Inference of Uniquely Terminating EML

S. Kannamma[1], D.G. Thomas[2], and K. Rangarajan[2]

[1] S.D.N.B. Vaishnav College for Women, Chennai - 600 044
[2] Madras Christian College, Chennai - 600 059
dgthomasmcc@yahoo.com

1 Introduction

In [3], we have provided an algorithm to infer a few subclasses of linear languages through labeled extended Petri nets. The family of equal matrix languages [6] meets both the families of context sensitive languages and context-free languages. In this paper, we prove that an equal matrix language is a Petri net language. We construct labeled extended Petri nets to infer uniquely terminating code k-equal matrix languages (utCk-EMLs), a subclass of EMLs. A similar algorithm can be employed to construct a labeled Petri net which generates a uniquely terminating regular language [4]. This algorithm can be modified to infer a given uniquely terminating code regular language [2] through a labeled extended Petri net.

2 Algorithm to Infer Uniquely Terminating Languages

For the notions of a Petri net language and an equal matrix language we refer to [5,6]. A code k-equal matrix language and uniquely terminating code k-equal matrix language can be defined similar to code regular language [1] and uniquely terminating code regular language [2] respectively. We can show the following results.

Theorem 1. *An equal matrix language (EML) is a Petri net language.*

Corollary 1. *A code k-equal matrix language (Ck-EML) is a Petri net language.*

We present an algorithm to infer a uniquely terminating code k - equal matrix language from positive data. This algorithm first develops a trie structure. It then gives rules to construct a labeled extended Petri net with its transitions labeled as code words over the given alphabet which generates the required language.

A set of sample words $\{w_1, w_2, \ldots, w_n\}$ from a uniquely terminating code k-equal matrix language and the code set K are given as inputs to the inference algorithm. The following is the procedure to construct a labeled extended Petri net with the transitions labeled with code words from K generating the required utCk - EML.

Algorithm PN - utCk - EML

1. For each word in the sample do

A. Clark, F. Coste, and L. Miclet (Eds.): ICGI 2008, LNAI 5278, pp. 289–291, 2008.

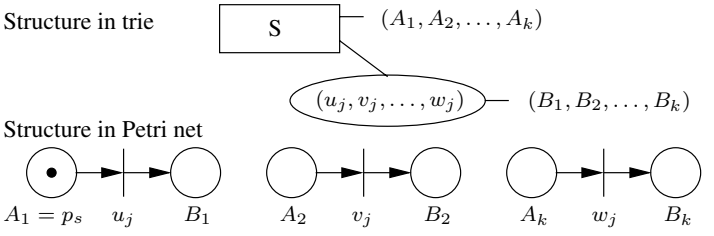
1a. Factorise the word using the code set K and partition into k-equal parts as $x_i = \alpha_1\alpha_2 \ldots \alpha_k$, $\alpha_i \in K^+$.

1b. If the α_i's are of the form $\alpha_1 = u_1u_2 \ldots u_n$; $\alpha_2 = v_1v_2 \ldots v_n$; ...; $\alpha_k = w_1w_2 \ldots w_n$, form the k-tuples $(u_1, v_1, \ldots, w_1)$, $(u_2, v_2, \ldots, w_2)$, ..., $(u_n, v_n, \ldots, w_n)$ and store them as labels of nodes in the trie structure and label the root node as a k-tuple of nonterminals, say $(A_1, A_2, \ldots, A_k)$; $(u_n, v_n, \ldots, w_n)$, is the label of terminal node. Insert associated k-tuple nonterminals to the new nodes other than the terminal nodes.

1c. If there are nodes having children which are equally labeled final nodes, then merge the associated nonterminals of these nodes.

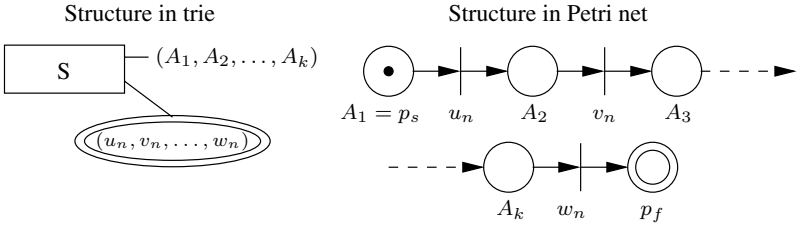
2. The construction of the resulting labeled extended Petri net is given as an output from the trie structure.

