 z = aaaabc$, which is still valid in F.
4. If $z = \epsilon$, then $aabc = xy$, so $y = c$. Pumping this gives $xy^2 = aabcc$, which is still valid in F.
5. If $|x| > 0$ and $|z| > 0$, then $aabc = xyz$, so $y = b$. Pumping this gives $xy^2 z = aabbc$, which is still valid in F.
6. Therefore, F satisfies the pumping lemma when $i > 1$. □

3.3 Part c

Explain why parts a and b do not contradict the pumping lemma.

Parts A and B do not contradict the pumping lemma because F represents two types of languages. When $i = 1$, then F is not a regular language and cannot be pumped. When $i > 1$, then F is a regular language and can be pumped.

4 Exercise 2.40

Say that a language is prefix-closed if the prefix of any string in the language is also in the language. Let C be an infinite, prefix-closed, context-free language. Show that C contains an infinite regular subset.

Proof.

1. Since C is a CFL, it must be pumpable.
2. Let p be the pumping length.
3. Let $C = uvxyz$ where $|vy| > 0$ and $|vxy| \leq p$.
4. So for every $i \geq 0$, $uv^i xy^i z$ is in C.

5. Since C is prefix-closed, then every substring of C that starts with the first character of the string is also in C.

6. Therefore, the infinite regular language $uv*$ is a subset of C.

$\square$

5 Exercise 2.44

If A and B are languages, define $A \diamond B = \{xy \mid x \in A \text{ and } y \in B \text{ and } |x| = |y|\}$. Show that if A and B are regular languages, then $A \diamond B$ is a CFL.

To satisfy the condition that $|x| = |y|$, the use of a stack is required when building a machine to recognize $A \diamond B$. A stack cannot be used in a DFA or NFA, but it can be used in an PDA. Thus, $A \diamond B$ can only be recognized by an PDA. By lemma 2.27 in Sipser, if a PDA recognizes a language, then that language is a CFL. Thus, $A \diamond B$ is a CFL.

6 Exercise 3.1d

Using the TM, M_2 from Example 3.7 in Sipser, give the sequence of configurations that M_2 enters when started on the string 000000.

1. $q_1000000$
2. blank q_200000
3. blank xq_30000
4. blank $x0q_4000$
5. blank $x0xq_300$
6. blank $x0x0q_40$
7. blank $x0x0xq_3$
8. blank $x0x0q_5x$
9. blank $x0xq_50x$
10. blank $x0q_5x0x$
11. blank xq_50x0x
12. blank q_5x0x0x
13. blank q_2x0x0x
14. blank xq_20x0x
15. blank xxq_3x0x
16. blank $xxxq_30x$
17. blank $xxx0q_4x$
18. blank $xxx0xq_4$
19. blank $xxx0xq_{reject}$

7 Exercise 3.2e

Using the TM, M_1 from Example 3.9 in Sipser, give the sequence of configurations that M_1 enters when started on the string 10#10.

1. $q_1 10\#10$
2. $xq_3 0\#10$
3. $x0q_3\#10$
4. $x0\#q_5 10$
5. $x0q_6\#x0$
6. $xq_7 0\#x0$
7. $q_7 x0\#x0$
8. $xq_1 0\#x0$
9. $xxq_2\#x0$
10. $xx\#q_4 x0$
11. $xx\#xq_4 0$
12. $xx\#q_6 xx$
13. $xxq_6\#xx$
14. $xq_7 x\#xx$
15. $xxq_1\#xx$
16. $xx\#q_8 xx$
17. $xx\#xq_8 x$
18. $xx\#xxq_8$
19. $xx\#xxq_{accept}$

8 Exercise 3.9

8.1 Part a

Show that 2-PDAs are more powerful than 1-PDAs.

Proof.

1. Given that a k-PDA has k stacks, so a 1-PDA is PDA with one stack which can recognize a CFL.
2. Let $\Sigma = \{0, 1\}$.
3. The language $B = a^i b^i c^i \mid a, b, c \in \Sigma$ is not a CFL and thus cannot recognized by a 1-PDA.
4. However, a 2-PDA can recognize the language B, by pushing a symbol onto both stacks for every a and then popping from one stack for every b and popping from the other stack for every c. If both stacks are empty at the end of the string, the string is in the language B.
5. Therefore, a 2-PDA is more powerful than a 1-PDA. $\square$

8.2 Part b

Show that 3-PDAs are not more powerful than 2-PDAs.

