Polynomial Time Learner for Inferring Subclasses of Internal Contextual Grammars with Local Maximum Selectors

Abhisek Midya[1(✉)], D.G. Thomas[2], Saleem Malik[3], and Alok Kumar Pani[4]

[1] Computer Science and Engineering, Icfai Tech School, Hyderabad 501203, India
abhisekmidyacse@gmail.com
[2] Department of Mathematics, Madras Christian College, Chennai 600059, India
dgthomasmcc@yahoo.com
[3] Computer Science and Engineering, Alliance University, Bangalore 562106, India
baronsaleem@gmail.com
[4] Computer Science and Engineering, Faculty of Engineering, Christ University, Bangalore 560074, India
alok.kumar@christuniversity.in

Abstract. Natural languages contain regular, context-free, and context-sensitive syntactic constructions, yet none of these classes of formal languages can be identified in the limit from positive examples. Mildly context-sensitive languages are capable to represent some context-sensitive constructions such as multiple agreement, crossed agreement, and duplication. These languages are important for natural language applications due to their expressiveness, and the fact that they are not fully context-sensitive. In this paper, we present a polynomial-time algorithm for inferring subclasses of internal contextual languages using positive examples only, namely strictly and k-uniform internal contextual languages with local maximum selectors which can contain mildly context-sensitive languages.

Keywords: Internal contextual grammar with local maximum selectors · Identification in the limit from positive data

1 Introduction

In theoretical computer science, formal language theory is one of the fundamental areas. This study has its origin in Chomskian grammars. Contextual grammars which are different from Chomskian grammars, have been studied in [9,13,14,17] by formal language theorists, as they provide novel insight into a number of issues central to formal language theory. In a total contextual grammar, a context is adjoined depending on the whole current string. Two special cases of total contextual grammars, namely internal and external are very natural and have been extensively investigated. (External) Contextual grammars are introduced by Marcus in 1969 [9] with a linguistic motivation in mind. An internal contextual

D.V. Hung and D. Kapur (Eds.): ICTAC 2017, LNCS 10580, pp. 174–191, 2017.
DOI: 10.1007/978-3-319-67729-3_11

grammar generates a language starting from a finite set of strings (the base) and iteratively adjoining to its contexts outside the current string. In other families of contextual grammars, such as internal contextual grammars [9], the contexts are adjoined inside the current string.

According to [12], it is known that many classes of formal languages, such as regular and context-free, cannot be learned from positive data only. Now it natural to look for subclasses of these languages which can be identified in the limit from positive data only.

In this paper, we present a polynomial-time algorithm for learning Strictly and k-uniform internal contextual languages with local maximum selector $(SLICG_M, k-UICG_{LM})$ from positive data. Using these two languages, mildly context sensitive languages can be generated. That is, they can express the context-sensitive syntactic constructions that are most prevalent in natural languages, such as multiple agreement, crossed agreement, and duplication [3].

Currently, there is an algorithm known for inferring the subclasses of the class of internal contextual grammars with finite selector set [10]. Also, polynomial time inferring algorithm is available for very attractive subclasses of the class of external contextual grammars [8].

The paper is organized as follows. Section 2 describes the basic classes of contextual grammars in more detail. Section 3 describes the newly defined subclasses. In Sect. 4, we discuss the generative power of the subclasses. Sections 5, 6 and 7 present the pseudocode and discuss the complete algorithm in detail along with the correctness. Section 8 discusses the characteristic sample of the algorithm. Running time complexity of the algorithm has been described in Sect. 9. In Sect. 10, we present a complex example for better understanding of the algorithm.

2 Basic Classes of Contextual Languages

This section recalls the definition of the basic classes of contextual languages. [11] For an alphabet Σ, we denote by Σ^* the free monoid generated by Σ, by λ its identity, and $\Sigma^+ = \Sigma^* - \{\lambda\}$.

Definition 1. *A Contextual grammar is a construct* $G = (\Sigma, A, (sel_1, C_1), (sel_2, C_2), \ldots, (sel_k, C_k))$, *for some* $k \geq 1$, *where* Σ *is an alphabet,* $A \subset \Sigma^*$ *is a finite set, called the axiom set,* $sel_i \subseteq \Sigma^*, 1 \leq i \leq k$, *are the sets of selectors, and* $C_i \subset \Sigma^* \times \Sigma^*$ *where* $1 \leq i \leq k$, *and* C_i *is a finite set of contexts. There are two basic modes of derivation as follows. For two words* $x, y \in \Sigma^*$, *we have the internal mode of derivation:*

$x \Longrightarrow_{in} y$ *iff* $x = x_1x_2x_3, y = x_1ux_2vx_3, x_2 \in sel_i, (u, v) \in C_i$, *for some* $1 \leq i \leq k$. *The external mode of derivation:*
$x \Longrightarrow_{ex} y$ *iff* $y = uxv, x \in sel_i, (u, v) \in C_i$, *for some* $1 \leq i \leq k$. *The language generated by G with respect to each of the two modes of derivation is:* $L_\alpha(G) = \{w \in \Sigma^* \mid x \in A, x \Longrightarrow_\alpha^* w\}$, *for* $\alpha \in \{in, ex\}$, *where* $\Longrightarrow_\alpha^*$ *denotes the reflexive - transitive closure of* $\Longrightarrow_\alpha$.

A contextual grammar with internal (external) mode of derivation is called an internal (external) contextual grammar. The corresponding languages are called an internal contextual languages and external contextual languages.

If $sel_1, sel_2, \ldots, sel_k$ are languages in the family of regular languages REG, then G is said to be with REG choice. The family of languages generated by contextual grammars with REG choice in the mode α of derivation is denoted by $L\alpha(REG)$.

Now consider the local maximum selector in internal local mode of derivation. One natural restriction has been imposed on the use of selectors as seen in [15]. In fact, there is a need for some length conditions on the selector to be used, such as minimality or maximality. It implies that we can put the restriction that any time when a context is adjoined around a selector, no factor of the selector (minimal case) can be used as a selector, or no word containing the current selector as a factor can be used as a selector (maximal case). This restriction can be imposed with respect to the specified pair of selectors or to the whole grammar. Now we discuss some details about the maximal case only because using maximal use of selectors, we will be able to generate mildly context-sensitive family of languages which is one the most important component to characterize natural languages.

Definition 2. *Given a contextual grammar $G = (\Sigma, A, (sel_1, C_1), (sel_2, C_2), \ldots, (sel_k, C_k))$, we define, for two words $x, y \in \Sigma^*$, the local maximal mode of derivation in G is defined as follows: $x \Longrightarrow_{lm} y$ iff $x = x_1x_2x_3, y = x_1ux_2vx_3$, for $x_2 \in sel_i, (u, v) \in C_i, i \leq i \leq k$ and for no $x'_1, x'_2, x'_3 \in sel_i, x = x'_1x'_2x'_3, x'_2 \in sel_i, x_2$ a factor of x'_2. Here lm denotes the local maximal mode.*

Example: Consider the following contextual grammar

– $G = (\{a, b, c\}, \{abc\}, (b^+, \{(a, bc)\}))$

Now we show one sample derivation - here [] denotes the contexts and underlined string is the selector.

