A characterization of s-languages

Wojciech Fraczak and Anna Podolak

IDT Canada, Inc.
1575 Carling Av., Ottawa, Ontario, K1Z 7M3 Canada

Abstract

We show that a language L is an s-language if and only if the set of the quotients of L (i.e., the set of the states of its minimal deterministic automaton seen as languages) is a subset of a free monoid generated by a finite set of prefix codes. We demonstrate through examples how to use this result for deciding whether a given language is an s-language.

Keywords: formal languages, minimal deterministic automaton, prefix code, s-language

1 Introduction

Let Σ be a finite *alphabet*. The set of all words over Σ will be denoted by Σ^* and the empty word by ε. We say that word v is a *prefix* of word w if there exists a word u such that $vu = w$. For every language $L \subseteq \Sigma^*$ and every word w we define $w^{-1}L \stackrel{\text{def}}{=} \{v \in \Sigma^* \mid wv \in L\}$. A language L' is called a *quotient* of L if there exists a word w such that $L' = w^{-1}L$.

A language $L \subseteq \Sigma^*$ is called a *prefix code* if and only if L does not contain two different words such that one is a prefix of the other.

A class of s-languages was defined as a class of deterministic context free languages which can be described by an s-grammar, i.e., LL(1) grammar under Greibach normal form [KH66, KS69, ABB97]. In [KH66], it has been shown that s-languages are prefix codes.

In this paper we show that a language L is an s-language if and only if the set of the quotients of L is a subset of a free monoid $\mathcal{S}^*$ generated by a finite set $\mathcal{S}$ of prefix codes. The paper also examines practical applications of the theorem to determine whether a given language is an s-language.

2 S-languages, prefix codes, and D-articulation points

A context free grammar $G = (\Sigma, V, P)$ is composed of a finite set Σ of *terminals*, a finite set V of *nonterminals* disjoint from Σ, and a finite set P of *production* *rules* of form $(x \to \alpha)$, with $x \in V$ and $\alpha \in (V \cup \Sigma)^*$. For every $\beta, \gamma \in (V \cup \Sigma)^*$, if $(x \to \alpha) \in P$, then $\beta x \gamma \to \beta \alpha \gamma$. A *derivation* $\beta \xrightarrow{*} \gamma$ is a finite sequence $(\alpha_0, \alpha_1, \ldots, \alpha_n)$ such that $\beta = \alpha_0$, $\gamma = \alpha_n$, and $\alpha_{i-1} \to \alpha_i$ for $i \in [1, n]$. For every nonterminal x of G we define a language $G(x) \subseteq \Sigma^*$ by:

$$G(x) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid x \xrightarrow{*} w\}.$$

A grammar $G = (\Sigma, V, P)$ is a *Greibach normal form* grammar if for every production rule $(x \to \alpha) \in P$, we have $\alpha \in \Sigma V^*$. A grammar $G = (\Sigma, V, P)$ is an *s-grammar* if G is a Greibach normal form grammar and for every pair $(x, a) \in V \times \Sigma$ there is at most one production rule $(x \to \alpha) \in P$ such that α starts by a, i.e., $\alpha \in \{a\}V^*$.

A language $L \subseteq \Sigma^*$ is an *s-language* if there exists an s-grammar $G = (\Sigma, V, P)$ such that $G(x) = L$ for some $x \in V$. The definition implies that every nonterminal of an s-grammar defines an s-language.

In [KH66], the authors observed that every s-language is a prefix code.

Prime prefix codes are prefix codes that cannot be represented as a nontrivial concatenation of two prefix codes. Since nonempty prefix codes under concatenation constitute a free monoid with $\{\varepsilon\}$ as unit [Pin84], every nonempty prefix code L admits a unique decomposition into prime prefix codes, $L = P_1 P_2 \ldots P_n$. By $\mathcal{P}(L)$ we will denote the set of all primes involved in the decomposition of L.

Lemma 1. *For any nonempty prefix code L, the set of primes $\mathcal{P}(L)$ is finite.*

Proof. Since the length of the prime decomposition of a nonempty prefix code L is not bigger than the length of the shortest word in L, $\mathcal{P}(L)$ is finite. $\square$

Given a language $L \subseteq \Sigma^*$, we define the following family of languages

$$Q(L) \stackrel{\text{def}}{=} \{w^{-1}L \mid w \in \Sigma^*\} \setminus \emptyset.$$

Elements of $Q(L)$ are called *quotients* of L. Notice that $Q(L)$ corresponds to the set of the states of the minimal deterministic (possibly infinite) automaton for L, with the initial state being L, final states being all those languages in $Q(L)$ which contain the empty word ε, and with the direct transition function $(Q(L) \times \Sigma) \mapsto (Q(L) \cup \emptyset)$ defined by $(K, a) \mapsto a^{-1}K$.

