INFERRING UNIQUELY TERMINATING REGULAR LANGUAGES FROM POSITIVE DATA

Erkki Mäkinen

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF TAMPERE

REPORT A-1996-9

UNIVERSITY OF TAMPERE
DEPARTMENT OF COMPUTER SCIENCE
SERIES OF PUBLICATIONS A
A-1996-9, NOVEMBER 1996

INFERRING UNIQEULY TEMRINATING REGULAR LANGUAGES FROM POSITIVE DATA

Erkki Mäkinen

University of Tampere
Department of Computer Science
P.O. Box 607
FIN-33101 Tampere, Finland

ISBN 951-44-4060-9
ISSN 0783-6910

INFERRING UNIQUELY TERMINATING REGULAR LANGUAGES FROM POSITIVE DATA

Erkki Mäkinen
Department of Computer Science, University of Tampere,
P.O. Box 607, FIN-33101 Tampere, Finland

E-mail: em@cs.uta.fi

Abstract. We define a new subclass of regular languages, called uniquely terminating regular languages, which can be inferred from positive data. The class of uniquely terminating regular languages and the previously known subclasses of regular languages inferable from positive data are incomparable.

Key Words and Phrases: regular language, grammatical inference, identification in the limit.

1. Introduction

Regular languages cannot be inferred from positive data only [5]. This negative result has initiated a search for subclasses of regular languages having the desirable inference property. The found subclasses include, among others, k-reversible languages [1], strictly regular languages [10], locally testable languages in the strict sense [4], Szilard languages of regular grammars [7], and regular code languages [3]. This paper introduces still another subclass of regular languages inferable from positive data. The new class of languages is called uniquely terminating regular languages. This class of languages and the previously known subclasses of regular languages inferable from positive data are incomparable.

2. Uniquely terminating regular languages

We assume a familiarity with the basics of formal language theory and grammatical inference as given e.g. in [6] and [2], respectively. As inference criterion we use "identification in the limit" [5,2].

If not otherwise stated we follow the notations and definitions of [6]. The length of a string w is denoted by lg(w), and the left-quotient of L and w by

$$T_L(w) = \{ v \mid wv \in L \}.$$

We consider here regular grammars only. A production of the form $A \to b$, where b is a terminal, is said to be *terminating*; otherwise a production is said to be *continuing*. A continuing production has the form $A \to bB$, where b is a terminal and B is a nonterminal. Other forms of productions are not allowed.

Definition 1. A regular grammar $G = (V,\Sigma,P,S)$ is *uniquely terminating* if the productions in P fulfil the following conditions for each nonterminal A in G:

1. $A \to aB$ and $A \to aC$ imply $B = C$; i.e. the right hand sides of continuing A-productions start with unique terminals,

and

2. A has a unique terminating production; i.e. each nonterminal has exactly one terminating production and the terminals appearing in the right hand sides of terminating productions are all different. ⊠

Notice that the right hand side b of a unique terminating production $A \to b$ may appear in the right hand side of any continuing production, but not in the right hand side of any other terminating production.

A regular language L is said to be *uniquely terminating* (utr-language for short) if there is a uniquely terminating grammar generating L.

Nondeterminism does not increase the computational power of finite automata. In terms of regular grammars this means that each such grammar can be transformed to fulfil condition 1 of Definition 1. This transformation increases the number of nonterminals in the grammar, and conditions 1 and 2 are, in general, not attainable simultaneously. This means that utr-languages form a restricted subclass of regular languages. On the other hand, utr-languages are not included in any of the known subclasses of regular languages inferable from positive data only.

Example 1. Consider a regular grammar with productions $S \to aA$, $S \to bB$, $A \to aA$, $A \to cC$, $B \to aB$, $B \to cC$, $S \to a$, $A \to b$, $B \to c$, and $C \to d$. Condition 1 of Definition 1 holds for the continuing S-, A-, and B-productions, and each nonterminal has a unique terminating production. Hence, the grammar is a uniquely terminating grammar. It produces the language $\{ a \} \cup \{ a^n b \mid n \geq 1 \} \cup \{ a^n cd \mid n \geq 1 \} \cup \{ ba^m c \mid m \geq 0 \} \cup \{ ba^m cd \mid m \geq 0 \}$. ⊠

Recall that a regular language is k-reversible if and only if whenever u_1vw and u_2vw are in L and $lg(v) = k$, then $T_L(u_1v) = T_L(u_2v)$ [1]. The language of Example 1 is not k-reversible for any k. Namely, we may choose $u_1 = a$, $u_2 = b$, $v = a^k$, and $w = cd$. Now, $T_L(u_1v) = T_L(a^{k+1}) \neq T_L(ba^k) = T_L(u_2v)$. On the other hand, it is easy to find k-reversible languages (even with k = 0) which are not utr-languages. It is also easy to give similar examples with strictly regular languages, locally testable languages in the strict sense, Szilard languages of regular grammars, and with regular code languages showing that these classes of languages and utr-languages are incomparable.