– $a\underline{b}c \Longrightarrow_G a[a]\underline{b}[bc]c$
– $aa\underline{bb}cc \Longrightarrow_G aa[a]\underline{bb}[bc]cc$
– $aaa\underline{bbb}ccc \Longrightarrow_G aaa[a]\underline{bbb}[bc]ccc$
– The language generated by G is $L_{lm}(G) = \{a^nb^nc^n \mid n \geq 1\}$.

3 Subclasses of the Class of Internal Contextual Grammars with Local Maximum Selectors

In this paper our learning paradigm is identification in the limit which is defined as follows:

Definition 3 [12]**.** *Method M identifies language L in the limit if, after a finite number of examples, M makes a correct guess and does not alter its guess thereafter. A class of languages is identifiable in the limit if there is a method M such that given any language of the class and given any admissible example sequence for this language, M identifies the language in the limit.*

Here our main focus is on designing an identification algorithm to infer internal contextual languages, but according to Gold model [12], no superfinite class of languages is inferable from positive data only. A class of languages which consists of all finite languages but atleast one infinite language, is called super finite class of languages. From [10], we have the following result.

Theorem 1 [10]. *The class of internal contextual languages (ICL), is not inferable from positive data only.*

As we know that the class ICL is not inferable from positive data only, it is natural to look for subclasses of these languages which can be identified in the limit from positive data only. We now define strictly internal contextual grammar with local maximum selectors ($SICG_{LM}$) and k-uniform internal contextual grammar with local maximum selectors ($k - UICG_{LM}$).

Definition 4. *A strictly internal contextual grammar with local maximum selectors ($SICG_{LM}$) is an internal contextual grammar* $G = (\Sigma, A, (sel_1, C_1), (sel_2, C_2), \ldots, (sel_k, C_k))$, *for some* $k \geq 1$, *where*

- Σ *is the alphabet.*
- $A \subset \Sigma^*$ *is a finite set, called axiom set.*
- $sel_i \subseteq \Sigma^*, 1 \leq i \leq k$, *are the sets of selectors.*
- $C_i \subset \Sigma^* \times \Sigma^*$, *are sets of contexts.*

with the following restrictions,

- *If the rule is* (sel_i, C_i) *where* $C_i = \{(u_i, v_i)\}$ *then* $first(u_i) \neq first(v_i)$ *where* $first(u)$ *denotes the first alphabet of* u.
- *for each selector, there exists exactly one context* (u, v).

The language generated by $SICG_{LM}G$ *is given by* $L_{slm}(G) = \{w \in \Sigma^* \mid x \Longrightarrow_{slm}^{*} w, x \in A\}$ *where* $\Longrightarrow_{slm}$ *denotes the one step derivation in strictly local maximal mode. Now, SLM denotes the family of languages generated by* $SICG_{LM}$.

Now we present two examples of $SICG_{LM}, G_1, G_2$:

- $G_1 = (\{a, b, c\}, \{abc\}, (b^+, \{(a, bc)\}))$ where $first(u) = a \neq first(v) = b, L(G_1) = \{a^n b^n c^n \mid n \geq 1\}$. For better understanding, see the derivation example of Definition 2.
- $G_2 = (\{a, b, c, d\}, \{abcd\}, (ab^+c, \{(a, c)\}), (bc^+d, \{(b, d)\}))$ where $first(u) = a \neq first(v) = b$ and $first(u) = b \neq first(v) = d, L(G_2) = \{a^n b^m c^n d^m \mid n \geq 1\}$.

Definition 5. *A k - uniform internal contextual grammar with local maximum selectors* $(k - UICG_{LM})$ *is an internal contextual grammar* $G = (\Sigma, A, (sel_1, C_1), (sel_2, C_2), \ldots, (sel_k, C_k))$, *for some* $k \geq 1$, *where*

- Σ *is the alphabet.*
- A *is the finite subset of* Σ^*, *called axiom set.*

- $sel_i \subseteq \Sigma^*, 1 \leq i \leq k$, *are the sets of selectors.*
- $C_i \subset \Sigma^* \times \Sigma^*$, *are sets of contexts.*
  With the following restrictions, if the rule is (sel_i, C_i) *where* $C_i = \{(u_i, v_i)\}$ *then* $|u| = |v| = k$.
- *The language generated by a* $k - UICG_{LM}$ G *is given by* $L_{klm}(G) = \{w \in \Sigma^* \mid x \Longrightarrow_{klm}^* w, x \in A\}$ *where* $\Longrightarrow_{klm}$ *denotes the one step derivation in k-local maximal mode. Now,* KLM *denotes the family of languages generated by* $k - UICG_{LM}$.

Now we present a $k - UICG_{LM}$ $G_3 = (\{a, b, c\}, \{c\}, (\{c\}\{a, b\}^*, \{(a, a), (b, b)\}))$, $L_{klm}(G_3) = \{wcw \mid w \in \{a, b\}^*\}$.

4 Power of the Subclasses

In this section we discuss the generative power of these subclasses. We know that several natural languages are not context-free and these languages are consisting of non-context-free properties. Thus, in order to obtain formal grammars focusing to model natural languages, we have to look for classes of grammars that are able to generate non-context-free languages. On the other hand they should not be too powerful, that means they should not generate languages without any linguistic relevance. So, the idea of keeping the generative power under control has lead to the notion of *mildly context-sensitive* family of languages. The properties of such families are the following [11]:

1. It contains all three basic non-context-free constructions in, that is,
   - *multiple agreements:* $L_1 = \{a^n b^n c^n \mid n \geq 1\}$
   - *crossed agreements:* $L_2 = \{a^n b^m c^n d^m \mid n, m \geq 1\}$
   - *duplication:* $L_3 = \{wcw \mid w \in (a + b)^*\}$.
2. All the languages in the family, are *polynomial time* parsable.
3. It contains *semilinear* languages.

Here, our defined subclasses can generate three basic non-context-free constructions.

Theorem 2

(i) $L_1, L_2 \in SLM$ *(See examples of Definition 4).*
(ii) $L_3 \in KLM$ *(See example of Definition 5).*

Lemma 1. $KLM - SLM \neq \phi$.

