Final

March 17, 2017

(804501476)

Problem 1.0:
Prove

$$L_1 \diamond L_2 = \{xy \mid x \in L_1, y \in L_2, \text{ and } |x|= 2|y|\} \tag{1.1}$$

is not context free.
Let $L_1 = \{0^{2n}1^{2n}\}$ and $L_2 = \{0^n 1^n\}$. Then,

$$L_1 \diamond L_2 = \{0^{2n}1^{2n}0^n 1^n \mid x \in L_1, y \in L_2, \text{ and } |x|= 2|y|\} \tag{1.2}$$

Proof.
Towards contradiction assume $L_1 \diamond L_2$ is context-free.

- By the pumping lemma $\exists$ pumping length p.
- Let $w = 0^{2p}1^{2p}0^p 1^p \in L_1 \diamond L_2$ and $|w| \geq p$.
- By pumping lemma $0^{2p}1^{2p}0^p 1^p = abcde$ s.t:
  1. $|bd| \geq 1$
  2. $|bcd| \leq p$

Case 1: $bcd = 0^\alpha 1^\beta$ (on the left side)

- We pump down then we have either:
  1. $ace = 0^{2p-\alpha}1^{2p}0^p 1^p \notin L_1 \diamond L_2$, since $2p - \alpha + 2p = 4p \implies \alpha = 0$ and $1 \leq \alpha \leq p$ $\implies\impliedby$
  2. $ace = 0^{2p}1^{2p-\beta}0^p 1^p \notin L_1 \diamond L_2$, since $2p - \beta + 2p = 4p \implies \beta = 0$ and $1 \leq \beta \leq p$ $\implies\impliedby$
  3. $ace = 0^{2p-\alpha}1^{2p-\beta}0^p 1^p \notin L_1 \diamond L_2$, since $2p - \alpha + 2p - \beta = 4p \implies \alpha + \beta = 0$ and $1 \leq \alpha + \beta \leq p \implies\impliedby$

Case 2: $bcd = 0^\alpha 1^\beta$ (on the right side)

- We pump up then we have either:
  1. $ace = 0^{2p}1^{2p}0^{p+\alpha}1^p \notin L_1 \diamond L_2$, since $2(p + \alpha + p) = 4p \implies \alpha = 0$ and $1 \leq \alpha \leq p$ $\implies\impliedby$
  2. $ace = 0^{2p}1^{2p}0^p 1^{p+\beta} \notin L_1 \diamond L_2$, since $2(p + \beta + p) = 4p \implies \beta = 0$ and $1 \leq \beta \leq p$ $\implies\impliedby$

3. $ace = 0^{2p}1^{2p}0^{p+\alpha}1^{p+\beta} \notin L_1 \diamond L_2$, since $2(p+\alpha+p+\beta) = 4p \implies \alpha+\beta = 0$ and $1 \leq \alpha+\beta \leq p \Longrightarrow\Longleftarrow$

Case 3: $bcd = 1^{\alpha}0^{\beta}$ (middle)

- We pump down then we have either:
  1. $ace = 0^{2p}1^{2p-\alpha}0^{p}1^{p} \notin L_1 \diamond L_2$, since $2p-\alpha+2p = 4p \implies \alpha = 0$ and $1 \leq \alpha \leq p$ $\Longrightarrow\Longleftarrow$
  2. $ace = 0^{2p}1^{2p}0^{p-\beta}1^{p} \notin L_1 \diamond L_2$, since $2(p-\beta+p) = 4p \implies \beta = 0$ and $1 \leq \beta \leq p$ $\Longrightarrow\Longleftarrow$
  3. $ace = 0^{2p}1^{2p-\alpha}0^{p-\beta}1^{p} \notin L_1 \diamond L_2$, since $2p-\alpha+2p = 2(p-\beta+p) \implies \beta = \alpha$. This is true if $\alpha = \beta = 0$ but $1 \leq \beta \leq p \Longrightarrow\Longleftarrow$. We can also have that $\alpha = \beta$ is true if p is even and each is half of p. However this destroys symmetry in L_1 and L_2, $0^{2p}1^{2p-\alpha} \notin L_1$ and $0^{p-\beta}1^{p} \notin L_2 \Longrightarrow\Longleftarrow$.

□

Problem 2.0:

(a)

Show that

$$HALT = \{(\langle M \rangle, x) \mid M \text{ halts on input } x\} \tag{2.3}$$

is oracle decidable.

Proof.
We construct OBTM $O(\langle M \rangle, x)$:

- O writes $\langle M \rangle$ to machine tape and w to input tape.
- O enters query state:

  1: $x \in L(M)$ then accept.
  2: $x \notin L(M)$ the reject.