Let $\mathcal{L}$ be a family of prefix codes. We define $\mathcal{P}(\mathcal{L})$ as the set of primes involved in the decompositions of elements of $\mathcal{L}$:

$$\mathcal{P}(\mathcal{L}) \stackrel{\text{def}}{=} \bigcup_{L \in \mathcal{L}} \mathcal{P}(L).$$

A directed graph (digraph) with a sink is a triple $H = (V, E, s)$, where V is a (possibly infinite) set of vertices, $E \subseteq V \times V$ is a set of edges, and $s \in V$ is a special vertex called *sink*. A path π in H from a vertex u to u' is a finite sequence of vertices $\pi = v_1 v_2 \dots v_n$ such that $u = v_1$, $u' = v_n$, and $(v_i, v_{i+1}) \in E$ for every $i \in [1, n)$.

A vertex v is called a *D-articulation point* of a vertex u in H, if and only if v is present on every path from u to the sink. It was shown in [CFPR03] that the order of first (or last) occurrences of the D-articulation points of a vertex v is the same in all paths from v to s. Thus, it is natural to represent the set of all D-articulation points for a given vertex v as an ordered list $\delta_H(v)$ of vertices, $\delta_H(v) = (u_0, u_1, \dots, u_n)$, with $u_0 = v$ and $u_n = s$. The vertex u_1, if it exists, is called *next D-articulation point* of v, and it will be denoted by $P_H(v)$. If $\delta_H(v_1) = (v_1, v_2, \dots, v_n)$ then $P_H(v_i) = v_{i+1}$ and $\delta_H(v_i) = (v_i, \dots, v_n)$, for every $i \in [1, n)$. Thus, P_H (seen as a parent relation) defines a tree, called *D-articulation tree*, over vertices V which are co-accessible from the sink.

As noticed in [Eil74], a language L over an alphabet Σ is a prefix code if and only if its minimal deterministic (possibly infinite) automaton has a single final state without outgoing transitions.

In [CFPR03], it was shown that a prefix code L is a prime prefix code if and only if the next D-articulation point of L in $H = (Q(L), \{(F, a^{-1}F) \mid F \in Q(L), a \in \Sigma\}, \{\varepsilon\})$ is the sink, i.e., $P_H(L) = \{\varepsilon\}$. Moreover, the list of D-articulation points $\delta_H(L) = (v_1, v_2, \dots, v_n)$ corresponds to the prime decomposition of the prefix code L, the factors being the languages defined by automata starting in v_i with v_{i+1} as the final state (with all outgoing transitions of the final state removed), for $i \in [1, n)$, respectively. Even though this result has been stated only in the context of regular languages, it is valid for any prefix code. The proof given in [CFPR03] applies for any prefix code, not only regular prefix codes.

3 A characterization of s-languages

Lemma 2. *If L is an s-language then $\mathcal{P}(Q(L))$ is finite.*

Proof. Since $L \subset \Sigma^*$ is an s-language then there is an s-grammar $G = (\Sigma, V, P)$ such that $G(x) = L$ for some $x \in V$.

For every $w \in \Sigma^*$ which is a prefix of a word from L there exists a unique left-most derivation in G such that $x \xrightarrow{*} w\alpha$ with $\alpha = y_0 y_1 \dots y_n \in V^*$. Since $w^{-1}L = G(\alpha) = G(y_0)G(y_1)\dots G(y_n)$, $\mathcal{P}(w^{-1}L) \subseteq \bigcup_{y \in V} \mathcal{P}(G(y))$. Hence, $\mathcal{P}(Q(L)) \subseteq \bigcup_{y \in V} \mathcal{P}(G(y))$ is a finite set of primes. $\square$

Lemma 3. *Let $L \subseteq \Sigma^*$ be a prefix code. If $\mathcal{P}(Q(L))$ is finite then L is an s-language.*

Proof. The direct constructive proof consists in building an s-grammar $G = (\Sigma, V, P)$ for L. We define G as follows:

- $V = \{x_p \mid p \in \mathcal{P}(Q(L))\} \cup \{x_L\}$, i.e., we introduce a nonterminal for every prime p in $\mathcal{P}(Q(L))$ and, if L is not a prime, we add a starting nonterminal x_L.
- For every pair $(x_p, a) \in V \times \Sigma$ such that $a^{-1}p$ is notempty, we introduce a new rule $(x_p \rightarrow a x_{p_1} x_{p_2} \dots x_{p_n})$, where $p_1, p_2, \dots, p_n \in \mathcal{P}(Q(L))$ and $a^{-1}p = p_1 p_2 \dots p_n$.

Grammar G is an s-grammar and $G(x_L) = L$. □

Lemmas 2 and 3 imply the following theorem.

Theorem 4. *For any prefix code L the following properties are equivalent:*

1. *L is an s-language*
2. *$\mathcal{P}(Q(L))$ is a finite set*
3. *$Q(L) \subseteq \mathcal{S}^*$ for some finite set $\mathcal{S}$ of prefix codes*

4 Practical applications of the theorem

In [KH66] it was shown that, in general, it is undecidable to determine whether a language is an s-language. The process of proving that a language L is an s-language seems straightforward: one has to find an s-grammar for L. However, it was not clear how to proceed in the other direction. For example, one could prove that L is not an s-language by contradiction, firstly by asserting some properties of the language, secondly by showing that no s-grammar (or no deterministic simple automaton) can produce a language that fulfills these properties. Such an approach is far from being obvious and seems to be error prone (e.g., erroneous proof in [ABB97, Fact 5.2, page 148]).

In this section we will present some practical applications of our result for deciding whether a language is an s-language.

4.1 Proving that a language is not an s-language

Due to Theorem 4, determining if L is an s-language is equivalent to determining if $\mathcal{P}(Q(L))$ is finite. A simple direct proof that L is not an s-language would be to show that $Q(L)$ contains an infinite number of primes.

Example 1

Let us consider the language L_1 generated by the following grammar:

S $\rightarrow$ XZ
X $\rightarrow$ aXY | ε
Y $\rightarrow$ bX | c
Z $\rightarrow$ d

The language was introduced in [KS69] in order to demonstrate that the class of s-languages is a strict subset of LL(1) languages. The author proves that L_1 is not an s-language by contradiction, firstly by asserting some properties of the language, secondly by showing that no s-grammar can produce a language that fulfills these properties.

Using Theorem 4, one can prove that L_1 is not an s-language by showing that $Q(L_1)$ contains an infinite number of primes; actually every element in $Q(L_1)$ is prime. The deterministic minimal automaton for this language is depicted in Figure 1.

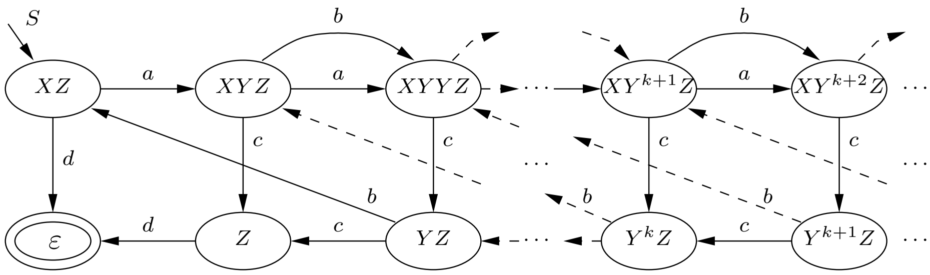


Figure 1: Minimal deterministic automaton for L_1

It is indeed minimal, because all states correspond to different languages: the length of the shortest word in states XY^kZ and Y^kZ is $(k+1)$ and $a^{-1}XY^kZ \neq a^{-1}Y^kZ$.

Every state of the automaton defines a prime prefix code because its next D-articulation point is the final state. While the number of states of the automaton is infinite, the language is not an s-language.