Proof. From Theorem 2 we know that $L_3 \in KLM$. The appropriate grammar to generate L_3 is $G_3 = (\{a, b, c\}, \{c\}, (\{c\}\{(a + b)^*\}, \{(a, a), (b, b)\}))$ where $k = 1$. But $L_3 \notin SLM$, as we know from Definition 3 that if the rule is (sel_i, C_i) where $C_i = \{(u_i, v_i)\}$ then $first(u_i) \neq first(v_i)$. Here it needs to be always $first(u_i) = a = first(v_i)$ or $first(u_i) = b = first(v_i)$. □

Lemma 2. $SLM - KLM \neq \phi$.

Proof. From Theorem 2, we can conclude that $L_1, L_2 \in SLM$. The appropriate grammar to generate L_1 and L_2 are respectively $G_1 = (\{a, b, c\}, \{abc\}, (b^+, \{(a, bc)\}))$ where $|u| = |a| = 1$ and $|v| = |bc| = 2$ and $G_2 = (\{a, b, c, d\}, \{abcd\}, (ab^+c, \{(a, bc)\}), (bc^+d, \{(b, d)\}))$ where for selector ab^+c the required contexts are always $|u| = |a| = 1$ and $|v| = |bc| = 2$. So it can be understood easily that $L_1, L_2 \notin KLM$. □

Lemma 3. $SLM \cap KLM \neq \phi$.

Proof. $L_5 = \{a^n cb^n \mid n \geq 0\}, L_5 \in SLM \cap KLM$. The appropriate grammar to generate L_5 is $G = (\{a, b, c\}, \{c\}, (c, \{(a, b)\}))$ and it satisfies Definitions 4 and 5. □

Theorem 3. *SLM is incomparable with KLM and but they are not disjoint.*

Proof. We can conclude this fact from Lemmas 1, 2 and 3. □

5 Identification of Subclasses of Internal Contextual Languages with Local Maximum Selectors and Correctness

In this section, we propose an identification algorithm IA to infer $SICG_{LM}$ from positive examples only. We recall the notion of an insertion rule. The insertion operation is first considered by Haussler in [6] and based on the operation, insertion systems are introduced by Kari in [7]. Informally, if a string α is inserted between two parts w_1 and w_2 of a string w_1w_2 to get $w_1\alpha w_2$, we call the operation as insertion.

Our identification algorithm IA takes finite sequences of positive examples i_{t_j} in the different time interval t_j where $j \geq 1$. Our goal is to find out $SICG_{LM}$, such that $IPS \subseteq L(G)$ where IPS is the input set. The algorithm works in the following way.

- After receiving the first set as an input, based on the size of each input, firstly the algorithm determines the axiom.
- Then it defines insertion rules in order to find out the contexts and selectors from input example.
- After that, insertion rules are converted into 1-sided[1] contextual rules.
- Next it updates with new contextual rules if the next input cannot be generated by the existing contextual rules, that is called the correction phase. All the guessing will be done in a flexible way in the sense that the correction can be done at every instance.
- Then it will convert 1-sided contextual rule into 2-sided contextual rule to take care of over generalization, that could be the temporary guess g_j at particular time interval t_j, about the unknown grammar.
- Finally we will take care of maximal use of selectors.

[1] In an 1-sided contextual rule either left context is λ or right context is λ.

Lemma 4. *Let $g_{t_1}, g_{t_2}, \ldots, g_{t_i}$ be the sequences of guesses (grammar) about the unknown grammar produced by identification algorithm IA at different time interval $t_1, t_2, \ldots, t_i$ based on different information, $i_{t_1}, i_{t_2}, \ldots, i_{t_i}$ such that $g_f = g_{f+1}$.*

Proof. The behavior of the algorithm, in particular, there is an upper bound(in terms of the size the current input set) to make the guess g_i about the unknown grammar where $L(g_{i-1}) \subset L(g_i)$. Thus, there exist a $f \geq 1$ such that $g_f = g_{f+1}$ where $L(g_{f-1}) \subset L(g_f)$. So, we conclude this lemma. □

From this, we have the following result.

Theorem 4. *SLM is identifiable in the limit from positive examples only.*

6 Pseudocode of Our Algorithm

In this section we present the pseudocode of our algorithm IA and also in further subsections we explain that in detail.

$axiom \leftarrow FIND-SMALLEST(IPS)$
$inser \leftarrow GENERATE-INSR(axiom, s_i)$
$1-Sided-Contextual-Rule \leftarrow \{\}$
$1-Sided-Correct-Rule \leftarrow \{\}$
$2-Sided-Correct-Rule \leftarrow \{\}$
$Table \leftarrow \sqcap$
$1-Sided-Contextual-Rule.push[CONVERT-into-CONTEXTUAL-RULE(inser)]$
$IPS \leftarrow REMOVE(IPS, s_i)$
for $(1-Sided-Contextual-Rule_i \in \{1-Sided-Contextual-Rule\})$ **do**
    for $(s_i \in IPS)$ **do**
        $S \leftarrow CHECK-CONTEXTUAL-RULE(1-Sided-Contextual-Rule_i, s_i)$
        if $S = 1$ **then**
            $1-Sided-Correct-Rule.push[1-Sided-Contextual-Rule_i]$
        if $S = 0$ **then**
            $1-Sided-Correct-Rule.push[CORRECTION-CONTEXTUAL-RULE(1-Sided-Contextual-Rule_i, s_i)]$
for $(1-Sided-Correct-Rule_i \in \{1-Sided-Correct-Rule\})$ **do**
    for $(s_i \in IPS)$ **do**
        $Table.insert[FIND-NOF-APP-of-EACHRULE-in-EACHMEMBER(1-Sided-Correct-Rule_i, s_i)]$
if $TableRow_i = TableRow_j$ **then**
    $2-Sided-Correct-Rule.push[MERGE(1-Sided-Correct-Rule_i, 1-Sided-Correct-Rule_j)]$
$LOC-MAX-SEL_i \leftarrow LMS(sel_i, u_i, v_i)$

6.1 Finding Axiom - Pseudocode-Step: 1

axiom ← FIND − SMALLEST(IPS):

- **case 1:** It finds the smallest string from the current IPS. The smallest string will be considered as an axiom.
- **case 2:** If two strings are given with same length then both of them will be there in the axiom set A.
- **case 3:** At any point of time a string can be given as an input which is smaller than some members of the existing axiom set. In such cases, if the strings existing in the axiom set can be generated from this new smaller string, then this new smaller string will replace them.
- **case 4:** If no member of the existing axiom set can be generated from the new smaller string then the new smaller string will be added to the axiom set as a new member of the axiom set.

Let IPS be the set of input strings. $IPS = \{s_1, s_2, \ldots, s_k\}$ where $s_j = s_{j_1} s_{j_2} \ldots s_{j_r}, 1 \leq j \leq k, 1 \geq r$. (i.e., s_j is of length r). Then the axiom will be $Min(IPS)$ where $Min(A)$ denotes the minimum size member of set A.

