Homework 3 Solutions

Formal Languages and Automata (CS322)

Last update: October 19, 2025

1. Give a context-free grammar (CFG), which is a 4-tuple, that generates each of the following languages.

(a) $L_1 = \{w \in \{a, b\}^* : w \notin S\}$ where $S = \{a^n b^n : n \in \mathbb{N}\}$

(b) $L_2 = \mathcal{L}(R)$ where $R = S^+(\#S^+)^*$, $S = (cv \cup cvc \cup cvcc)$ are regular expression with the alphabet $\Sigma_2 = \{c, v, \#\}$.
Note: L_2 be an abstraction of *Hangul* sentence, where c, v, $\#$, S denotes consonant, vowel, space, *Hangul* syllable respectively.

Give an unambiguous CFG that generate the following.

(c) $L_3 = \{w \in \{a, b\}^* : \text{every prefix } u \text{ of } w \text{ satisfies } |u|_a \leq |u|_b\}$

Solution.

(a) CFG $G_1 = (V_1, \Sigma_1, R_1, S_1)$ generates L_1 where

- $V_1 = \{S, A, B, C, D\}$
- $\Sigma_1 = \{a, b\}$
- R_1 contains following rules:
  - $S \to aA \mid Bb \mid DbaD$
  - $A \to aA \mid C$
  - $B \to Bb \mid C$
  - $C \to aCb \mid \epsilon$
  - $D \to aD \mid bD \mid \epsilon$
- $S_1 = S \in V_1$ is the start symbol.

(b) CFG $G_2 = (V_2, \Sigma_2, R_2, S_2)$ generates L_2 where

- $V_2 = \{X, Y, Z, S\}$
- $\Sigma_2 = \{c, v, \#\}$
- R_2 contains following rules:
  - $X \to SZY$
  - $Y \to \#SZY \mid \epsilon$
  - $Z \to SZ \mid \epsilon$
  - $S \to cv \mid cvc \mid cvcc$
- $S_2 = X \in V_2$ is the start symbol.

(c) The context-free grammar $G_3 = (\{S, T\}, \{a, b\}, R, S)$ where the rules R are

$$\begin{array}{lcl} S & \to & TbS \mid T \\ T & \to & bTaT \mid \epsilon \end{array}$$

is unambiguous and generates L_3.

□

2. Construct pushdown automata (PDAs) recognizing the following languages. Along with a PDA, which is a 6-tuple, provide a transition diagram of it.

(a) $L_4 = \{w \in \{a, b\}^* : |w|_a = |w|_b + 3\}$

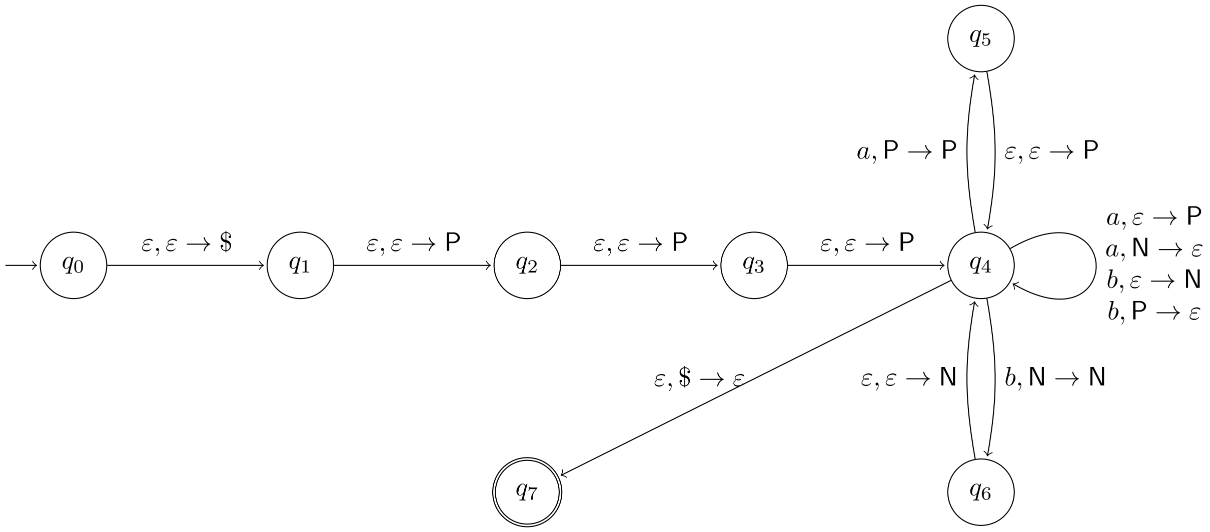
(b) $L_5 = \{w$ is a well-formed parentheses with `(`, `)`, `{`, `}`, `[`, `]`$\}$

Solution.