Example 2

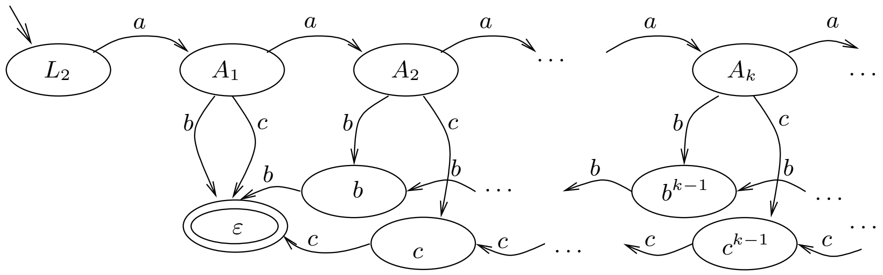
Let us consider the language $L_2 = \{a^nb^n \mid n \geq 1\} \cup \{a^nc^n \mid n \geq 1\}$.

In [KH66], the authors show that s-languages are not closed under intersection. They give L_2 as an example of a language that is an intersection of two s-languages[1] but is not an s-language. However, an explicit proof that L_2 is not an s-language is not given. We shall provide it here.

The minimal deterministic automaton for L is depicted in Figure 2. It is infinite, as L_2 is not regular.

We show that all the states A_k for $k \geq 1$, are different. The length of the shortest word from A_k is k, so for all $i \neq j$ there is $A_i \neq A_j$. The next D-articulation point of A_k, for $k \geq 1$, is $\{\varepsilon\}$, so A_k is prime. Hence, according to Theorem 4, L_2 is not an s-language.

[1] In [KH66] authors introduce $L_2\$ = \{a^n(b \cup c)^n\$ \mid n \geq 1\} \cap \{a^*b^*\$ \cup a^*c^*\$\}$. As L_2 is a prefix code, L_2 is an s-language if and only if $L_2\$$ is an s-language.



$A_k = \{a^n b^{n+k} \mid n \geq 0\} \cup \{a^n c^{n+k} \mid n \geq 0\}$, for $k \geq 1$.

Figure 2: Deterministic Minimal Automaton for L_2

4.2 Proving that a language is an s-language

When L is an s-language, the construction described in the proof of Lemma 3 may be used in order to rediscover an s-grammar for L. The construction is straightforward once we have the D-articulation tree of the minimal deterministic automaton for L.

Example 3

Language L_3 generated by the following grammar is an s-language.

S $\rightarrow$ ZW
Z $\rightarrow$ cX | aXZ | bZYZ
Y$\rightarrow$ aX | bZY
X$\rightarrow$ d | ε
W$\rightarrow$ e

It is not obvious how to find an s-grammar for L_3. However, an s-grammar for L_3 can be obtained from the D-articulation tree for its minimal deterministic automaton. The D-articulation tree is depicted in Figure 3. The primes generated by parts of the graph from vertices to their D-articulation points are repeated many times. In fact we can see that there are not more than 6 different primes involved: $\alpha, \beta, \gamma, \delta, \psi, \phi$.

We can read an s-grammar describing L_3 directly from the D-articulation tree:

S $\rightarrow$ c α | a $\beta\alpha$ | b $\delta\gamma\beta\alpha$
$\alpha \rightarrow$ e | d ψ
$\psi \rightarrow$ e
$\delta \rightarrow$ c | a β | b $\delta\gamma\beta$
$\beta \rightarrow$ c | a β | d δ | b $\delta\gamma\beta$
$\gamma \rightarrow$ a | d ϕ | b $\delta\gamma$
$\phi \rightarrow$ a | b $\delta\gamma$

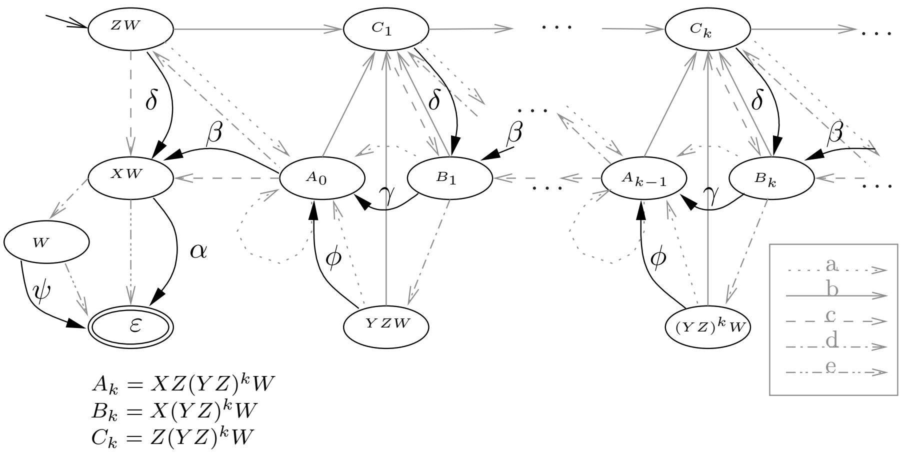


Figure 3: D-articulation tree for L_3.