6.2 Defining Insertion Rule and Converting It into Contextual Rule - Pseudocode-Steps: 2, 7, 8

- **insr ← GENERATE − INS(axiom, $\mathbf{s_i}$):** It generates the insertion rule from axiom and any member (s_i) of input set IPS. The output of the function will be stored in $insr$ as an insertion rule.
- **1 − Sided − Contextual − Rule.push[CONVERT − into − CONTEXT −UAL − RULE(inser)]:** It converts $insr$ into $1-Sided-Contextual-Rule$ and push that into $1 - Sided - Contextual - Rule$ set.
- **IPS ← REMOVE(IPS, $\mathbf{IP_i}$):** It removes the current input member IP_i from IPS.

We now shortly describe about the intuitive idea of the parts 1–4. We try to identify the selectors from the axiom and contexts from examining input. If the format of the insertion rule is uxv where $u, x, v \in \Sigma^+$ are left context, inserted portion, and right context respectively.

- Let the axiom be $s_j^a = s_{j_1}^a s_{j_2}^a s_{j_3}^a \ldots s_{j_n}^a$ and the examining (scanning) string be $s_j^e = s_{j_1}^e s_{j_2}^e \ldots s_{j_r}^e$ where r = length of the examining string. Now from the axiom we can have the following consideration. In the following four parts, if a string x is a substring of y, then it is denoted by $x \in sub(y)$.
- **Part 1:** let the initial rule be $(u, x, v)_{ins}$ where $u = s_{j_1}^a, v = s_{j_2}^a s_{j_3}^a \ldots s_{j_n}^a$, check whether any $|x| \leq r$ exists with $uxv \in sub(s_j^e)$ or not. If yes then fix that x (i.e., and go to part 3. Else, go to part 2.
- **Part 2:** Remove the last alphabet of the right context v and the rule becomes $(u, x, v)_{ins}$ where $u = s_{j_1}^a, v = s_{j_2}^a s_{j_3}^a \ldots s_{j_{n-1}}^a$, Check whether any $|x| \leq r$ exists with $uxv \in sub(s_j^e)$ or not, if yes, go to part 3. Else, go to (recursively) part 2 until the rule becomes of the form $(u, x, v) \mid u = s_{j_1}^a, v = s_{j_2}^a$. Then go to part 4.

- **Part 3: Conversion into Contextual Rule: 7.** After getting correct insertion rules (which necessarily satisfy $uxv \in sub(s_j^e)$), they are converted into 1-sided contextual rules as follows: $(u, x, v)_{ins} \longrightarrow (sel, (u, v))_{icg}$ where $sel_{icg} = u_{ins}, v_{icg} = x, u_{icg} = \lambda$ and the omitted right context v_{ins} will be treated as the left context u_{ins} for the next insertion rule. Now, we remove $(ux)_{ins}$ as a substring from the examining string and only u_{ins} from the axiom. Once we get a selector and associated context with it, we have the following conditions for each insertion rule.
- **Condition 1:** If $(|u| + |x| + |v|)_{ins} = |E|$ where $|E|$ denotes the length of examining string, it implies that only one rule has been applied and we have obtained that already.
- **Condition 2:** If $(|u|+|v|)^{ins} \leq |s_j^a|$ where $|s_j^a|$ denotes the length of the axiom, then we remove u_{ins} from axiom s_j^a, and obtain a new temporary axiom. Also consider $v_{ins} = u_{ins}$ for the next insertion rule. Next, it removes $(ux)_{ins}$ as a substring from s_j^e and obtain a new temporary input. Here after we continue our procedure with this temporary axiom and temporary examining input in the same way.
- **Condition 3:** If $(|u| + |x| + |v|)_{ins} \leq |E|$ but $(|u| + |v|)_{ins} = |s_j^a|$, it implies that some part is still left to scan and that is left context u_{icg} of the first selector sel_{icg}^{first} or right context v_{icg} of the last selector sel_{icg}^{last}, then we will include them as a new rule.
  $(sel_{new}, \{u_{mew}, v_{new}\})_{new}$ where $u_{new} = u_{icg}, v_{new} = \lambda, sel_{new} = sel_{icg}^{first}$, in another case, $v_{new} = v_{icg}, sel_{new} = sel_{icg}^{last}$. For these rules, we will never go for correction.

- **Part 4:** If $u_{ins} = s_{j\,1}^a$ and $v_{ins} = s_{j\,2}^a$, this time we consider $u_{ins} = s_{j\,1}^a s_{j\,2}^a$. Rest of the axiom part will be considered as right context v_{ins} of the new rule as follows, $(u, x, v)_{ins}$ where $u = s_{j\,1}^a s_{j\,2}^a, v = s_{j\,3}^a \ldots s_{j\,n}^a$ and go to part 1 until $u_{ins} = s_{j\,1}^a s_{j\,2}^a s_{j\,3}^a \ldots s_{j\,n}^a$. In that case, defining insertion rule is not possible. Here our selection of axiom is wrong, so we need to start with different axiom.

In this section, we get the selectors from axiom and contexts from examining input. Later on for new input, we may need to change our it for wrong guess (next section).

6.3 Making Correction and Updating Rules - Pseudocode-Steps: 9–15

- $\mathbf{S \leftarrow CHECK - CONTEXTUAL - RULE(1 - Sided - Contextual - Rule_i, s_i)}$**:** It checks the correctness of $1 - Sided - Contextual - Rule_i$ for another input.
- If S is true then the correct $1 - Sided - Contextual - Rule_i$ will be pushed onto set $\{1 - Sided - Correct - Rule_i\}$ and continue the process for the next input.

- $\mathbf{CORRECTION - CONTEXTUAL - RULE(1 - Sided - Contextual - Rule_i, s_i)}$**:** Otherwise it goes for correction.

Below we have discussed that if the new examining string is not derivable with the existing set of contextual rules, then we need to go for correction and updating with new rules.

Let the rule be $R_i : (sel_i, (u_i, v_i))_{icg}$ where $u_i = \lambda$. Examining string $s^e_j = s^e_{j1}s^e_{j2}\ldots s^e_{jr}$. We can represent the examining as $X\ sel_i s^e_{jy+1}s^e_{jy+2}\ldots s^e_{jy'}\ sel_{i+1}\ Z$ where $X, Z \in \Sigma^*$ and the remaining parts of the string. The examining string is presented in this form $X\ sel_i s^e_{jy+1}s^e_{jy+2}\ldots s^e_{jy'}\ sel_{i+1}\ Z$ because we make the correction of rule R_i using rule R_{i+1}, so it is needed to introduce the sel_i and sel_{i+1} both.

Proposition 1. *In case of correction, we deal with only 1-sided contextual rules where left context is always empty. (see condition 3 of Subsect. 6.2)*

If $sel_i = s^e_{jl}s^e_{jl+1}\ldots s^e_{jy}$. If selector sel_i, sel_{i+1} are not present in s^e_j then new insertion rule has to be defined again to find out the correct selectors and go to Sect. 6.2. If defining insertion rule is not possible even after this step, then it indicates that the chosen axiom is wrong. In that case, we will choose some other axiom, if available. If no other axiom is available then we add the examining string into the axiom set as a new member of axiom set (recall that we have positive examples only).