Since stacks operate by pushing items onto them and then popping items off, they lend themselves to CFLs where productions are mirrored across the halves of the string. With a 2-PDA, you can go from languages in the form a^ib^i to $a^ib^ic^i$. Logically this could be expanded so that a 3-PDA would permit the language $a^ib^ic^id^i$, but this language can be expressed by a 2-PDA by pushing a symbol a stack while reading a, then popping from one stack and pushing to the other while reading b, and likwise for c, then finally popping from the stack for d. In this way, a 2-PDA is equivalent to a 3-PDA.

9 Exercise 3.13

Show that a Turing machine with a stay put instruction instead of a left instruction is not equivalent to the usual Turing machine.

Proof.

1. Given a Turing machine M_1 with a right and a stay put instruction, assume that M_1 is equivalent to a Turing machine, $M2$, with a left and right instruction.
2. Let B be the language $\{0^{2^n} \mid n \geq 0\}$.
3. Example 3.7 in Sipser shows that the machine described by M_2 can recognize the language B by using a method of recursively crossing-off every other 0 in the string until either no 0s remained, in which case it accepts the string, or an odd number of 0s greater than 1 remained, in which case it rejects the string.
4. Since M_1 cannot go left, it cannot recursively cross-off a pattern of 0s in the string. If M_1 is to succeed, it must count all 0s in one pass and then determine if that number is in 2^n. But determining that without recursively passing over the string requires a Turing machine that is hard-wired to recognize a bounded number of elements in 2^n, which does not match the power of M_2.
5. The ability for M_1 to make a single pass over a string and write and read to a tape matches the ability of a PDA, not a full Turing machine.
6. Therefore, M_1 is not equivalent to M_2. □

10 Exercise 3.15

10.1 Part a

Show that the collection of decidable languages is closed under union.

Proof.

1. Let L_1 and L_2 be decidable languages.
2. Let M_1 be the machine that decides L_1.
3. Let M_2 be the machine that decides L_2.
4. Build a machine, M', that decides $L_1 \bigcup L_2$.
5. M' runs the string through both M_1 and M_2. If either M_1 or M_2 accepts the string, M' accepts it. If both M_1 and M_2 reject the string, then M' rejects it.
6. Therefore, M' is a decider for $L_1 \bigcup L_2$ and the collection of decidable languages is closed under union. □

10.2 Part b

Show that the collection of decidable languages is closed under concatenation.

Proof.

1. Let L_1 and L_2 be decidable languages.
2. Let M_1 be the machine that decides L_1.
3. Let M_2 be the machine that decides L_2.
4. Build a machine, M', that decides L_1L_2.
5. M' connects M_1 and M_2 by bypassing the accept state of M_1 and connecting to the start state of M_2 if more input exists at the state where M_1 would normally move to the accept state. If M_2 reaches the accept state, then M' accepts the string. If either M_1 or M_2 reject the string, then M' rejects the string.
6. Therefore, M' is a decider for L_1L_2 and the collection of decidable languages is closed under concatenation. □

10.3 Part c

Show that the collection of decidable languages is closed under star.

Proof.

1. Let L_1 be a decidable language.
2. Let M_1 be the machine that decides L_1.
3. Build a machine, M', that decides L_1^*.
4. M' is built by taking M_1 and adding a transition back to the start state if M_1 would normally transition to the accept state but more input exists. If at any time M_1 rejects w, then M' rejects w. If the string ends on the accept state of M_1, then M' accepts the string.
5. Therefore, M' is a decider for L_1^* and the collection of decidable languages is closed under star. □

10.4 Part d

Show that the collection of decidable languages is closed under complementation.

Proof.

1. Let L_1 be a decidable language.
2. Let M_1 be the machine that decides L_1.
3. Build a machine, M', that decides $\not{L}_1$.
4. M' is built by running the input through M_1. If M_1 accepts the string, M' rejects it. If M_1 rejects the string, M' accepts it.
5. Therefore, M' is a decider for $\not{L}_1$ and the collection of decidable languages is closed under complementation. □

10.5 Part e

Show that the collection of decidable languages is closed under intersection.

Proof.