The query is immediate therefore if $x \notin L(M)$, we can reject without looping. Therefore O always terminates thus it is a decider for HALT. □

Problem 2.0 continued on next page...

(b)

Show that

$$NEQ = \{(\langle M_1 \rangle, \langle M_2 \rangle) \mid L(M_1) \neq L(M_2)\} \tag{2.4}$$

is oracle recognizable.

Proof.
We construct OBTM $O(\langle M_1 \rangle, \langle M_2 \rangle)$:
Tapes:
In class we showed that a multiple tapes can be simulated with a single tape so we split the regular tape into 4 tapes w_1, w_2, w_3, and w_4.

1 Write $\langle M_1 \rangle$ onto w_1

2 Write $\langle M_2 \rangle$ onto w_2

3 Will keep a binary count starting at 0 in w_3.

   - We are assuming that all strings can be converted to binary.

4 Will maintain a tuple starting at $(\$, \$)$ in w_4

States:
We will have states S_1, S_{oracle}, S_3, S_4, q_{accept}

S_1: Write contents of tape w_1 onto the machine tape and contents of w_3 onto the input tape.

S_{oracle}: Enter query state and write the contents of the first cell in the input tape onto w_4 and move head of w_4 right.

S_3: Clear the machine tape and write the contents of w_2 onto machine tape. Clear input tape and write w_3 onto input tape.

S_4: Reset tape w_4 to (\$,\$) and increment tape w_3 by one.

Transitions:

$$\delta(S_1, w_4 = (\$, \$)) \to (S_{oracle}) \tag{2.5}$$
$$\delta(S_{oracle}, w_4 = (x, \$)) \to (S_3),\ x \in \{0, 1\} \tag{2.6}$$
$$\delta(S_3, w_4 = (x, \$)) \to (S_{oracle}) \tag{2.7}$$
$$\delta(S_3, w_4 = (x, y)) \to (q_{accept}) \text{ if } x \neq y \tag{2.8}$$
$$\delta(S_3, w_4 = (x, y)) \to (S_4) \text{ if } x = y \tag{2.9}$$
$$\delta(S_4, w_4) \to (S_1) \tag{2.10}$$

This is still an OBTM as we have not changed the function of query and do not misuse the input and machine tapes. We make use of the regular tape like a tape of any TM and supplied states

Problem 2.0 continued on next page...

which allow us to recogize NEQ by determining whether the binary representations of strings is ever accepted by one an rejected by the other if so we will accept. If not the machine will continue to process strins and potentially loop if these two machines indeed accept the same language. □

(c)

The language:

$$Infinite = \{\langle M\rangle \mid |L(M)| = \infty\} \tag{2.11}$$

is not oracle recognizable. An OBTM that would try to recognize this language would have to check and infinite amount of strings to determine whether they all belong to M and so it would never halt.

Problem 3.0:

(a)

Show that

$$CLOSEBY = \{(\langle M_1\rangle, \langle M_2\rangle) \mid \forall x \in L(M_1)\ \exists y \in L(M_2) : ||x - y|| \leq 1\} \tag{3.12}$$

is undecidable.

Proof. Assume for contradiction $\exists$ a decider D for CLOSEBY, create a TM N:

- $N(w)$:
  - Let $u = \langle N\rangle$, by Recursion Theorem.
  - Let $\langle M\rangle$ be a TM that only accepts ε.
  - Run $D(u, \langle M\rangle)$:
    1: $D(u, \langle M\rangle)$: Accepts
      · Accept all w
    2: $D(u, \langle M\rangle)$: Rejects
      · Accept w iff $w = \varepsilon$

Analysis:

- Case 1: $D(u, \langle M\rangle)$: **Accepts** $\implies L(N) = L(M) = \{\varepsilon\}$. This is true since the length of ε is zero $\implies$ the only string in $L(N)$ is ε. However we accept all $w \implies L(N) = \{0, 1\}^*$ and this is contradiction.
- Case 2: $D(u, \langle M\rangle)$: **Rejects** $\implies L(N) \neq L(M)$, since $L(M) = \{\varepsilon\}$. However we only accept $\varepsilon \implies L(N) = \{\varepsilon\}$ and this is a contradiction.

□

Problem 3.0 continued on next page...

(b)

Show that

$$CLOSEBY = \{(\langle M_1\rangle, \langle M_2\rangle) \mid \forall x \in L(M_1)\ \exists y \in L(M_2) : ||x - y|| \leq 1\} \tag{3.13}$$

is unrecognizable.