If $v_i \neq s^e_{jy+1}s^e_{jy+2}\ldots s^e_{jy'}$, then correction and updating is required. Let v_i be $V_1V_2\ldots V_w$ and $s^e_{jy+1}s^e_{jy+2}\ldots s^e_{jy'}$ be $Q_1Q_2\ldots Q_z$ for convenience sake.

To apply the rule properly the following condition is required, $V_1V_2\ldots V_w = Q_1Q_2\ldots Q_z$ where $w = z$.

Here we are making an analysis to find out the partially equal part (prefix/suffix) of $V_1V_2\ldots V_w$ and $Q_1Q_2\ldots Q_z$.

We have shown that the correction part for one rule, in the same way the correction can be done for other rules.

Theorem 5. *If the analysis starts with equality such that $Q_1 = V_1, Q_2 = V_2, \ldots, Q_f = V_s$, and $Q_{f+1} \neq V_{s+1}$ or $f = z$ or $s = w$, then we can have four different type of errors which are stated in terms of following lemmas. (Finding common prefix part).*

Lemma 5. *If ($f = z$ and $s = w$) then it implies that matching is correct, so no need to make any correction for this rule and the rule is correct.*

Lemma 6. *If ($f = z$ and $s < w$) then we infer two new rules.*

Proof.
- $Rule_{i'} : (sel_{i'}, C_{i'})$ where $C_i = \{(u_{i'}, v_{i'})\}, v_{i'} = V_1V_2\ldots V_s = Q_1Q_2\ldots Q_z, u_{i'} = \lambda, sel_{i'} = sel_i$.
- $Rule_{(i+1)'} : (sel_{(i+1)'}, C_{(i+1)'})$ where $C_{(i+1)'} = \{(u_{(i+1)'}, v_{(i+1)'})\}, u_{(i+1)'} = V_{s+1}V_{s+2}\ldots V_w, v_{(i+1)'} = \lambda, sel_{(i+1)'} = sel_{(i+1)}$. □

Lemma 7. *If ($f < z$ and $s = w$) then we infer two new rules.*

Proof. – $Rule_{i'} : (sel_{i'}, C_{i'})$ where $C_i = \{(u_{i'}, v_{i'})\}, v_{i'} = V_1V_2 \ldots V_w = Q_1Q_2 \ldots Q_f, u_{i'} = \lambda, sel_{i'} = sel_i$.
– $Rule_{(i+1)'} = (sel_{(i+1)'}, C_{(i+1)'})$ where $C_{(i+1)'} = \{(u'_{(i+1)}, v_{(i+1)'})\}, u_{(i+1)'} = Q_{f+1}Q_{f+2} \ldots Q_z, v_{(i+1)'} = \lambda, sel_{(i+1)'} = sel_{(i+1)}$. □

Lemma 8. *If* $(f < z$ *and* $s < w)$ *then we infer three new rules.*

Proof. – $Rule_{i'} : (sel_{i'}, C_{i'})$ where $C_i = \{(u_{i'}, v_{i'})\}, v_{i'} = V_1V_2 \ldots V_s = Q_1Q_2 \ldots Q_f, u_{i'} = \lambda, sel_{i'} = sel_i$.
– $Rule_{(i+1)'} : (sel_{(i+1)'}, C_{(i+1)'})$ where $u_{(i+1)'} = V_{s+1}V_{s+2} \ldots V_w, u_{(i+1)'} = \lambda, sel_{(i+1)'} = sel_{i+1}$.
– $Rule_{(i+2)'} : Rule_{(i+2)'} : (sel_{(i+2)'}, C_{(i+2)'})$ where $u_{(i+2)'} = Q_{f+1}Q_{f+2} \ldots Q_z, v_{(i+2)'} = \lambda, sel_{(i+2)'} = sel_{i+1}$. □

Theorem 6. *If the analysis starts with inequality such that* $Q_1 \neq V_1$*, but* $Q_z = V_w, Q_{z-1} = V_{w-1} \ldots Q_f = V_s$*, and* $Q_{f-1} \neq V_{s-1}$ *then we can have three different type of errors which can be seen in the following lemmas. (Finding common suffix part).*

Lemma 9. *If* $(s = 1, f > 1)$ *then we infer two new rules.*

Proof. – $Rule_{i'} : (sel_{i'}, C_{i'})$ where $C_{i'} = \{(u_{i'}, v_{i'})\}, u_{i'} = V_1V_2 \ldots V_w, v_{i'} = \lambda, sel_{i'} = sel_{i+1}$.
– $Rule_{(i+1)'} : (sel_{(i+1)'}, C_{(i+1)'})$ where $C_{(i+1)'} = (u'_{(i+1)}, v_{(i+1)'})$ where $v_{(i+1)'} = Q_1Q_2 \ldots Q_{f-1}, u_{(i+1)'} = \lambda, sel_{(i+1)'} = sel_i$. □

Lemma 10. *If* $(s > 1)$ *then we infer three new rules.*

Proof. – $Rule_{i'} : (sel'_i, C'_i)$ where $C_{i'} = \{(u_{i'}, v_{i'})\}, u_{i'} = V_sV_{s+1} \ldots V_w, v_{i'} = \lambda, sel_{i'} = sel_{i+1}$.
– $Rule_{(i+1)'} : (sel_{(i+1)'}, C_{(i+1)'})$ where $C_{(i+1)'} = (u_{(i+1)'}, v_{(i+1)'}), v_{(i+1)'} = Q_1Q_2 \ldots Q_{f-1}, u_{(i+1)'} = \lambda, sel_{(i+1)'} = sel_i$.
– $Rule_{(i+2)'} : (sel_{(i+2)'}, C_{(i+2)'})$ where $C_{(i+2)'} = (u_{(i+2)'}, v_{(i+2)'}), u_{(i+2)'} = \lambda, v_{(i+2)'} = V_1V_2 \ldots V_{s-1}, sel_{(i+2)'} = sel_i$. □

Lemma 11. *If* $Q_z \neq R_w$ *then we infer two new rules. (In this case Theorem 6 is not applicable here because common prefix/suffix part is absent).*

Proof. – $Rule_{i'} : (sel'_i, C'_i)$ where $C_i = \{(u_{i'}, v_{i'})\}, v_{i'} = V_1V_2 \ldots V_w, u_{i'} = \lambda, sel_{i'} = sel_i$.
– $Rule_{(i+1)'} : (sel_{(i+1)'}, C_{(i+1)'})$ where $C_{(i+1)'} = (u'_{(i+1)}, v_{(i+1)'} = Q_1Q_2 \ldots Q_z, u_{(i+1)'} = \lambda, sel_{(i+1)'} = sel_i$. □

In this section, we must notice that we have different rules with same selectors. According to Definition 4, for each selector there must be one rule. As we are inferring 1-sided contextual rule, it does not satisfy our Definition 4. In the next section we will convert 1-sided contextual rule into 2-sided contextual rule in order to take care of over generalization.

