

Regular Languages and Associative Language Descriptions¹

Marcella Anselmo	Alessandra Cherubini	Pierluigi San Pietro
Dip. Informatica ed Appl.	Dip. di Matematica	Dip. di Elettronica e Inform.
Università di Salerno	Politecnico di Milano	Politecnico di Milano
84081 Baronissi	P.za Leonardo da Vinci 32	P.za Leonardo da Vinci 32
(Salerno), Italia	20133 Milano, Italia	20133 Milano, Italia
anselmo@dia.unisa.it	aleche@mate.polimi.it	

Abstract

The Associative Language Description model (*ALD*) is a combination of locally testable and constituent structure ideas. It is consistent with current views on brain organization and can rather conveniently describe typical technical languages such as Pascal or HTML. *ALD* languages are strictly enclosed in context-free languages but in practice the *ALD* model equals *CF* grammars in explanatory adequacy. Various properties of *ALD* have been investigated, but many theoretical questions are still open. For instance, it is unknown, at the present, whether the *ALD* family includes the regular languages. Here it is proved that several different classes of regular languages are *ALD*.

1 Introduction

The Associative Language Description model [4] (*ALD*) was originally motivated by the want of a brain compatible theory of language. In essence, this definition combines the concepts of local testability and of phrase structure in as simple a way as possible, aligned with current views on information processing in the brain [3]. A similar model was already proposed in [9]. The *ALD* definition introduces a solution to a few shortcomings of Context-Free (*CF*) grammars, disallowing certain “unpractical” languages (such as counting context-free languages, where derivation trees are characterized by numerical congruences) and not requiring metasymbols (the nonterminals) which are “external” to the language. Many basic properties of the *ALD* model were established in [6] and in [5], such as: nonclosure under union, concatenation and homomorphism; strict inclusion in the *CF* family (only noncounting context-free languages are *ALD*); comparison with other families; hierarchy theorems; etc. The expressive adequacy of the *ALD* family in terms of common artificial languages such as Pascal and HTML has been shown in [7].

The main open problem is the inclusion of regular languages in the *ALD* family. The lack of closure properties does not allow the use of standard ways to prove the (non)inclusion of regular languages in other families of languages, while many “ad hoc” methods enable to prove the inclusion of several classes of regular languages in *ALD* family, but do not generalize to all regular languages.

The objective of this paper is to prove some inclusion results of subfamilies of regular languages into *ALD*. More in details, we show that regular threshold locally testable languages [2], regular

¹Work partially supported by Miur: Progetto Cofinanziato “Linguaggi Formali e Automi: Metodi, Modelli e Applicazioni” and Progetto FIRB “Applicazioni della Teoria degli Automi all’Analisi, alla Compilazione e alla Verifica di Software Critico e in Tempo Reale”, and by Progetto Inter-gruppi INDAM: “Aspetti analitici, geometrici e combinatorici dei sistemi dinamici e dei linguaggi formali”.

commutative languages on an alphabet of at most two elements and regular group languages are all *ALD* languages. Moreover, there is an *ALD* language in each level of the star hierarchy. All these results seem to show that *ALD* languages are "well-distributed" over the class of regular languages. These considerations, together with many efforts towards finding a negative answer to the question, seem to support the conjecture that every regular language is *ALD*. Unfortunately, the proof techniques applied to show that each of these families are *ALD* are very varied and do not seem to be generalizable to every regular language.

Section 2 recalls the basic definitions and some relevant properties of the model, while Section 3 shows the main results of the paper. Section 4 draws a few conclusions.

2 Basic definitions and properties

Let Σ be a finite alphabet, and let $\Delta \notin \Sigma$ be the *placeholder*. A *stencil tree* is a tree T such that: the internal nodes of T are labeled by Δ and the leaves of T have labels in $\Sigma \cup \{\epsilon\}$. The *constituents* of a stencil tree are its subtrees of height one and leaves with labels in $\Sigma \cup \{\epsilon\} \cup \{\Delta\}$. The *frontier* of a stencil tree T or of a constituent K is denoted, respectively, by $\tau(T)$ and $\tau(K)$. A *maximal subtree* of T is a subtree of T whose leaves are also leaves of T .