(a) PDA $P_4 = (Q_4, \Sigma_4, \Gamma_4, \delta_4, q_0, F_4)$ recognizes L_4 where

- $Q_4 = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$,
- $\Sigma_4 = \{a, b\}$,
- $\Gamma_4 = \{\mathsf{P}, \mathsf{N}, \$\}$,
- $F_4 = \{q_7\}$, and
- δ_4 is given by the following transition diagram:

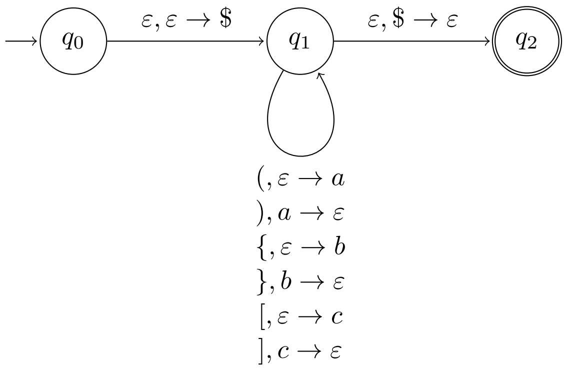
δ_4:



(b) PDA $P_5 = (Q_5, \Sigma_5, \Gamma_5, \delta_5, q_0, F_5)$ recognize L_5 where

- $Q_5 = \{q_0, q_1, q_2\}$
- $\Sigma_5 = \{$`(`, `)`, `{`, `}`, `[`, `]`$\}$
- $\Gamma_5 = \{\$, a, b, c\}$
- δ_5 is given by following transition diagram:

δ_5:



- $q_0 \in Q_5$
- $F_5 = \{q_2\}$

$\square$

3. (a) Convert the CFG $G_1 = (\{S, T\}, \{a, b\}, R_1, S)$ into an equivalent PDA, where the set of rules R_1 is:

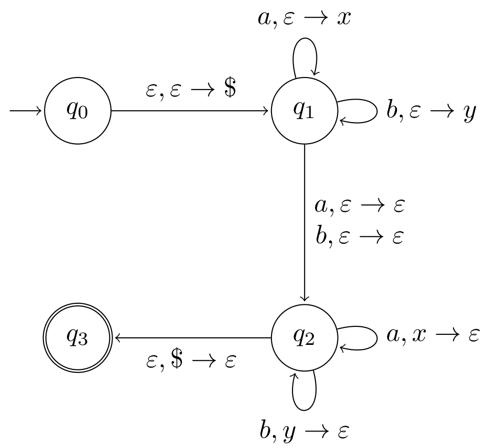
$S \to aSa \mid bT$

$T \to bTb \mid aS \mid \varepsilon$

Provide a transition diagram as well as a PDA.

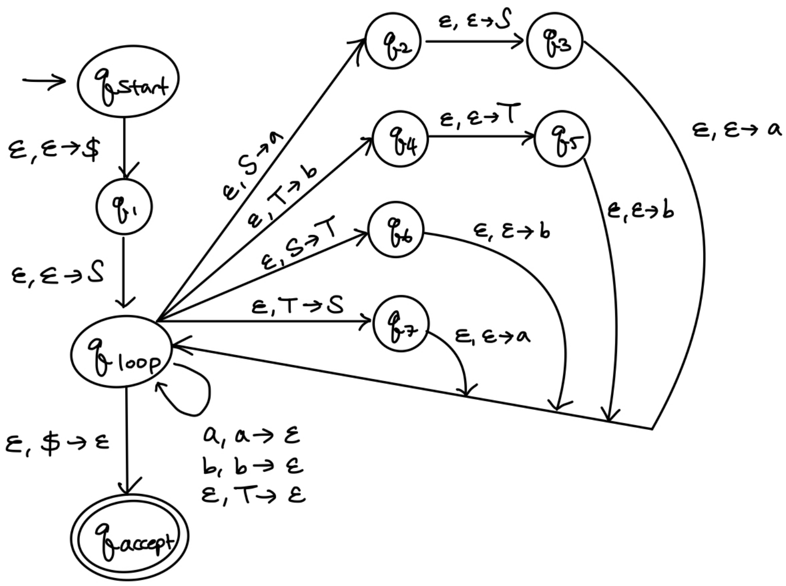
(b) Convert the PDA $P_1 = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{x, y, \$\}, \delta_1, q_0, \{q_3\})$ into an equivalent CFG where δ_1 is described below.

