Learning Subclasses of Pure Pattern Languages

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

[1] Department of Mathematics, Madras Christian College, Chennai - 600 059
[2] Department of Mathematics, Crescent Engineering College, Chennai - 600 048
sindhujkumaar@yahoo.co.in

Abstract. Pattern language learning algorithms within the inductive inference model and query learning setting have been of great interest. In this paper, we study the problem of learning pure pattern languages using queries and examples.

1 Introduction

Inductive inference introduced by Gold [6], is a model that treats as an infinite process, which identifies the unknown concept in the limit. Inferring a pattern common to all words in a given sample is a typical instance of inductive inference. Motivated by the study of Angluin [2], a generative device called pattern grammar is defined by Dassow et al. [5]. In [1], a new generative device called the pure pattern grammar is defined. We do not specify variables, instead constants themselves are replaced by axioms initially and the process is continued with the current set of words to get the associated language. As the study of pattern languages is motivated by the inference problem and there are algorithms to learn subclasses of pure languages, it is of interest to analyze the inference problem for pure pattern languages. Here we give algorithms to learn two subclasses of the family of pure pattern languages using queries.

2 Pure Pattern Grammars

Definition 1. *A pure pattern grammar is a triple $G = (\Sigma, A, P)$ where Σ is an alphabet, $A \subseteq \Sigma^*$ is a finite set of elements of Σ^* called axioms and P is a finite subset of Σ^+ called the set of patterns. For a set P and a language $L \subseteq \Sigma^*$, let $P(L)$ be the set of strings obtained by replacing, uniformly and in parallel each letter of the patterns in P, by strings in L, occurrences of the same letter of a pattern in a particular step being replaced by the same string. The language (PPL) generated by G, denoted by $L(G)$, is the smallest $L \subseteq \Sigma^*$ for which we have $P \subseteq L$, $A \subseteq L$ and $P(L) \subseteq L$. Here $L(G) = P \cup A \cup P(A) \cup P(P(A)) \cup \ldots$.*

Proposition 1. *(i) Any finite set F with at least one word p of length 1 is a pure pattern language, (ii) The families of pure context free languages (PCF), context free languages (CFL), regular languages (RL) are incomparable with the family of pure pattern languages (PPL). The family of pure pattern languages generated by grammars with a single pattern is strictly included in the family of deterministic tabled 0L languages.*

A. Clark, F. Coste, and L. Miclet (Eds.): ICGI 2008, LNAI 5278, pp. 280–282, 2008.
© Springer-Verlag Berlin Heidelberg 2008

3 Learning a Subclass of PPL

We now give a polynomial time algorithm to learn a subclass of PPL using the restricted subset queries and restricted superset queries. The inclusion problem for this class is decidable, as the difference between the pattern languages with a single pattern over only variables and the pure pattern languages of this class lies mainly in the patterns. The technique of the algorithm is as follows: First, the pattern is learnt using restricted superset queries and then the axioms are learnt using restricted subset queries. We assume that length of the pattern is known (fixed), say 'n' and the length of the longest axioms is known, say 'm'. If $\Sigma = \{a_1, a_2, \ldots, a_k\}$ is the known alphabet of the language to be learnt then the axiom set A is a subset of all words over Σ of length i, $1 \leq i \leq m$. The words in Σ^n are lexicographically arranged and restricted superset queries for $\left(\Sigma, \bigcup_{i=0}^{m} \Sigma^i, p_i\right)$ where $p_i \in \Sigma^n$ are made. If the answer is yes, p_i is the pattern p; otherwise repeat the same procedure for the next word in Σ^n. Now, to learn axiom set A, initially fix $A = \phi$. Arrange the words in Σ^i, $i = 0$ to m, according to increasing order of length and among the words of equal length lexicographically. Let them be $w_1, w_2, \ldots, w_s$. At the t^{th} step, ask the restricted subset query for $(\Sigma, A \cup \{w_t\}, p)$. If the answer is 'yes', increment A to $A \cup \{w_t\}$. If the answer is 'no', A is not incremented. The output at the last step is the required PPG.