Definition 2.1 (Left and right contexts)

Let T be a stencil tree. For an internal node i of T , let K_i and T_i be respectively the constituent and the maximal subtree of T having root i . Consider the tree T' obtained by excising the subtree T_i from T , leaving only the root labeled Δ of T_i behind. Let $s, t \in \Sigma^*$ be two words such $\tau(T') = s\Delta t$. The left context of K_i in T and of T_i in T is $\text{left}(K_i, T) = \text{left}(T_i, T) = s$; the right context of K_i in T and of T_i in T is $\text{right}(K_i, T) = \text{right}(T_i, T) = t$.

Definition 2.2 (ALD, pattern, permissible contexts of a rule) Let $\perp \notin \Sigma$ be the left/right terminator. An *Associative Language Description (ALD)* A is a finite collection of triples (x, z, y) , called rules, where $x \in (\epsilon \cup \perp)\Sigma^*$, $y \in \Sigma^*(\perp \cup \epsilon)$, and $z \in (\Sigma \cup \{\Delta\})^* - \{\Delta\}$. The word z is called the pattern of the rule (x, z, y) and the words x and y are called the permissible left/right contexts.

Definition 2.3 (Constituent matched by a rule, valid stencil tree) Let A be an ALD. A constituent K_i of a stencil tree T is matched by a rule (x, z, y) of an ALD A iff: 1) $z = \tau(K_i)$, 2) x is a suffix of $\perp \text{left}(K_i, T)$, and 3) y is a prefix of $\text{right}(K_i, T) \perp$. A stencil tree T is valid for A iff each constituent K_i of T is matched by a rule of A .

Definition 2.4 (Tree language and word language of an ALD) The (stencil) tree language defined by an ALD A , denoted by $\mathcal{T}(A)$, is the set of all stencil trees valid for A . The (word) language defined by an ALD A , denoted by $L(A)$, is the set $\{x \in \Sigma^* \mid x = \tau(T) \text{ for some tree } T \in \mathcal{T}(A)\}$.

Note that when a left/right context is irrelevant for a pattern, it is represented by the empty word ϵ or omitted. Given words v, w and finite sets X, Y, Z of words, the notation (X, vZw, Y) denotes the set of rules: $\{(x, vzw, y) \mid x \in X, y \in Y, z \in Z\}$. Also, the new symbol Λ may be used to denote the optionality of one occurrence of Δ , that is to merge two rules $(x, z'\Delta z'', y)$ and $(x, z'\Lambda z'', y)$ into the rule $(x, z'\Lambda z'', y)$.

An ALD is a device for defining a set of stencil trees and a word language, corresponding to their frontiers by means of a test; hence, it is not a generative grammar. A simple example is the finite language ab , defined by the ALD: $(\perp, \Delta\Delta, \perp), (\perp, a, b), (a, b, \perp)$. Clearly, the two insertions

must be done in parallel, because they interlock. On the other hand the semicontextual grammar $\perp b \rightarrow \perp ab, a \perp \rightarrow ab \perp$, with initial set $\perp \perp$ generates nothing.

We give here a more complex example of a regular *ALD*. Many other examples of regular and non regular languages belonging to *ALD* family as well as of *CF* languages which are not *ALD* can be found in [6] and in [5].

Example 2.5 Consider the (locally-testable) language L on the alphabet $\Sigma = \{a, b\}$: $L = \{w \in \Sigma^* \mid \text{if there is an occurrence of the subword } aa \text{ then there is no occurrence of the subword } bb\}$. An *ALD* A for L is:

$$\{(\perp, \Sigma\Lambda, \perp), (a, a\Lambda\Sigma, \perp), (a, a\Lambda, \Sigma \perp), (a, b\Delta, \epsilon), (a, b, a)(b, a\Lambda, \epsilon), (b, b\Lambda\{\Sigma^2 - aa\}, \perp), (b, b\Lambda, \Sigma^2 \perp)\}$$

To see that A defines exactly L , consider any constituent K of valid stencil tree T for A . Then the rules of A are such that: if $\text{right}(K, T)$ is \perp then neither aa nor bb occur in $\text{left}(K, T)$; if $\text{right}(K, T)$ is either $a \perp$ or $b \perp$ then aa is a subword of $\text{left}(K, T)$; similarly, if $\text{right}(K, T)$ is $ab \perp, ba \perp, bb \perp$ then bb is a subword of $\text{left}(K, T)$. The rules do not allow the case of a left context of K with both aa or bb , and so define exactly L (taking care also of the border cases).