δ_1:



Solution.

(a) $M_1 = (\{q_{start}, q_{loop}, q_{accept}, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}, \{0, 1\}, \delta_1, q_{start}, \{q_{accept}\})$ with δ_1:



(b) We omit all variables that derive no strings. $G = (\{A_{03}, A_{11}, A_{12}, A_{22}\}, \{a, b\}, R, A_{03})$ with R:

$$A_{03} \to \epsilon A_{12} \epsilon$$

$$A_{11} \to \epsilon$$

$$A_{22} \to \epsilon$$

$$A_{12} \to A_{11}A_{12} \mid A_{12}A_{22} \mid aA_{12}a \mid bA_{12}b \mid a \mid b$$

This can be abbreviated into $G = (\{S\}, \{a, b\}, R, S)$ with R:

$$A \to aSa \mid bSb \mid a \mid b$$

□

4. Let $\Sigma = \{0, 1\}$. For languages $A, B \subseteq \Sigma^*$ define

$$A \diamond B = \{xy \mid x \in A, y \in B, |x| = |y|\}.$$

Also let

$$D = \{xy \mid x, y \in \Sigma^*, |x| = |y|, x \neq y\}.$$

Answer the following.

(a) Construct a CFG or a PDA that generates D. Give a formal description of your machine/grammar.

(b) Prove that if A and B are regular languages, then $A \diamond B$ is context-free by giving a CFG or a PDA.

Solution. Although a proof of correctness and intuitive justifications are given below, you were not required to write them as parts of your homework submission; they serve only to guide your understanding.

(a) **Claim.** Let $\Sigma = \{0, 1\}$ and let $G = (V, \Sigma, R, S)$ be the CFG with R consisting of the following rules:

$$S \to AB \mid BA, \qquad A \to TAT \mid 0, \qquad B \to TBT \mid 1, \qquad T \to 0 \mid 1.$$

Then $L(G) = D = \{xy \mid x, y \in \Sigma^*,\ |x| = |y|,\ x \neq y\}$. $\diamond$

Proof of the claim. For a variable X, denote with $L(X)$ the set of strings generated if the starting variable is X, *i.e.*, $L(X) := L(G_X)$, $G_X = (V, \Sigma, R, X)$. Note that $L(A) = \cup_{k \geq 0}\ \Sigma^k 0 \Sigma^k$ and $L(B) = \cup_{k \geq 0} \Sigma^k 1 \Sigma^k$.

Let $w \in D$ and $|w| = 2n$. Then there exists $i \in \{1, \ldots, n\}$ with $w_i \neq w_{n+i}$. By symmetry we may assume $w_i = 0$ and $w_{n+i} = 1$. Set $u := w_1 w_2 \cdots w_{2i-1}$ and $v := w_{2i} w_{2i+1} \cdots w_{2n}$. Then $|u| = 2i - 1$ and the middle symbol of u is $w_i = 0$, so $u \in L(A)$. Also $|v| = 2(n - i) + 1$ and the middle symbol of v is $w_{n+i} = 1$, so $v \in L(B)$. Hence $w = uv$ and $w \in L(G)$ (via $S \to AB$). This proves $D \subseteq L(G)$.

Let $w \in L(G)$. Then w is derived either by $S \to AB$ or by $S \to BA$. We show that if the former is the case then $w \in D$ (the arguments for the latter case is symmetrical). If the first production is $S \to AB$, then $w = uv$ with $u \in L(A)$ and $v \in L(B)$. Write $|u| = 2k + 1$ and $|v| = 2m + 1$ for some $k, m \geq 0$. Thus $|w| = 2(k + m + 1)$; put $n := k + m + 1$, so $|w| = 2n$. Consider the index $i := k + 1$ (note $1 \leq i \leq n$). The i-th symbol of w is the middle symbol of u, hence equals 0. The symbol of w at position $n + i$ is the middle symbol of v, hence equals 1. Therefore $w_i \neq w_{n+i}$, which shows the first half and the second half of w differ; consequently $w \in D$. This proves $L(G) \subseteq D$. Therefore, $D = L(G)$ $\square$