7 Controlling over Generalization - Pseudocode-Steps: 16–20

In this section we determine the number of applications of each rule to generate the given input set. It is presented in table. We put priority in applying rules where left context is empty and context is smaller in size. If it is found that without using any rule we can generate full input set then we can ignore that rule.

- Using steps 16, 17 - we scan all the correct contextual rule for all the member of input set.
- **Table.insert[FIND − NOF − APP − of − EACHRULE − in − EACH MEMBER(1 − Sided − Correct − Rule$_i$, IP$_i$)]:** It finds out the application of each rule on each member of the input and insert that record into the table.
- **2 − Sided − Correct − Rule.push[MERGE(1 − Sided − Correct − Rule$_i$, 1 − Sided − Correct − Rule$_j$)]:** In this case if we find that ith row ($TableRow_i$) and jth row ($TableRow_j$) of the table are same then we merge these two rules and store as a $2 - Sided - Correct - Rule$.

Actually all the rules are 1-sided where left contexts or right contexts are empty that generates more elements. Thus, to control this over generalization, we check that how many times each rule is applied in each member of the input set. Rules which are applied equal number of times in each member, those can be merged into one rule based on condition.(discussed in Lemmas 12 and 13)

Lemma 12. *If consecutive selectors are sel_i, sel_j with $(j - i) = 1$ and left contexts(right contexts) are null in both of the rule then we can get 2-sided internal contextual rule after merging them.*

Proof. Here sel_i, sel_j denote ith and jth selector, v_i, v_j are right contexts of them respectively, and u_i, u_j are ith and jth left context of them respectively.

- $R_i : (sel_i, (u_i, v_i))_{icg}, R_j : (sel_j, (u_j, v_j))_{icg}$.
- **case 1:** If sel_i, sel_j where $(j - i) = 1$, if $v_i = v_j = \lambda$ then rule becomes $R_{new} : (sel_{new}, (u_{new}, v_{new}))_{icg}$ where $sel_{new} = sel_i, v_{new} = u_j$.
- **case 2:** If sel_i, sel_j where $(j - i) = 1$, if $u_i = u_j = \lambda$ then rule becomes $R_{new} : (sel_{new}, (u_{new}, v_{new}))_{icg}$ where $u_{new} = v_i, sel_{new} = sel_j$.

Lemma 13. *If consecutive selectors are sel_i, sel_j with $(j - i) = 1$ and left contexts of ith rule and right context of jth rule is null then we can get 1-sided internal contextual rule after merging them.*

Proof. Here sel_i, sel_j denote ith and jth selector, u_i, v_j are left contexts of ith rule and right context of jth rule respectively.

- $R_i : (sel_i, (u_i, v_i))_{icg}, R_j : (sel_j, (u_j, v_j))_{icg}$.
- If sel_i, sel_j where $(j - i) = 1$, if $u_i = v_j = \lambda$ then rule becomes $R_{new} : (sel_{new}, (u_{new}, v_{new}))_{icg}$ where $sel_{new} = sel_i, v_{new} = v_i v_j$.

7.1 Finding Maximal Use of Selectors - Step 21

In this subsection we show how to identify regular selector set and use the maximal idea. In this section we denote our already obtained individual selectors as SEL.

$COM(A, B)$ computes the common subword between A, B. On the other hand, $PREF(A), SUFF(B)$ denote the prefix and suffix part of A, B respectively, sel stands for selector.

Lemma 14. *For any selector and associated context with it, if* $COM(PREF(SEL), SUFF(u)) \neq \lambda$ *or* $COM(PREF(v), SUFF(SEL)) \neq \lambda$ *then we get regular selector set and we focus on the maximal use of selectors.*

Proof. – $COM(PREF(sel)), SUFF(u))$: If a rule is $(SEL, (u, v))_{icg}$ where $SEL = X_1X_2..X_k, u = u_1u_2...u_m$ where $X_1 = u_j, X_2 = u_{j+1}, \ldots, X_n = u_m, j \geq 1$ then the regular selector set becomes $SEL = (X_1X_2...X_n)^*X_{n+1}$ $X_{n+2}....X_k$.

– $COM(PREF(v), SUF(SEL))$: If a rule is $(SEL, (u, v))_{icg}$ where $SEL = X_1X_2...X_k, v = v_1v_2...v_m$ where $X_j = v_1, X_{j+1} = v_2, \ldots, X_k = v_n, j \geq 1$ then the selector set becomes $SEL = X_1X_2...(X_jX_{j+2}...X_k)^*$. (see example)

Remark 1. The above algorithm can also be used to identify a k-uniform internal contextual grammar with local maximum selectors. A required modification is that k is also given along with the input set.

In this case, at the time of defining insertion rule (Sect. 6.2), we need to focus on the size of selectors and contexts in terms of column as k is given as an input. Defining insertion rule should be done in the following way, $uxv \in sub(s_j^e)$ where $|u| = |v| = k$.

8 Characteristic Sample

The most widely used definition of data efficiency relies on the notion of characteristic sample. The characteristic sample is a finite set of data from a language L that ensures the correct convergence of the algorithm on any presentation of L as soon as it is included in the data seen so far.

Definition 6 (Characteristic Sample - CS). *If Let* L *be a* $SICL_{LM}$ *then a finite set* CS *is called a characteristic sample of* L *if and only if* L *is the smallest* $SICL_{LM}$ *containing* CS.

Consider $G_1 = (\{a, b, c\}, \{abc\}, (b^+, \{(a, bc)\}))$ where $first(u) = a \neq first(v) = b, L(G_1) = \{a^nb^nc^n \mid n \geq 1\}$. Here, $CS = \{abc, aaabbbccc, aabbcc, aaaabbbbcccc\}$.

When the input set IPS of the identification algorithm IA contains all the elements of CS, the algorithm converges to a correct final guess for the target $SICL_{LM}$. Hence, it is clear from the manner in which the characteristic sample CS is formed that, the class SLM is identifiable in the limit from positive data.

9 Time Complexity of Our Algorithm

We analyze the time complexity of our algorithm in two aspects, *time for updating a conjecture* and a bound on the *number of implicit errors* of guesses. We adapt this idea of time complexity analysis from [16]. Here we make an analysis of our pseudocode step by step.