ALD languages have the following properties:

- Every *ALD* language is a *CF*-language [5].
- The *ALD* family is not closed under any of the following operations [5]: union; (alphabetic, nonerasing) homomorphism; concatenation; Kleene star; intersection with regular languages; complementation; inverse alphabetic homomorphism.
- both the degree (i.e., the length of the largest permissible context) and the width (i.e., the length of the longest pattern) classify the *ALD* family in an infinite, strict hierarchy.
- Every context-free language is the homomorphic image of an *ALD* language [6].
- Every 1-letter regular language is *ALD* [6].
- All regular languages of restricted star-height one are *ALD* [1].

3 Relations between regular languages and the *ALD* family

The nonclosure properties presented in Section 2 have been proved in [5] using nonregular *ALD* languages, and therefore do not help in deciding whether some regular language is *ALD*. Also the obvious analogy between the *ALD* model and 1-variable *CF* grammars (definable by *ALDs* where all contexts are empty) do not help, even though regular languages are not included in the family of 1-variable *CF* languages. In fact, all languages of the form Ra , where R is a regular language on an alphabet Σ and $a \notin \Sigma$ are not 1-variable *CF* languages, but they are indeed *ALD* provided that R is an *ALD* language (if A is an *ALD* for R , an *ALD* for Ra is easily obtained by A transforming all the rules whose right context is \perp in rules whose right context is $a \perp$ and adding the rule $(\perp, \Delta a, \perp)$ to A if the empty word does not belong to R , the rule $(\perp, \Lambda a, \perp)$ otherwise). Indeed, at present we do not know any regular language for which we can not construct an *ALD*. Moreover, the only known general tool to prove that a language is not *ALD* is given by a Swapping Lemma [5], which allows the exchange of two subtrees of an *ALD* tree, provided they have similar profiles. Unfortunately, this lemma, similar in application to the traditional pumping lemmata of *CF* languages, appears to be useful only to prove that certain *nonregular* languages are not *ALD*.

In the following, some classes of regular *ALD* are presented.

3.1 Regular threshold locally testable languages

Let Σ be a finite alphabet. Denote with $\Sigma^{<h}$ the set $\epsilon \cup \Sigma \cup \Sigma^2 \cup \dots \cup \Sigma^{h-1}$. For every word $x, w \in \Sigma^+$ let $|w|_x$ be the number of occurrences of x as a factor of w and let $\mathcal{F}(\sqsubseteq)$ be the set of factors of v . For every integer h , let $F(x, h) = \{w \in \Sigma^* \mid |w|_x \geq h \geq 0\}$. For instance, $F(x, 1)$ is the language $\Sigma^* x \Sigma^*$ and its complement $\overline{F(x, 1)}$ is the set of words where x is a forbidden factor. Let $A \subseteq \Sigma^+$ be a finite set. A regular language L is said to be *threshold locally testable* (tlt [2]) if it is in the boolean algebra generated by languages of the form $\Sigma^* x, x \Sigma^*, \{x\}, F(x, h)$, with $x \in \Sigma^+, h \geq 0$. A locally testable language is a special case of tlt where each language of the form $F(x, h)$ has $h \leq 1$.