(b) Let $M_A = (Q_A, \Sigma, \delta_A, q_A^0, F_A)$ and $M_B = (Q_B, \Sigma, \delta_B, q_B^0, F_B)$ be DFAs for A and B, respectively (assume $Q_A \cap Q_B = \varnothing$). We build a PDA $P = (Q, \Sigma, \Gamma, \Delta, s, F)$ recognizing $A \diamond B$ that uses a bottom-of-stack marker `$` and a counter symbol `#` as its stack symbols to partially keep track of past information. Define $Q = \{s, \text{acc}\} \cup Q_A \cup Q_B$, $\Gamma = \{\texttt{\$}, \texttt{\#}\}$, the start state to be s, and the set of accepting states $F = \{\text{acc}\}$.

The idea is to split a run of P into roughly two phases: Phase A simulates M_A on the first block x while pushing one `#` per input symbol; we may *only* switch to Phase B from an M_A-accepting state. Phase B simulates M_B on the remainder y while popping one `#` per input symbol. We accept exactly when all `#`s are popped and the top-of-stack is the bottom marker `$` (which we push in the very beginning) while M_B is accepting; at that moment we pop `$` and move to acc. This enforces $|x| = |y|$ and $x \in A, y \in B$. Refer to the next paragraph for a concrete description of the transition function we want to achieve.

Recall from the lectures that we write transitions in the form "$a, b \to c$" meaning: read $a \in \Sigma_\varepsilon$, see top $b \in \Gamma_\varepsilon$, replace it by $c \in \Gamma_\varepsilon$.

(a) Initialize bottom-of-stack and enter Phase A by moving onto the start state of M_A;

$$s \xrightarrow{\varepsilon,\ \varepsilon \to \texttt{\$}} q_A^0.$$

(b) Phase A (simulate M_A and count $|x|$): read a as the next input symbol, move to the new state in the same way M_A would following δ_A, and push a counter symbol to account for the new symbol read as part of x; for all $p \in Q_A$ and $a \in \Sigma$,

$$p \xrightarrow{a,\ \varepsilon \to \#} \delta_A(p, a).$$

(c) Guess the split point (only from A-accepting states): if P is currently sitting on a accepting state of M_A, then we let it guess that what it has read so far from the input string might be what it should assign to x and let it move to the starting state of M_B so that it can start simulating the Phase B (note: no stack change at the switch); for all $p \in F_A$,

$$p \xrightarrow{\varepsilon,\ \varepsilon \to \varepsilon} q_B^0.$$

(d) Phase B (simulate M_B and match $|y|$ to $|x|$): read a as the next input symbol, move to the new state in the same way M_B would following δ_B, and pop a counter symbol to match the new symbol read as part of y to one of the symbols P already, supposedly has read; for all $q \in Q_B$ and $a \in \Sigma$,

$$q \xrightarrow{a,\ \# \to \varepsilon} \delta_B(q, a).$$

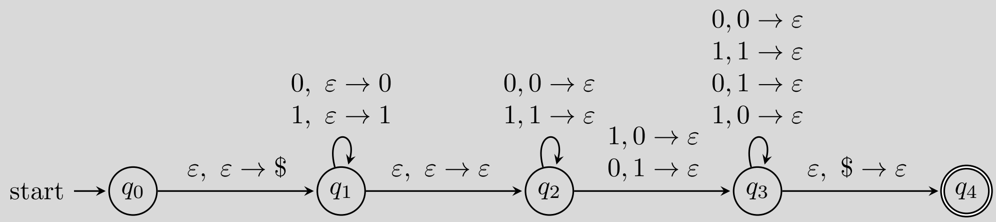
(e) Accept exactly when the counter is exhausted and B accepts: for all $q \in F_B$,

$$q \xrightarrow{\varepsilon,\ \$ \to \varepsilon} \text{acc}.$$

A wrong solution worth mentioning. Some students proposed the following PDA P for D. Define $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ with

$$Q = \{q_0, q_1, q_2, q_3, q_4\}, \quad \Sigma = \{0, 1\}, \quad \Gamma = \{0, 1, \$\}, \quad q_0 \text{ start}, \quad F = \{q_4\}.$$

The transition function δ is given by the following diagram:



The intuitive explanation, much like the one from Subproblem (b), of the PDA is supposed to be as follows: The PDA first pushes a bottom marker \$ (state $q_0 \to q_1$), then *guesses* a split after some prefix x by nondeterministically moving from q_1 to q_2. While reading x in q_1, it mirrors x onto the stack (pushes each input bit). In q_2 it compares the next block y against the stack: matching bits pop (so long as no difference is seen). At the *first* mismatch it moves to q_3, thereby certifying $x \neq y$. State q_3 then finishes reading the rest of the input while popping one stack symbol per input symbol (regardless of equality), guaranteeing $|x| = |y|$. When exactly the bottom marker \$ remains, the PDA pops it and accepts in q_4. If no mismatch ever occurs, the machine never reaches q_3 and thus rejects. ◇

A careful reader may have already noticed a flaw in the explanation and realized that PDA P does not recognize D. A certificate to show this would be the string 0110: this string is not recognized by P but is in D. The reason this approach does not work is because at q_2 and onward when P tries to compare y to x and looks into the stack, it looks at x in reverse. We did not have to worry about this issue in Subproblem (b) because we only needed to compare $|x|$ to $|y|$, not x to y.

There is, however, a PDA, mechanisms of which can be explained intuitively (of course, a PDA recognizing D exists—trivially because of the CFG-PDA equivalence). Borrowing notations from the proof of Subproblem (a) above, we may say that such a PDA tries to non-deterministically guess the index i which gives the mismatch $w_i \neq w_{i+n}$, marks what w_i was (possible because Σ is finite), and tries to match the count of symbols (basically in the same way as the PDA did in Subproblem (b)) in $w_i w_{i+1} \cdots w_{i+n-1}$ to the rest of the symbols in w to certify the mismatch happens exactly n symbols apart. The details are left as an exercise.

□

5. Recall the definition of homomorphisms in HW2 Q6. We say that a context-free grammar is *linear* if in every rule $A \to w$, the string w contains at most one variable. A context-free language is *linear* if it is generated by a linear context-free grammar.

Let Σ_1, Σ_2 and Γ be alphabets, and for $i \in \{1, 2\}$, let $h_i : (\Sigma_i)^* \to (\Gamma)^*$ be a homomorphism. Prove the following; i.e., provide a linear context-free grammar that generates each language, and prove that the grammar generates the desired language.

(a) The language $\{xy^R : h_1(x) = h_2(y)\}$ is linear context-free.

(b) The language $\{xy^R : h_1(x) \neq h_2(y)\}$ is linear context-free.

Solution. Let $L_1 = \{xy^R : h_1(x) = h_2(y)\}$ and $L_2 = \{xy^R : h_1(x) \neq h_2(y)\}$.

(a) We first let $k = \max\{|h_1(a)| : a \in \Sigma_1\} + \max\{|h_2(b)| : b \in \Sigma_2\}$. Consider $G_1 = (V_1, \Sigma_1 \cup \Sigma_2, R_1, S)$, where $V_1 = \{S_u : u \in \Gamma^* \text{ and } |u| \leq k\}$, $S = S_\epsilon$, and R_1 consists of the following rules.

$$\begin{aligned} &S_\epsilon \to \epsilon && \\ &S_v \to aS_w && \text{for } a \in \Sigma_1 \text{ and } w = v \cdot h_1(a) \\ &S_w \to S_v b && \text{for } b \in \Sigma_2 \text{ and } w = h_2(b) \cdot v \end{aligned}$$

The grammar G_1 is clearly linear. We show that for all $S_u \in V$, if $S_u \Rightarrow^* s$, then s is of the form xy^R such that $u \cdot h_1(x) = h_2(y)$. Then it follows that if $S_\epsilon \Rightarrow^* s$, then $s \in L_1$. We prove by induction on the length n of derivation. If $n \leq 1$, then it is of the form $S_\epsilon \Rightarrow \epsilon$, and the statement holds. Suppose now that the derivation has length $n + 1$. The derivation can either start with the rule $S_v \to aS_w$ or with the rule $S_w \to S_v b$.

- Suppose that the derivation starts with the rule $S_v \to aS_w$. Then by the induction hypothesis, every string that can be derived from S_w is of the form xy^R such that $w \cdot h_1(x) = h_2(y)$. Then we have $S_v \Rightarrow^* axy^R$ such that

$$v \cdot h_1(ax) = v \cdot h_1(a) \cdot h_1(x) = w \cdot h_1(x) = h_2(y),$$

as we have $w = v \cdot h_1(a)$.