Example 4

The same technique can be used to prove that language $L_4 = \{a^nba^n \mid n \geq 1\}$ is indeed an s-language, correcting an error in [ABB97, Fact 5.2, page 148], where the authors state otherwise.

As we can observe from the D-articulation tree for L_4 (Figure 4) there are only two primes involved, α and β. Hence $Q(L_4) \subseteq \{\alpha, \beta\}^*$, and thus L_4 is an s-language.

We can read the s-grammar for L_4 directly from the automaton as follows:

S $\rightarrow a\beta\alpha$
$\alpha \rightarrow a$
$\beta \rightarrow a\beta\alpha \mid b$

5 Conclusion

We have proved that a language L is an s-language if and only if the set $Q(L)$ of its quotients is a subset of a free monoid generated by a finite set $\mathcal{S}$ of prefix codes, i.e., $Q(L) \subseteq \mathcal{S}^*$. Such a characterization of the family of s-languages gives us a powerful tool for finding whether a given language is an s-language. A direct proof that a language L is not an s-language consists of showing that the set of prime prefix codes involved in the prime decomposition of elements of $Q(L)$ is infinite. Moreover, our characterization of s-languages yields (through the construction presented in the proof of Lemma 3) a method for rediscovering s-grammars.

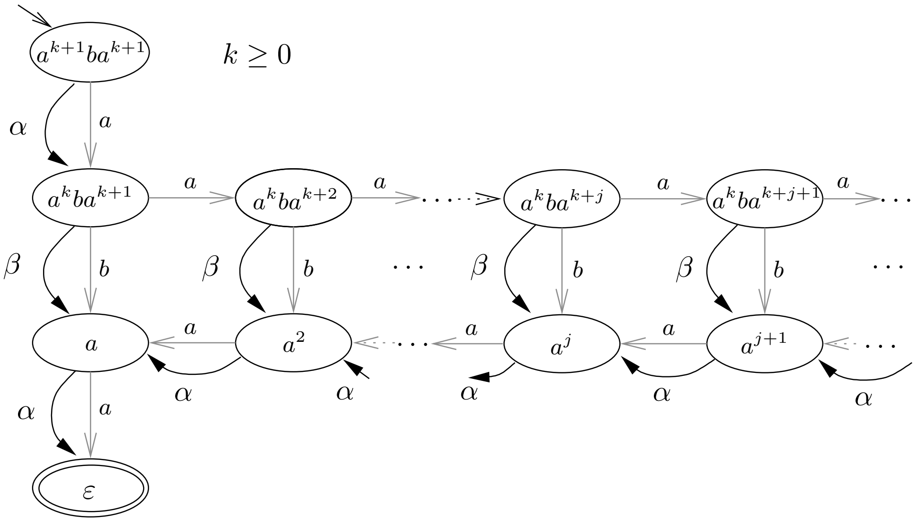


Figure 4: D-articulation tree for L_4.

References

[ABB97] Jean-Michel Autebert, Jean Berstel, and Luc Boasson. Context-free languages and pushdown automata. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 1, Word Language Grammar, pages 111–174. Springer-Verlag, Berlin, 1997.

[CFPR03] Jurek Czyzowicz, Wojciech Fraczak, Andrzej Pelc, and Wojciech Rytter. Prime decompositions of regular prefix codes. In *Implementation and Application of Automata, CIAA 2002*, volume 2608 of *LNCS*, pages 85–94. Springer, Tours, France, July 2003.

[Eil74] Samuel Eilenberg. *Automata, Languages, and Machines*, volume A. Academic Press, 1974.

[KH66] A. J. Korenjak and J. E. Hopcroft. Simple deterministic languages. In *Proc. IEEE 7th Annual Symposium on Switching and Automata Theory*, IEEE Symposium on Foundations of Computer Science, pages 36–46, 1966.

[KS69] Renio Kurki-Suonio. Notes on top-down languages. *BIT*, 9:225–238, 1969.

[Pin84] Jean-Eric Pin. *Variétés de langages formels*. Masson, Paris, 1984.