- Step 1: If k number of strings are given in the input set, then we need to find out the size of each member of the set, so here the time complexity depends on the number of input member k and size of each member of input set. So, the time taken by step 1 is $SumofSize(IPS)$.
- Step 2: In order to generate all the possible subarrays, it takes polynomial time in the size of the axiom that is $Size(axiom)$. Also when we search the substring (uxv) in the examining input then it takes even linear time of the size of the examining input that is $Size(ExaminingInput)$.
- Step 3–6: It is only declaration.
- Step 7, 8: It can be seen easily that these two steps take constant time. Also removing one element from the input set, takes constant time. 5
- Step 9–15: Let k be the number of input arrays in the input set. If all the rules are correct then we do not need to go for any correction, so time complexity depends on k and $SumofSize(IPS)$. Also for any incorrect rule we need go for correction and the correction part takes polynomial time in the size of the input set that is $SumofSize(IPS`)$, finally these step 9–15 can be executed in polynomial time in the size of the set.
- Step 16–18: These three steps depend on the number of 1-sided-contextual-rule, let it be l and again the size of the input set.
- Step 19–20: In these two steps, firstly we search the table we if we find any two rows are same then merge these two 1-sided-contextual-rule. So, it depends on the size of the table.
- Step 21: It finds the regular selector. In this case, running time depends on the size of the correct contextual rule.

Lemma 15 (Time for updating a conjecture). *The identification algorithm IA identifies a target grammar g_f, in the limit, from positive data, satisfying the property that the time for updating the conjecture is bounded by polynomial in the size of the, $SumofSize(IPS)$.*

Proof. From the above discussion we can conclude that. □

Lemma 16 (Number of implicit errors of guesses). *The number of implicit errors of guesses of IA, is bounded by polynomial in the cardinality of set IPS.*

Proof. From the previous step by step running time discussion we can conclude that. □

Summing up the previous discussion about the definition of running time complexity [16] and last two lemmas, we have the following theorem.

Theorem 7. *The identification algorithm IA can be implemented to run in time polynomial in the size of IPS for updating conjecture, and the number of implicit errors of guesses is bounded by polynomial in the cardinality of set IPS.*

10 Example Run

Given input at time-unit t_1 is $i_{t1} = IPS = \{s_1 = abbcdd, s_2 = aabbbccddd\}$. Examining string $s_2^e = s_{21}^e s_{22}^e \dots s_{210}^e = aabbbccddd$, Axiom $s_1^a = s_{11}^a s_{12}^a \dots s_{16}^a = abbcdd$.

Defining Insertion Rule:

- $(u, x, v)_{ins}$ where $u = a, v = bbcdd$, Check whether any $|x| \leq r$ exists with $uxv \in sub(s_2^e)$? No-go to part 2.
- $(u, x, v)_{ins}$ where $u = a, v = bbcd$, Check whether any $|x| \leq r$ exists with $uxv \in sub(s_2^e)$? No-go to part 2.
- $(u, x, v)_{ins}$ where $u = a, v = bbc$, Check whether any $|x| \leq r$ exists with $uxv \in sub(s_2^e)$? Yes- $x = ab$, go to part 3.

According to condition 2 of Subsect. 6.2, $(|u| + |v|)^{ins} \leq |s_j^a|$, so for the next insertion rule - $u = a, x = ab$ are removed from the examining string and u is removed from axiom. Therefore string becomes $bbccddd$ and temporary axiom will be $bbcdd$. Existing $v = bbc$ will be considered as u (left context) for the next insertion rule. $(u, x, v)_{ins}$ where $u = bbc, v = dd, x = c$.

Now according to condition 3 of Subsect. 6.2, $(|u| + |x| + |v|)_{ins} \leq |E|$ but $(|u|+|v|)_{ins} = |s_j^a|$, so in this new insertion rule existing $v = dd$ will be considered as a u (for last selector) of the new rule. Now the axiom is covered completely and the rest part of the string will be considered as x of the next insertion rule. $(u, x, v)_{ins}$ where $u = dd, v = \lambda, x = d$. Finally the insertion rules are

- $(u, x, v)_{ins}$ where $u = a, v = bbc, x = ab$
- $(u, x, v)_{ins}$ where $u = bbc, v = dd, x = c$
- $(u, x, v)_{ins}$ where $u = dd, v = \lambda, x = d$

Converting into Contextual Rule: For A_1

- $R_1 : (sel_1, (u_1, v_1))_{icg}$ where $sel_1 = a, v_1 = ab, u_1 = \lambda, R_2 : (sel_2, (u_2, v_2))_{icg}$ where $sel_2 = bbc, v_2 = c, u_2 = \lambda, R_3 : (sel_3, (u_3, v_3))_{icg}$ where $sel_3 = dd, v_3 = d, u_3 = \lambda$.

Next input set at time-unit t_2 is $i_{t2} = IPS = \{s_3 = aaabcccd, s_4 = aabccd\}$. s_4 will be the new member of axiom set because s_1, s_4 both are of same lengths. $abbcdd, aabccd$ are considered as A_1, A_2 respectively.

Try to apply R_1, R_2, R_3 on s_3 but here we are not getting proper selectors also, so we need to define the insertion rule again. But here defining insertion is not possible from axiom A_1, so we define he insertion rule from A_2.

- $(u, x, v)_{ins}$ where $u = a, v = abcc, x = a$.
- $(u, x, v)_{ins}$ where $u = abcc, v = d, x = c$.

After converting into contextual rules: for A_2-

- $R_1 : (sel_1, (u_1, v_1))_{icg}$ where $sel_1 = a, v_1 = a, u_1 = \lambda, R_2 : (sel_2, (u_2, v_2))_{icg}$ where $sel_2 = abcc, v_2 = c, u_2 = \lambda$.

Now we will check that from A_2, generating S_1 is possible or not. Here A_2 is correct axiom for s_1. So the rules will be after converting into contextual rule-

- $R_3 : \{(sel_3, (u_3, v_3))_{icg}$ where $sel_3 = aa, v_3 = bb, u_6 = \lambda \mid (i \geq 1)\}, R_4 : \{(sel_4, (u_4, v_4))_{icg}$ where $sel_4 = bccd, v_4 = d, u_4 = \lambda \mid (i \geq 1)\}$.

Input at time-unit t_3 is $i_{t3} = IPS = \{s_5 = aabbccdd\}$.

- From A_1, deriving s_5 is possible because selectors are matching but needs to make the correction.
- In the same way we can verify that from A_2 it is possible to generate s_5 or not, using second set of rule. Selectors are not matching, so need to define insertion rule. Here defining insertion rule is possible, it suggests that axiom is correct for S_5.
- In the same way we can define insertion rules and convert insertion rules into contextual rule to reach S_5 from A_2.
- $R_5 : (sel_4, (u_4, v_4))_{icg}$ where $sel_4 = aa, v_1 = b, u_1 = \lambda, R_6 : (sel_5, (u_5, v_5))_{icg}$ where $sel_5 = bccd, v_5 = d, u_5 = \lambda$.
- Now with this new existing set of rule for A_2, generating S_2 is possible.