- Suppose that the derivation starts with the rule $S_w \to S_v b$. Then by the induction hypothesis, every string that can be derived from S_v is of the form xy^R such that $v \cdot h_1(x) = h_2(y)$. Then we have $S_w \Rightarrow^* xy^R b$ such that

$$w \cdot h_1(x) = h_2(b) \cdot v \cdot h_1(x) = h_2(b) \cdot h_2(y) = h_2(by),$$

as we have $w = h_2(b) \cdot v$.

We now show that for $u \in \Gamma^*$ with $|u| \leq k$, every string $s = xy^R$ with $u \cdot h_1(x) = h_2(y)$ can be derived from $S_u \in V$. Then it follows that if $s \in L_1$, then s can be derived from S_ϵ. We prove by induction on the length n of s. If $n = 0$, then s is ϵ, which can be derived by the rule $S_\epsilon \to \epsilon$. Now, suppose that $n \geq 1$. Then neither x nor y can be ϵ, as otherwise, s must be ϵ. We may thus assume that $y^R = (y')^R b$ for some $b \in \Sigma_2$. Then we have

$$u \cdot h_1(x) = h_2(by') = h_2(b) \cdot h_2(y').$$

This means that either u is a prefix of $h_2(b)$ or vise versa.

- Assume that $h_2(b)$ is a prefix of u, i.e., $u = h_2(b) \cdot v$, and $v \cdot h_1(x) = h_2(y')$ for some $v \in \Gamma^*$. There is a rule $S_u \to S_v b$ with $u = h_2(b) \cdot v$. Moreover, we have $|x(y')^R| < |xy^R|$ and $x(y')^R$ is a string with $v \cdot h_1(x) = h_2(y')$. By the induction hypothesis, there is a derivation $S_v \Rightarrow^* x(y')^R$. Thus, we have a derivation

$$S_u \Rightarrow S_v b \Rightarrow^* x(y')^R b = xy^R.$$

- Assume that u is a prefix of $h_2(b)$. Suppose that $x = ax'$ for some $a \in \Sigma_1$. Then we have

$$u \cdot h_1(ax') = u \cdot h_1(a) \cdot h_1(x') = h_2(y).$$

As $|u| \leq |h_2(b)|$, we have $|u \cdot h_1(a)| \leq k$. Thus, there is a rule $S_u \to aS_w$ with $w = u \cdot h_1(a)$. Moreover, we have $|x'y^R| < |xy^R|$, and $x'y^R$ is a string with $w \cdot h_1(x') = h_2(y)$. By the induction hypothesis, there is a derivation $S_w \Rightarrow^* x'y^R$. Thus, we have

$$S_u \Rightarrow aS_w \Rightarrow ax'y^R = xy^R.$$

(b) Observe that $h_1(x) \neq h_2(y)$ if and only if $h_1(x)$ or $h_2(y)$ is a strict prefix of the other, or neither is a prefix of the other. Consider $G_2 = (V_2, \Sigma_1 \cup \Sigma_2, R_2, S)$, where $V_2 = V_1 \cup \{L, R, M\}$, $S = S_\epsilon$, and R_2 consists of the following rules.

$S_v \to aS_w$	for $a \in \Sigma_1$ and $w = v \cdot h_1(a)$	(1)
$S_w \to S_v b$	for $b \in \Sigma_2$ and $w = h_2(b) \cdot v$	(2)
$S_v \to L$	if $v \neq \epsilon$	(3)
$L \to aL \mid \epsilon$	for $a \in \Sigma_1$	(4)
$S_v \to Rb$	for $b \in \Sigma_2$, and v is a strict prefix of $h_2(b)$	(5)
$R \to Rb \mid \epsilon$	for $b \in \Sigma_2$	(6)
$S_v \to Mb$	for $b \in \Sigma_2$, and v is not a prefix of $h_2(b)$ or vise versa	(7)
$M \to aM \mid Mb \mid \epsilon$	for $a \in \Sigma_1$ and $b \in \Sigma_2$	(8)

Note that the rules (1) and (2) are the same as in (a). The rule $S_\epsilon \to \epsilon$ in (a) is replaced by the rules (3) - (8). The rules (3) and (4) ensure that $h_2(y)$ is a strict prefix of $h_1(x)$. The rules (5) and (6) ensure that $h_1(x)$ is a strict prefix of $h_2(y)$. The rules (7) and (8) ensure that neither $h_1(x)$ or $h_2(y)$ is a prefix of the other. We can prove that G_2 generates L_2 as in (a) and by using the above observation.

□