Learning Subclasses of Parallel Communicating Grammar Systems

Sindhu J. Kumaar[1], P.J. Abisha[2], and D.G. Thomas[2]

[1] Department of Mathematics, B.S. Abdur Rahman University
Chennai - 600 048, Tamil Nadu, India
sindhujkumaar@yahoo.co.in
[2] Department of Mathematics, Madras Christian College
East Tambaram, Chennai - 600 059, Tamil Nadu, India
pjabisha@yahoo.com, dgthomasmcc@yahoo.com

Abstract. Pattern language learning algorithms within the inductive inference model and query learning setting have been of great interest. In this paper an algorithm to learn a parallel communicating grammar system in which the master component is a regular grammar and the other components are pure pattern grammars is given.

1 Introduction

Inferring a pattern common to all words in a given sample is a typical instance of inductive inference [5]. Motivated by the study of Angluin on pattern languages [3] a generative device called pattern grammar is defined by Dassow et al. [4]. In [1], a generative device called pure pattern grammar is defined. In pure pattern grammar variables are not specified, instead constants themselves are replaced by axioms initially and the process is continued with the current set of words to get the associated language. A parallel communicating (PC) grammar system consists of several grammars working synchronously, each on its own sentential form and communicating by request. Here we give an algorithm to learn a parallel communicating grammar system, in which the master component is a regular grammar and the remaining components are pure pattern grammars.

2 Pure Pattern Grammar

Definition 1. *[1] A pure pattern grammar (PPG) is a triple $G = (T, A, P)$ where T is an alphabet, $A \subseteq T^*$ is a finite non empty set of elements of T^* called axioms and P is a finite non empty subset of T^+ called the set of patterns. For a set P and a language $L \subseteq T^*$, let $P(L)$ be the set of strings obtained by replacing uniformly and in parallel, each letter of all patterns in P by strings in L, all different occurrences of the same letter in a pattern being replaced by the same string.*

The pure pattern language (PPL) generated by G denoted by $L(G)$ is the smallest language $L \subseteq T^*$, for which we have $P \subseteq L$, $A \subseteq L$, and $P(L) \subseteq L$. In fact $L(G) = P \cup A \cup P(A) \cup P(P(A)) \cup \ldots$.

J.M. Sempere and P. García (Eds.): ICGI 2010, LNAI 6339, pp. 301–304, 2010.

Example 1. $G_1 = (\{a\}, \{a\}, \{aa\})$,
$L(G_1) = \{a, aa, aaaa, \dots\} = \{a^{2^n}/n \geq 0\}$
$P = \{aa\}$, $A = \{a\}$, $P(A) = \{aa\}$, $P(P(A)) = \{aaaa\}$, ...

Definition 2. *A parallel communicating pure pattern grammar system PC(PPG) is a construct* $\Gamma = (N, T, K, (P_0, S_0), (P_1, A_1), \dots, (P_n, A_n))$ *where* N, T, K *are non empty pairwise disjoint finite alphabets,* N *is the set of nonterminals,* T *is set of terminals, with* $K = \{Q_1, \dots, Q_n\}$ *set of query symbols,* $S_0 \in N$. $(N, T \cup K, P_0, S_0)$ *is a regular grammar and* (T, A_i, P_i) *are pure pattern grammars.*

The rewriting in the component (P_i, A_i) *is done according to the PPG. i.e.,* $P_i^k(A_i)$ *is considered in the* k^{th} *step, until a query is asked. If a query symbol* Q_j *appears in the master component* (P_0, S_0)*, then the strings in the* j^{th} *component are communicated to the master component. The language generated by such a system is the set of all words in* T^* *generated by the master component and it is called a parallel communicating pure pattern language and we write in short as PC(PPL).*

Example 2. $\Gamma = (N, T, K, (P_0, S_0), (P_1, A_1))$, $N = \{S_0\}$; $T = \{a, b\}$;
$K = \{Q_1\}$; $P_0 = \{S_0 \to aS_0, S_0 \to bS_0, S_0 \to aQ_1, S_0 \to bQ_1\}$;
$P_1 = \{aba\}$, $A_1 = \{a, ab\}$
$(S_0, \{aba\}) \Rightarrow (aS_0, \{aaa, aaba, abaab, ababab\})$
$\Rightarrow (abQ_1, \{a^9, aaaaabaaaa, aaaabaabaaa, \dots\})$
$\Rightarrow (ab\{a^9, aaaaabaaaa, aaaabaabaaa, \dots\}, y)$ where $y = aba$.
$L(\Gamma) = \{\{a, b\}^+\{a^9, aaaaabaaaa, aaaabaabaaa, \dots\}\}$, if Γ works in the returning mode.