Algorithm

Input: An alphabet $\Sigma = \{a_1, a_2, \ldots, a_k\}$, $m = max\{|x_i| : x_i \in A\}$, $n = |p|$
Words $p_1, p_2, \ldots, p_r$ from Σ^n arranged lexicographically.
Words $w_1, w_2, \ldots, w_s$ are given in the increasing length order, among words of equal length according to lexicographic order.

Output: $G = (\Sigma, A, p)$

```
begin
    for t = 1 to r do
        begin
            ask restricted superset query for (Σ, ⋃_{i=0}^{m} Σ^i, p_t)
            If 'yes' then output p = p_t
                else t = t + 1
        end
    A = φ
    for t = 1 to s do
    begin
        ask restricted subset query for G = (Σ, A ∪ {w_t}, p)
        If 'yes' then A = A ∪ {w_t} and t = t + 1
            else output G
    end
end
```

4 Learning Another Subclass of PPL

In the MAT learning [3], the teacher / oracle can answer membership query and equivalence query. We consider any subclass of PPL for which membership and equivalence queries are decidable. In addition, we require that, the axiom set A is a code [4]. However, the PPG need not have only a single pattern. The algorithm to learn a pure pattern language $L(G)$ from the above subclass works as follows:

The fixed alphabet Σ of cardinality k and the axiom set A whose cardinality is greater than or equal to k are the inputs to the algorithm. Let the target PPG be $G = (\Sigma, A, P)$. Initially, when $j = 0$, the pattern set P_j is assumed to be empty by the algorithm. The learner asks the oracle, first the equivalence query for $L(G)$ and $L(G_j)$ where $G_j = (\Sigma, A, P_j)$. We have either $L(G_j) \subset L(G)$ or $L(G_j) = L(G)$. If the answer is positive, we obtain an equivalent grammar G_j of the target grammar. Otherwise, a positive sample word $x \in L(G) - L(G_j)$ is returned. If $x \notin A^+$. Then $P_{j+1} = P_j \cup \{x\}$ and $G_{j+1} = (\Sigma, A, P_{j+1})$. If $x \in A^+$, then the learner factorises x over A. It should be noted that this can be done in linear time by Sardinas-Patterson algorithm [4]. Let $x = x_1x_2x_3 \dots x_m$, $x_i \in A$. The learner finds the minimal prefix of x which belongs to $L(G)$. It is denoted by $min(x)$ and this is found by asking membership query to the oracle. Let $min(x) = w = x_1x_2 \dots x_r$ and $bagmin(x) = \{x'_1x'_2 \dots x'_s\}$ where x'_j $(1 \leq j \leq s)$ are distinct and $\{x_1, x_2, \dots, x_r\} = \{x'_1, x'_2, \dots, x'_s\}$. It should be noted that the number of distinct elements in $bagmin(x)$ is less than or equal to number of elements in A. Let $\{f_1, f_2, \dots, f_n\}$ be the set of all 1-1 morphisms from $bagmin(x)$ to Σ. There is atmost $k!$ such morphisms (i.e.,) $n \leq k!$. The learner asks membership query for $f_q(min(x))$ $(q = 1, 2, \dots, n)$. If the answer is positive the learner updates the pattern set P_j to $P_{j+1} = P_j \cup \{f_q(min(x))\}$ and asks the equivalence query for $L(G)$ and $L(G_{j+1})$, where $G_{j+1} = (\Sigma, A, P_{j+1})$. This process is repeated until we get a PPG equivalent to G.

References

1. Abisha, P.J., Subramanian, K.G., Thomas, D.G.: Pure Pattern Grammars. In: Proceedings of International Workshop on Grammar Systems, Austria, pp. 253–262 (2000)
2. Angluin, D.: Finding patterns common to a set of strings. Journal of Computer and System Sciences 21, 46–62 (1980)
3. Angluin, D.: Learning regular sets from queries and counter examples. Information and Computation 75, 87–106 (1987)
4. Berstel, J., Perrin, D.: The Theory of Codes. Academic Press, New York (1985)
5. Dassow, J., Paun, G., Salomaa, A.: Grammars based on patterns. International Journal of Foundations of Computer Science 4, 1–14 (1993)
6. Gold, E.M.: Language identification in the limit. Information and Control 10, 447–474 (1967)