1. Let L_1 and L_2 be decidable languages.
2. Let M_1 be the machine that decides L_1.
3. Let M_2 be the machine that decides L_2.
4. Build a machine, M', that decides $L_1 \bigcap L_2$.
5. M' runs the string through both M_1 and M_2. If both M_1 and M_2 accept the string, M' accepts it. If either M_1 or M_2 rejects the string, then M' rejects it.
6. Therefore, M' is a decider for $L_1 \bigcap L_2$ and the collection of decidable languages is closed under intersection. □

11 Exercise 3.16b

Show that the collection of Turing-recognizable languages is closed under concatenation.

Proof.

1. Let L_1 and L_2 be Turing-recognizable languages.
2. Let M_1 be the machine that recognizes L_1.
3. Let M_2 be the machine that recognizes L_2.
4. Build a machine, M', that recognizes $L_1 L_2$.
5. M' connects M_1 and M_2 by bypassing the accept state of M_1 and connecting to the start state of M_2 if more input exists at the state where M_1 would normally move to the accept state. If M_2 reaches the accept state, then M' accepts the string. If either M_1 or M_2 reject the string, then M' rejects the string. If either M_1 or M_2 do not halt, then M' does not halt.
6. Therefore, M' is a recognizer for $L_1 L_2$ and the collection of Turing-recognizable languages is closed under concatenation. □

12 Exercise 4.15

Let $A = \{\langle R \rangle \mid R$ is a regex describing a language containing at least one string w that has 111 as a substring $\}$. Show that A is decidable.

Proof.

1. Since R is a regex, then it is also a regular language.
2. Let Σ be the alphabet of R.
3. The machine R can be described by a finite string using the alphabet $\{\Sigma, *, +\}$. The alphabet does not contain ϵ since R contains at least the string 111. The description of R does not contain any loops or recursion, so it has an equivalent DFA.
4. If the DFA of $\langle R \rangle$ accepts, then A accepts. If the DFA of $\langle R \rangle$ rejects, then A rejects.
5. Therefore, $\langle R \rangle$ is decidable. □

13 Exercise 4.16

Prove that EQ_{DFA} is decidable by testing the two DFAs on all strings up to a certain size. Calculate a size that works.

This question seems to be asking about specific DFAs, but I can't find any reference in the text to those DFAs.

14 Exercise 4.19

Let $S = \{\rangle M \langle \mid M$ is a DFA that accepts w^R whenever it accepts w$\}$. Show that S is decidable.

Proof.

1. Create a Turing machine T that takes $\{\langle M, w \rangle\}$ as its input.
2. The machine T simulates M on the string w. Whenever M accepts w, T accepts. Whenever M rejects, then T rejects.
3. Since M is a DFA, it will always either accept or reject, so T will always either accept or reject.
4. Therefore, S is decidable. □

15 Exercise 3.3

Using lambda calculus, create the IMPLIES function.

Since $p \implies q$ is equivalent to $\not p \vee q$, the lambda calculus function for implies is ite(p, ite(q, T, F), T).

16 Exercise 2.20a

Let $A/B = \{w \mid wx \in A$ for some $x \in B\}$. Show that, if A is a Turing machine and B is regular, then A/B is a Turing machine.

Proof.

1. If A is a Turing machine, then it is either Turing recognizable or Turing decidable.
2. B will always accept or reject, so since it is at the end of A, it implies that A is Turing decidable.
3. Removing a portion from the end of a Turing decidable language may make it Turing recognizable, but it will not weaken it further than that. It may be reduced to a PDA or a DFA, but those are subsets of Turing recognizable languages.
4. Therefore, A/B is a Turing machine. □

17 Exercise 2.20b

Let $A/B = \{w \mid wx \in A$ for some $x \in B\}$. Show that, if A is the CFL $S \rightarrow aSb \mid SS \mid \epsilon$ and B is the regular language $(\Sigma\Sigma)^+$ where $\Sigma = \{a, b\}$, then A/B is regular.

Proof.

1. Since B must contain at least two characters, then the minimum length A is the string ab, so A/B would be ϵ.

2. A is always built up from two character strings of a and b, whether it is in the form ab, $aabb$, $abab$, or $aabbaabb$.
3. The maximum length of A/B for the examples listed above would be ϵ, aa, ab, and $aabbaa$.
4. Since B is in multiples of two characters, it will remove aa, bb, or ab from the end of productions of A.
5. Therefore, the language A/B can be expressed by the regular expression $(a^+b^\star)^\star$. $\square$