3 Learning Parallel Communicating Grammar Systems PC (PPL)

In this section we attempt to learn the parallel communicating grammar systems in which the master component is a regular grammar and the second component is a pure pattern grammar with a single pattern. The algorithm to learn PCPPL is as follows:

1. From the language generated by the parallel communicating grammar system, one string w of length 'r' is given as the input. We assume that the length of the pattern n, the maximum length m of the axiom and alphabet T are known, using restricted superset queries the pattern is learnt and by restricted subset queries the set of axioms of the required pure pattern grammar are learnt. This checking is done till an equivalent pure pattern grammar is got.
2. Now, we check whether $x = x_1x_2 \dots x_r$ is a member of the pure pattern language. If yes, the program halts because we have already learnt pure pattern grammar. Otherwise split the first character x_1 from the left extreme of the sample x and check whether the remaining string is a member of

the required pure pattern language. This process is repeated till we get the longest suffix $x_{i+1}x_{i+2}\dots x_r$ of x which is a member of the pure pattern language. For the remaining prefix $x_1x_2\dots x_i$ of x, we ask membership query i.e., we first check if the string $x_1x_2\dots x_i$ is a member of the required regular set the language generated by the master component, then taking $x_1x_2\dots x_i$ as a sample we try to learn the regular grammar.

Algorithm 1

Input: The alphabet T, a positive sample $w \in T^+$ of length r with $w = w_1w_2\dots w_r$, the length 'n' of the pattern, the maximum length 'm' of the axiom, $r \geq n$, words $x_1, x_2, \dots, x_s$ of $\bigcup_{i=1}^{m} T^i$ given in the increasing length order, among words of equal length according to lexicographic order.

Output: A parallel communicating grammar system $\Gamma' = (N, T, Q, (P_0', S_0'), (P_1', A_1'))$ with $L(\Gamma') = L(\Gamma)$

Procedure (Pattern)

begin
  Let $u_1, u_2, \dots, u_t$ be the words in T^n in the lexicographic order
  for $i = 1$ to t
    begin
      ask the restricted super set query for $\left(T, \bigcup_{i=1}^{m} T^i, \{u_i\}\right)$, $u_i \in T^n$
      if yes then $p = u_i$
        call(Axiom)
      else $i = i + 1$
  end

Procedure (Axiom)

Let $x_1, x_2, \dots, x_s$ be the words in $\bigcup_{i=1}^{m} T^i$ arranged in lexicographic order
$A = \phi$
  for $t = 1$ to s do
    begin
      ask restricted subset query for $G = (T, A \cup \{x_t\}, \{p\})$
      If 'yes' then $A = A \cup \{x\}$ and $t = t + 1$
        else output G
    end
Print the pure pattern grammar (T, A, p)

Procedure (Master)

for $i = 1$ to $r - 1$
  begin
    Ask membership query for $w_{i+1}\dots w_r$, is $w_{i+1}\dots w_r \in (T, A, \{p\})$?
      If yes, then
        for the prefix $x = w_1w_2\dots w_i$, ask membership query,
          is $xq \in L(N, T \cup \{q\}, P_0, S_0)$
          If yes, then run L^* using prefixes of x. If L^* gives the
            correct automaton, write the corresponding regular

grammar which is equivalent to $G_0 = (N, T \cup \{q\}, P_0, S_0)$
else $i = i + 1$
else $i = i + 1$
Print $\Gamma' = (N, T, Q, (P'_0, S'_0), (P'_1, A'_1))$ the PC grammar system
end
end

Time Analysis: As each of the procedures runs in polynomial time the algorithm to learn PC(PPL) is also polynomial.

References

1. Abisha, P.J., Subramanian, K.G., Thomas, D.G.: Pure Pattern Grammars. In: Proceedings of International Workshop Grammar Systems, Austria, pp. 253–262 (2000)
2. Abisha, P.J., Thomas, D.G., Sindhu J. Kumaar: Learning Subclasses of Pure Pattern Languages. In: Clark, A., Coste, F., Miclet, L. (eds.) ICGI 2008. LNCS (LNAI), vol. 5278, pp. 280–283. Springer, Heidelberg (2008)
3. Angluin, D.: Learning Regular Sets from Queries and Counter Examples. Information and Computation 75, 87–106 (1987)
4. Dassow, J., Paun, G., Rozenberg, G.: Generating Languages in a Distributed Way: Grammar Systems. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages. Springer, Heidelberg (1997)
5. Gold, E.M.: Language Identification in the Limit. Information and Control 10, 447–474 (1967)
6. Salomaa, A.: Formal Languages. Academic Press, New York (1973)