Proof. Assume for contradiction $\exists$ a recognizer R for CLOSEBY, create a TM N:

- $N(w)$:
  - Let $u = \langle N\rangle$, by Recursion Theorem.
  - Let $\langle M\rangle$ be a TM that only accepts ε.
  - if $w = \varepsilon$ accept.
  - Run $R(u, \langle M\rangle)$:
    1: $R(u, \langle M\rangle)$: Accepts
      · Accept all w
    2: $R(u, \langle M\rangle)$: Rejects
      · Accept w iff $w = \varepsilon$

Analysis:

- Case 1: $R(u, \langle M\rangle)$: **Accepts** $\implies$ $L(N) = L(M) = \{\varepsilon\}$. This is true since the length of ε is zero $\implies$ the only string in $L(N)$ is ε. However we accept all w $\implies$ $L(N) = \{0, 1\}^*$. $\Longrightarrow\Longleftarrow$
- Case 2: $R(u, \langle M\rangle)$: **Rejects** $\implies$ $L(N) \neq L(M)$, since $L(M) = \{\varepsilon\}$. However we only accept ε $\implies$ $L(N) = \{\varepsilon\}$. $\Longrightarrow\Longleftarrow$
- Case 3: $R(u, \langle M\rangle)$: **Loops** $\implies$ $L(N) \neq L(M)$, since $L(M) = \{\varepsilon\}$. But by construction $L(N) = \{\varepsilon\}$. $\Longrightarrow\Longleftarrow$

□

(c)

Show that

$$LEQ - HALT = \{(\langle M\rangle, \langle N\rangle) \mid \forall x \in \Sigma^* : M(x) \text{ halts in fewer steps than } N(x)\} \tag{3.14}$$

is unrecognizable.

Proof. Assume for contradiction $LEQ - HALT$ is regular $\implies$ *exists* an enumarator E for $LEQ - HALT$. We construct M:

- M(w):
  - Let $z = \langle x\rangle$, by Recursion Theorem.
  - $task_A =$ run $E(\varepsilon)$.
  - $task_B =$ look at the sequence of produced by E. Wait until we find a tuple of form $(z, \langle N\rangle)$ is found.
  - run $task_A$ and $task_B$ in parallel.
    - When $task_B$ finishes, run N(w).

This is a contradicition because $\langle N\rangle$ now takes longer than $\langle N\rangle$ despite $(\langle M\rangle, \langle N\rangle) \in LEQ - HALT$. □

Problem 4.0:
Show that

$$ALICE = \{(M, R) \mid (M, R) \text{ is recognizable}\} \tag{4.15}$$

is undecidable.
We can convert this problem into a tiling problem and we also convert a turing machine into the tiling,

Tiles: have the following elements within them

1 $w_i \in \{ON, OFF\}$

2 $q_i \in Q$ if head is in cell or ☻ if head is not in cell.

3 r, the result of i % M, where % is the modulo operator and i is the position of the head on the tape.

There are boundry tiles which are marked by an X.

Problem 4.0 continued on next page. . .

Rules:

Initial State: For simpicity assume ␣ = ␣\☻

X	q_0 / ␣	␣
X	X	X

q_0 / ␣	␣	␣
X	X	X

␣	␣	␣
X	X	X

The rules $(b_1, \ldots, b_n) \to (c_1, \ldots, c_n)$ in Alice in Turing Land correspond to tiling rules where different n will create a different tiling size. These tiling rules correspond to a set of transitions for a TM. For simplicity assume that $w_i = w_i \backslash$☻$, r$ we always start with the head in a position where $r = 0$.

Transition Right:

w_{i-1}	w_i	w'_{i+1}	w'_{i+2}	$\cdots$	w'_{i+n}
w_{i-1}	w_i	w'_{i+1}	w'_{i+2}	$\cdots$	q_{i+n} / w_{i+n}
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
w_{i-1}	w_i	w'_{i+1}	w'_{i+2}	$\cdots$	w_{i+n}
w_{i-1}	w_i	w'_{i+1}	q_{i+2} / w_{i+2}	$\cdots$	w_{i+n}
w_{i-1}	w_i	q_{i+1} / w_{i+1}	w_{i+2}	$\cdots$	w_{i+n}
w_{i-1}	$0,\ q_i$ / w_i	w_{i+1}	w_{i+2}	$\cdots$	w_{i+n}

$$\begin{cases} \delta(w_i, q_i) \to (w_i, q_{i+1}, R) \\ \delta(w_{i+1}, q_{i+1}) \to (w'_{i+1}, q_{i+2}, R) \\ \delta(w_{i+2}, q_{i+2}) \to (w'_{i+2}, q_{i+3}, R) \\ \qquad\qquad \cdots \\ \delta(w_{i+(n-1)}, q_{i+(n-1)}) \to (w'_{i+(n-1)}, q_{i+n}, R) \\ \delta(w_{i+n}, q_{i+n}) \to (w'_{i+n}, q_{i+(n+1)}, R) \end{cases} \tag{4.16}$$