2a. If S has a child, a node labeled by the k-tuple $(u_j, v_j, \ldots, w_j)$ with associated nonterminal $(B_1, B_2, \ldots, B_k)$, then make places $p_s = A_1$, B_1, A_2, B_2, ..., A_k, B_k with a single token in p_s and no tokens in other places and make transitions respectively labeled as $u_j, v_j, \ldots, w_j$ and the flow relation as shown below:

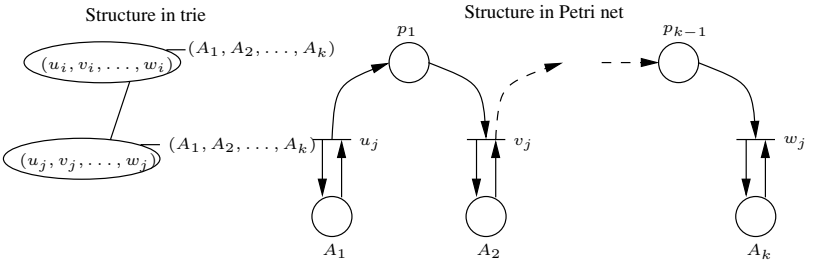


Remark: If S is not the parent node, then, the procedure mentioned above holds good with the only condition that none of the places introduced have tokens in them.

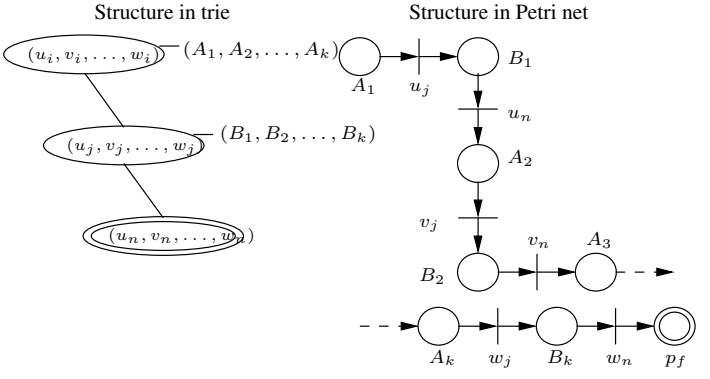
2b. If the node associated with S has a child which is a final node labeled $(u_n, v_n, \ldots, w_n)$, then make places $p_s = A_1, A_2, \ldots, A_k$ and p_f, where, p_f is a final place, with a single token in $p_s = A_1$ and no tokens elsewhere, as shown below:



2c. If a node labeled by the k-tuple $(u_j, v_j, \ldots, w_j)$ has a nonterminal $(A_1, A_2, \ldots, A_k)$ associated with it and if $(A_1, A_2, \ldots, A_k)$ is again the nonterminal associated with its parent, then make places $A_1, A_2, \ldots, A_k$ and $p_1, p_2, \ldots, p_{k-1}$; make transitions with labels as the words $u_j, v_j, \ldots, w_j$; the flow relation connecting these places and transitions is shown below:



2d. If the trie structure leading to a final node is a chain, the corresponding structure of the labeled extended Petri net is shown below:



The initial marking of the constructed Petri net is with one token in p_s and no tokens in other places and the final marking is with one token in p_f and no tokens in other places. The language generated by this Petri net is the utCk - EML inferred in a sequence of conjectures.

References

1. Emerald, J.D., Subramanian, K.G., Thomas, D.G.: Learning code regular and code linear languages. In: Miclet, L., de la Higuera, C. (eds.) ICGI 1996. LNCS, vol. 1147, pp. 211–221. Springer, Heidelberg (1996)
2. Emerald, J.D., Subramanian, K.G., Thomas, D.G.: A note on inferring uniquely terminating code languages. Information Processing Letters 70, 217–222 (1999)
3. Kannamma, S., Thomas, D.G., Rangarajan, K.: On inference of uniquely terminating linear languages. In: Thangavel, K., Balasubramaniam, P. (eds.) Computing and Mathematical Modeling, pp. 291–298. Narosa Publishing House (2006)
4. Makinen, E.: Inferring uniquely terminating regular languages from positive data. Information Processing Letters 62, 57–60 (1997)
5. Peterson, J.L.: Petri net theory and modeling of systems. Prentice Hall, Englewood Cliffs (1981)
6. Siromoney, R.: On equal matrix languages. Information and control 14, 135–151 (1969)