3. The inference algorithm

Suppose we are given a set of sample words $\{ w_1, w_2, \ldots, w_n \}$ from a uniquely terminating regular language. We can give the corresponding derivations in the form

$$S \Rightarrow w_{i1}A_{i1} \Rightarrow \ldots \Rightarrow w_{i1}\ldots w_{ik}A_{ik} \Rightarrow w_{i1}\ldots w_{ik}w_{ik+1}$$

where $w_i = w_{i1}\ldots w_{ik}w_{ik+1}$, for i = 1,..., n, with the appropriate length k + 1. We can take the following steps when merging nonterminals A_{ij} in the inference algorithm:

1. If two sample words have a common proper prefix w, we know by condition 1 of Definition 1 that the common proper prefix is produced by using the same productions in both derivations. This means that we can merge the corresponding nonterminals.
2. By the uniqueness of terminating productions (condition 2 of Definition 1), we can merge the last nonterminals of derivations producing words which end with the same terminal. For example, if the sample contains words b and cdbab, we know that the last nonterminal appearing in the derivation of the longer sample word is the start symbol of the grammar in question.

Example 2. Consider a sample { abcd, abef, ab }. In order to apply step 1 above, we store the sample words in a trie structure as shown in Figure 1. Double circles indicate final nodes where a sample word ends. All the corresponding derivations start with the same production $S \rightarrow aA$ (the names of the nonterminals are naturally arbitrary). Two derivations (for abcd and abef) continue with the production $A \rightarrow bB$, while the third derivation terminates with production $A \rightarrow b$. There are two different B-productions, $B \rightarrow cC$ and $B \rightarrow eD$. The unique terminating C-production is $C \rightarrow d$ and the unique D-production is $D \rightarrow f$. If the sample is incremented with the word ccf, we obtain new productions $S \rightarrow cE$, $E \rightarrow cD$, and $D \rightarrow f$. Notice that we can indeed merge the nonterminals so that D is in the right hand side of the only E-production. ☒

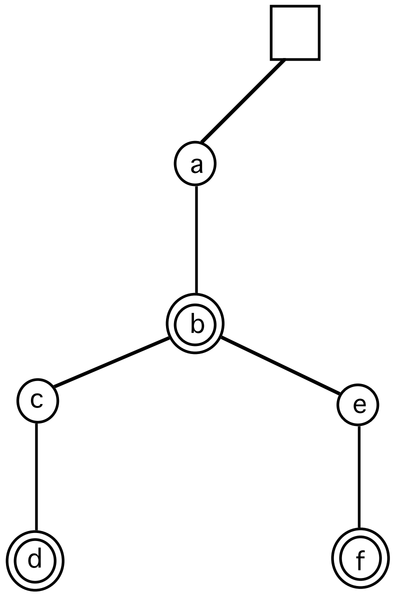

Figure 1. The trie corresponding to the sample { abcd, abef, ab }.

The output grammar obtained in Example 2 is not a uniquely terminating grammar, because some of the nonterminals (S, B, and F) do not have their unique terminating productions. Hence, the hypothesis space of our algorithm is the class of regular grammars fulfilling condition 1 of Definition 1. When the algorithm receives more samples, it will find more and more unique terminating productions, and finally, also condition 2 of Definition 1 will be fulfilled.

The inference algorithm for utr-languages reads its sample words and stores them in the trie. Each trie node (excluding the leaves) has an associated nonterminal. The root of the trie has S (the start symbol) associated with it. Figure 2 shows the trie of Figure 1 with its nodes augemented with the associated nonterminals. If a node labelled by b has B associated with it and A is the nonterminal associated with its parent, then we take the production $A \to bB$ to the resulting grammar.