Transition Left:

w'_{i-n}	$\cdots$	w'_{i-2}	w'_{i-1}	w_i	w_{i+1}
w_{i-n} / q_{i-n}	$\cdots$	w'_{i-2}	w'_{i-1}	w_i	w_{i+1}
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
w_{i-n}	$\cdots$	w'_{i-2}	w'_{i-1}	w_i	w_{i+1}
w_{i-n}	$\cdots$	w'_{i-2} / q_{i-2}	w'_{i-1}	w_i	w_{i+1}
w_{i-n}	$\cdots$	w_{i-2}	w_{i-1} / q_{i-1}	w_i	w_{i+1}
w_{i-n}	$\cdots$	w_{i-2}	w_{i-1}	w_i / $0,\ q_i$	w_{i+1}

$$\begin{cases} \delta(w_i, q_i) \rightarrow (w_i, q_{i-1}, L) \\ \delta(w_{i-1}, q_{i-1}) \rightarrow (w'_{i-1}, q_{i-2}, L) \\ \delta(w_{i-2}, q_{i-2}) \rightarrow (w'_{i-2}, q_{i-3}, L) \\ \qquad\qquad \cdots \\ \delta(w_{i-(n-1)}, q_{i-(n-1)}) \rightarrow (w'_{i-(n-1)}, q_{i-n}, L) \\ \delta(w_{i-n}, q_{i-n}) \rightarrow (w'_{i-n}, q_{i-(n+1)}, L) \end{cases} \tag{4.17}$$

Edge Cases:

Head in left corner

w_i	w'_{i+1}	w'_{i+2}	$\cdots$	w'_{i+n}
w_i	w'_{i+1}	w'_{i+2}	$\cdots$	w_{i+n} / q_{i+n}
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
w_i	w'_{i+1}	w'_{i+2}	$\cdots$	w_{i+n}
w_i	w'_{i+1}	w_{i+2} / q_{i+2}	$\cdots$	w_{i+n}
w_i	w_{i+1} / q_{i+1}	w_{i+2}	$\cdots$	w_{i+n}
w_i / $0,\ q_i$	w_{i+1}	w_{i+2}	$\cdots$	w_{i+n}

$$\begin{cases} \delta(w_i, q_i) \rightarrow (w_i, q_{i+1}, R) \\ \delta(w_{i+1}, q_{i+1}) \rightarrow (w_{i+1}^{'}, q_{i+2}, R) \\ \delta(w_{i+2}, q_{i+2}) \rightarrow (w_{i+2}^{'}, q_{i+3}, R) \\ \qquad\qquad \cdots \\ \delta(w_{i+(n-1)}, q_{i+(n-1)}) \rightarrow (w_{i+(n-1)}^{'}, q_{i+n}, R) \\ \delta(w_{i+n}, q_{i+n}) \rightarrow (w_{i+n}^{'}, q_{i+(n+1)}, R) \end{cases} \tag{4.18}$$

w_i	w_{i+1}	w_{i+2}	$\cdots$	w_{i+n}
w_i	w_{i+1}	w_{i+2}	$\cdots$	w_{i+n}
w_i	w_{i+1}	w_{i+2}	$\cdots$	w_{i+n}
$0,\ q_i$ w_i	w_{i+1}	w_{i+2}	$\cdots$	w_{i+n}

$$\begin{cases} \delta(w_i, q_i) \rightarrow (w_i, q_{i-1}, L) \\ \delta(w_{i-1}, q_{i-1}) \rightarrow (w_{i-1}^{'}, q_{i-2}, L) \\ \delta(w_{i-2}, q_{i-2}) \rightarrow (w_{i-2}^{'}, q_{i-3}, L) \\ \qquad\qquad \cdots \\ \delta(w_{i-(n-1)}, q_{i-(n-1)}) \rightarrow (w_{i-(n-1)}^{'}, q_{i-n}, L) \\ \delta(w_{i-n}, q_{i-n}) \rightarrow (w_{i-n}^{'}, q_{i-(n+1)}, L) \end{cases} \tag{4.19}$$

Head in right corner:

$w_{i-n}^{'}$	$\cdots$	$w_{i-2}^{'}$	$w_{i-1}^{'}$	w_i
q_{i-n} w_{i-n}	$\cdots$	$w_{i-2}^{'}$	$w_{i-1}^{'}$	w_i
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
w_{i-n}	$\cdots$	$w_{i-2}^{'}$	$w_{i-1}^{'}$	w_i
w_{i-n}	$\cdots$	q_{i-2} $w_{i-2}^{'}$	$w_{i-1}^{'}$	w_i
w_{i-n}	$\cdots$	w_{i-2}	q_{i-1} w_{i-1}	w_i
w_{i-n}	$\cdots$	w_{i-2}	w_{i-1}	$0,\ q_i$ w_i

Problem 4.0 continued on next page...

$$\begin{cases} \delta(w_i, q_i) \rightarrow (w_i, q_{i-1}, L) \\ \delta(w_{i-1}, q_{i-1}) \rightarrow (w_{i-1}^{'}, q_{i-2}, L) \\ \delta(w_{i-2}, q_{i-2}) \rightarrow (w_{i-2}^{'}, q_{i-3}, L) \\ \qquad\qquad\qquad \cdots \\ \delta(w_{i-(n-1)}, q_{i-(n-1)}) \rightarrow (w_{i-(n-1)}^{'}, q_{i-n}, L) \\ \delta(w_{i-n}, q_{i-n}) \rightarrow (w_{i-n}^{'}, q_{i-(n+1)}, L) \end{cases} \tag{4.20}$$

w_{i-n}	$\cdots$	w_{i-2}	w_{i-1}	w_i
w_{i-n}	$\cdots$	w_{i-2}	w_{i-1}	w_i
w_{i-n}	$\cdots$	w_{i-2}	w_{i-1}	w_i
w_{i-n}	$\cdots$	w_{i-2}	w_{i-1}	0, q_i w_i

$$\begin{cases} \delta(w_i, q_i) \rightarrow (w_i, q_{i+1}, R) \\ \delta(w_{i+1}, q_{i+1}) \rightarrow (w_{i+1}^{'}, q_{i+2}, R) \\ \delta(w_{i+2}, q_{i+2}) \rightarrow (w_{i+2}^{'}, q_{i+3}, R) \\ \qquad\qquad\qquad \cdots \\ \delta(w_{i+(n-1)}, q_{i+(n-1)}) \rightarrow (w_{i+(n-1)}^{'}, q_{i+n}, R) \\ \delta(w_{i+n}, q_{i+n}) \rightarrow (w_{i+n}^{'}, q_{i+(n+1)}, R) \end{cases} \tag{4.21}$$

Head occupyiung corner after transition:

r, q_i w_i	w_{i+1}	w_{i+2}	$\cdots$	w_{i+n}
w_i	w_{i+1}	w_{i+2}	$\cdots$	w_{i+n}

w_{i-n}	$\cdots$	w_{i-2}	w_{i-1}	r, q_i w_i
w_{i-n}	$\cdots$	w_{i-2}	w_{i-1}	w_i

In class we showed that if D decides Tiling then we can decide $M(\varepsilon)$ halts. Therfore since we can convert Alice in Turing Land into a Tiling problem and convert a turing machine into this tiling then we cannot decide Alice in turing land.

Problem 5.0:
First I will create a function

$$Pattern(i, j, b) = b^{i\times j} \tag{5.22}$$

This function takes 3 paraments

1 i - the width of a block.

2 j - the length of a block.

3 b - the base we will use to produce the patterns.

It will produce patterns and map them to a block with dimensions $i \times j$. For example:

$$Pattern(2,2,2) = \left\{ \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array} \; \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & 0 \\ \hline \end{array} \; \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 0 & 0 \\ \hline \end{array} \; \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 0 & 0 \\ \hline \end{array} \; \cdots \; \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 1 \\ \hline \end{array} \; \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & 1 \\ \hline \end{array} \right\} \tag{5.23}$$

(a)

```
Weaver 1:
for i from 1 → ∞: // Tile size 1 × i
    for p from 0 → 2^i: /* Here we traverse the pattern encoded with binary digits*/
```

...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4															•	...
3							•	•	•	•	•	•	•	•		...
2			•	•	•	•										...
1	•	•														...
	0	1	00	01	10	11	000	001	010	011	100	101	110	111	0000	...

(b)

```
Weaver 2:
for i from 1 → ∞:
    for j from 1 → i
        for p from 0 → 2^(i×j): /* Here we traverse the pattern encoded with binary digits*/
```

(c)

Problem 5.0 continued on next page. . .

```
Weaver 3:
for i from 1 → ∞:
    for j from 1 → i:
        for c in {c_i}:
            for p from 0 → c^{i×j}: /* Here we traverse the pattern encoded with binary digits*/
```

(d)