Making Correction and Updating Rules

- Here we are making the correction of R_1 (for A_1) to generate S_5. Now let v_1 be $V_1V_2 = ab$ where $w = 2$. $Xsel_1Q_1sel_2Z = aabbccdd$ where $sel_1 = a, Q_1 = a, sel_2 = bbc, X = \lambda, Z = cdd$.
- $Q_1 = a = V_1$, Here ($f = z = 1$ and $s < w = 2$). (According to Lemma 6)
- R_1 is changed and it becomes $R_1 : (sel_1, (u_1, v_1))_{icg}$ where $sel_1 = a, v_1 = a, u_1 = \lambda$.

New the set of rule will be after making the correction-for A_1 to generate S_2, S_5.

- $R_1 : (sel_1, (u_1, v_1))_{icg}$ where $sel_1 = a, v_1 = a, u_1 = \lambda$
- $R_2 : (sel_2, (u_2, v_2))_{icg}$ where $sel_2 = bbc, v_2 = c, u_2 = \lambda$
- $R_3 : (sel_3, (u_3, v_3))_{icg}$ where $sel_3 = dd, v_3 = d, u_3 = \lambda$
- $R_4 : (sel_4, (u_4, v_4))_{icg}$ where $sel_4 = bbc, v_4 = \lambda, u_4 = b$

From A_2-possible to generate s_2, s_3, s_5

- $R_1 : (sel_1, (u_1, v_1))_{icg}$ where $sel_1 = a, v_1 = a, u_1 = \lambda$
- $R_2 : (sel_2, (u_2, v_2))_{icg}$ where $sel_2 = abcc, v_2 = c, u_2 = \lambda$
- $R_3 : (sel_3, (u_3, v_3))_{icg}$ where $sel_4 = aa, v_4 = b, u_4 = \lambda$
- $R_4 : (sel_4, (u_4, v_4))_{icg}$ where $sel_5 = bccd, v_5 = d, u_5 = \lambda$

Controlling over Generalization and Finding Maximum Selectors. So two sets of rules are here, one is for axiom A_1 and another one is for axiom A_2. Now we will check that how many times each rule has been used in each string and that controls the over generalization. Table 1 contains application of each rule for $A_1, s_2 = aabbbccddd, s_5 = aabbccdd$. Also Table 2 contains application of each rule for $A_2, s_2 = aabbbccddd, s_3 = aaabcccd, s_5 = aabbccdd$.

Table 1. Finding application of each rule for A_1

Rules	s_2	s_5
$R_1 = (a, (\lambda, a))$	1	1
$R_2 = (bbc, (\lambda, c))$	1	1
$R_3 = (dd, (\lambda, d))$	1	0
$R_4 = (bbc, (b, \lambda))$	1	0

For A_1, we can merge $(R_1, R_2), (R_3, R_4)$, So according to Lemmas 12, 13 and 14. $R_{12} = (bbc^+, (a, c)), R_{34} = (b^+bc, (b, d))$, we can write $R_{12} = (b^+c^+, (a, c))$, $R_{34} = (b^+c^+, (b, d))$.

Table 2. Finding application of each rule for A_2

RULE	s_2	s_3	s_5
$R_1 = (a, (\lambda, a))$	0	1	0
$R_2 = (abcc, (\lambda, c))$	0	1	0
$R_3 = (aa, (\lambda, b))$	2	0	1
$R_4 = (bccd, (\lambda, d))$	2	0	1

For A_2, we can merge $(R_1, R_2), (R_3, R_4)$, So according to Lemmas 12, 13 and 14 - $R_{12} = (a^+b^+c^+, (a, c)), R_{34} = (b^+c^+d^+, (b, d))$.

References

1. Marcus, G.F.: Negative evidence in language acquisition. Cognition **46**, 53–85 (1993)
2. Oates, T., Desai, D., Bhat, V.: Learning k-reversible context-free grammars from positive structural examples. In: Proceedings of the Nineteenth International Conference on Machine Learning (2002)
3. Oates, T., Armstrong, T., Harris, J., Nejman, M.: On the relationship between lexical semantics and syntax for the inference of context-free grammars. In: Proceedings of AAAI, pp. 431–436 (2004)
4. Giammarresi, D., Restivo, A.: Two dimensional languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, pp. 215–267. Springer, Heidelberg (1997). doi:10.1007/978-3-642-59126-6_4
5. Rosenfeld, A., Sironmoney, R.: Picture languages - a survey. Lang. Des. **1**, 229–245 (1993)
6. Haussler, D.: Insertion and iterated insertion as operations on formal languages. Ph.D. Thesis, University of Colorado, Boulder (1982)
7. Kari, L.: Contextual insertions/deletions and computability. Inf. Comput. **1**, 47–61 (1996)

8. Oates, T., Armstrong, T., Bonache, L.B., Atamas, M.: Inferring grammars for mildly context sensitive languages in polynomial-time. In: Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., Tomita, E. (eds.) ICGI 2006. LNCS, vol. 4201, pp. 137–147. Springer, Heidelberg (2006). doi:10.1007/11872436_12
9. Marcus, S.: Contextual grammars. Revue Roumane de Mathematiques Pures et appliques **14**(10), 1525–1534 (1969)
10. Emerald, J.D., Subramanian, K.G., Thomas, D.G.: Inferring subclasses of contextual languages. In: Oliveira, A.L. (ed.) ICGI 2000. LNCS, vol. 1891, pp. 65–74. Springer, Heidelberg (2000). doi:10.1007/978-3-540-45257-7_6
11. Ilie, L.: Some recents results on contextual languages. TUCS Technical report No 96 (1997)
12. Gold, E.M.: Language identification in the limit. Inf. Control **10**, 447–474 (1967)
13. Ehrenfeucht, A., Paun, G., Rozenberg, G.: Contextual grammars and formal languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Language, vol. 2, pp. 237–293. Springer, Heidelberg (1997). doi:10.1007/978-3-662-07675-0_6
14. Fernau, H., Freund, R., Holzer, M.: Representations of recursively enumerable array languages by contextual array grammars. Fundamenta Informatica **64**, 159–170 (2005)
15. Martin-Vide, C., Mateescu, A., Miguel-Verges, J., Paun, G.: Internal contextual grammars: minimal, maximal, and scattered use of selectors. In: Kappel, M., Sgamir, E. (eds.) Bisfai 95 Conference on Natural Languages and AI, Jerusalem, pp. 132–142 (1995)
16. Yokomori, T.: Polynomial-time identification algorithm of very simple grammars from positive data. Theor. Comput. Sci. **1**(298), 179–206 (2003)
17. Rama, R., Smitha, T.A.: Some results on array contextual grammars. Int. J. Pattern Recogn. Artif. Intell. **14**, 537–550 (2000)