If a node is associated with A and has a child labelled with b such that the child is a final node, then we take the production $A \to b$ to the resulting grammar. (This is the case in the trie of Figure 2.) The parents of all appearances of a leaf with the same label must have the same nonterminal associated with them. When the original sample { abcd, abef, ab } was incremented in Example 2 with the new word ccf, we had two words ending with f. The corresponding terminating production is $D \to f$. After the incrementation, the trie has two nodes associated with D. Notice that although a node in a trie can have as many children as there are terminals in the alphabet, only one of the children can be a final node.

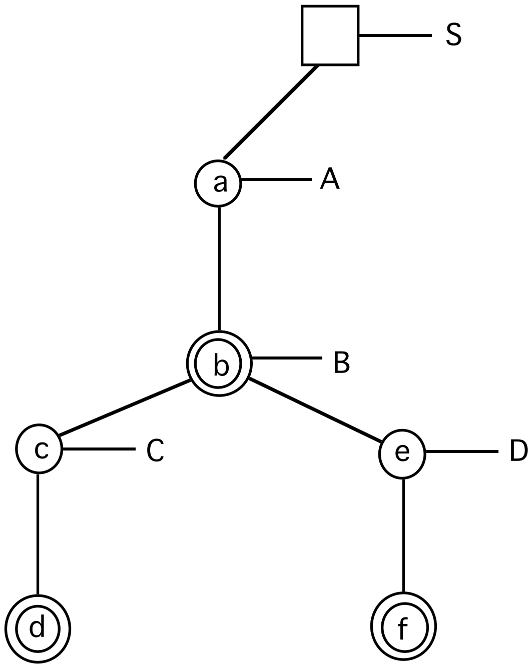

Figure 2. The augmented trie corresponding to the sample { abcd, abef, ab }.

The inference algorithm can be given as follows:

Algorithm UT;

1. **for** each word in the sample **do**

    1a. store the word in the trie; insert associated nonterminals to the new nodes;

    1b. **if** there are nodes having children which are equally labelled final nodes

    then merge the associated nonterminals of these nodes

    od

2. output the productions of the resulting grammar from the trie:

    2a. **if** a node is associated with A and its child labelled with b is associated with B

    then take $A \rightarrow bB$ to the grammar;

    2b. **if** the child of a node associated with A is a final node labelled with b

    then take $A \rightarrow b$ to the grammar;

Since each nonterminal has a unique terminating production, the desired uniquely terminating grammar will be eventually found. Terminating productions expose the use of the nonterminals in the derivations.

UT clearly produces a new conjecture using only a polynomial time in the size of the resulting grammar and in the length of the sample words. However, UT is not a "polynomial-time inference algorithm" in the original strong sense of Pitt [9], since the number of implicit errors of prediction is not polynomially bounded by the size of the resulting grammar. Notice that the class of Szilard languages of regular grammars has a polynomial-time inference algorithm in the sense of Pitt [8].

We can formulate our results as follows.

Theorem. Utr-languages can be inferred from positive data such that a new conjecture is always produced in polynomial time.

References

[1] Angluin, D., Inference of reversible languages. *J. ACM* **29** (1982), 741-765.

[2] Angluin, D., and Smith, C.H., Inductive inference: theory and methods. *ACM Comput. Surv.* **15** (1983), 237-269.

[3] Emerald, J.D., Subramanian, K.G., and Thomas, D.G., Learning code regular and code linear languages. In: *Proceedings of ICGI-96, Lecture Notes in Artificial Intelligence* **1147** (1996), 211-221.

[4] Garcia, P., Vidal, E., and Oncina, J., Learning locally testable languages in the strict sense. In: *Proceedings of the First International Workshop on Algorithmic Learning Theory*, 1990, 325-338.

[5] Gold, E.M., Language identification in the limit. *Inform. Contr.* **10** (1967), 447-474.

[6] Harrison, M.A., *Introduction to Formal Language Theory.* Addison-Wesley, 1978.

[7] Mäkinen, E., The grammatical inference problem for the Szilard languages of linear gramamrs. *Inf. Process. Lett.* **36** (1990), 203-206.

[8] Mäkinen, E., A family of languages which is polynomial-time learnable from positive data in Pitt's sense. *Intern. J. Computer Math.* **61** (1996), 175-179.

[9] Pitt, L., Inductive inference, DFAs, and computational complexity. In: *Proceedings of 2st Workshop on Analogical and Inductive Inference*, *Lecture Notes in Artificial Intelligence* **397** (1989),18-44.

[10] Tanida, N., and Yokomori, T., Polynomial-time identifiacation of strictly regular languages in the limit. *IEICE Trans. Inf & Syst.* **E75-D** (1992), 125-132.