To prepare the proof of the next theorem, we need a few more definitions. Each tlt L may be described as a boolean composition \mathcal{B} of languages of the form $\Sigma^* x, x \Sigma^*, \{x\}, F(x, h)$. Hence, there is a finite set $\mathcal{G}_L \subseteq (\perp \cup \Sigma)^+$ such that: $\mathcal{G}_L = \{x \perp \mid \Sigma^* x \in \mathcal{B}\} \cup \{\perp x \mid x \Sigma^* \in \mathcal{B}\} \cup \{\perp x \perp \mid \{x\} \in \mathcal{B}\} \cup \{x \mid \exists h \geq 0 : F(x, h) \in \mathcal{B}\}$. We also extend the notion of stencil tree to a *generalized stencil tree*: a tree α is a generalized stencil tree if the internal nodes of α are labeled by Δ and the leaves of α have labels in $\Sigma \cup \{\epsilon, \Delta\}$ (i.e., Δ is allowed on the frontier of α). The concepts of constituents and validity can be immediately extended to generalized stencil trees. The generalized tree language $G_{\mathcal{T}}(A)$ of an *ALD* A is the set of generalized stencil trees valid for A . Notice that the word language $L(A) = \{x \in \Sigma^* \mid x = \tau(\alpha) \text{ for some tree } \alpha \in G_{\mathcal{T}}(A)\}$.

Proposition 3.1 *Each regular threshold locally testable language is an ALD language.*

PROOF. Let L be a tlt, with $\mathcal{B}, \mathcal{G}_L$ defined as above. Let $n = \max(h \mid \exists x \in \Sigma^+ : F(x, h) \in \mathcal{B})$, and let k be the length of the largest word in \mathcal{G}_L . To show that there exists an *ALD* Q such that $L(Q) = L$, we first define an enumeration function of factors in a word. Let $g : \mathcal{G}_L \rightarrow \mathcal{P}$, where \mathcal{P} is the set of prime numbers, and g is one-to-one. We extend g to every x in the set $(\perp \cup \Sigma)^*$, by defining $g(x) = 1$ for each $x \notin \mathcal{G}_L$. Define \bar{g} to count all relevant factors of each word $v \in \Sigma^*$ as follows: $\bar{g}(\epsilon) = 0$ and for $v \neq \epsilon$:

$$\bar{g}(v) = g(\perp v \perp) \prod_{w \in \perp \Sigma^* \cap \mathcal{G}_L} g(w) \prod_{w \in \perp \Sigma^* \cap \mathcal{G}_L} g(w)^{|v|_w} \prod_{w \in \Sigma^* \perp \cap \mathcal{G}_L} g(w) \quad (1)$$

As an example, consider $\mathcal{G}_L = \{\perp a, \perp b, aa, ab, ba, abab, abba, b \perp, \perp aba \perp\}$, with $n = 2$. An enumeration g is $g(\perp a) = 2, g(\perp b) = 3, g(ab) = 5, g(ba) = 7, g(aa) = 11, g(b \perp) = 13, g(\perp aba \perp) = 17, g(abab) = 19, g(abba) = 23$. Then, $\bar{g}(aabab) = 2 \times 3 \times 5^2 \times 7 \times 11 \times 13 \times 19 = \bar{g}(aabaab)$.

Last, given the decomposition $p_1^{e_1} p_2^{e_2} \dots, p_i \in \mathcal{P}, e_i \geq 0$ in prime factors of an integer $q > 0$, and an integer $j \geq 0$, define $[q]_j = p_1^{\min(j, e_1)} p_2^{\min(j, e_2)} \dots$. The following properties of \bar{g} are immediate, if $n \gg k$, for all $x, y, z \in \Sigma^+$:

$$|y| \geq k \Rightarrow g(xyz) = g(xy) \cdot g(yz) / g(y) \quad (2)$$

$$[[\bar{g}(x)]_n \cdot [\bar{g}(z)]_n]_n = [\bar{g}(x) \cdot \bar{g}(z)]_n \quad (3)$$

$$|y| \leq k \Rightarrow [\bar{g}(y)]_n = \bar{g}(y) \quad (4)$$

Let $M = \{m \geq 0 \mid \exists v \in \Sigma^+ : v \in L \wedge m = [\bar{g}(v)]_n\}$. We claim that $v \in L$ iff $[\bar{g}(v)]_n \in M$ (with M finite). The only if" part is obvious from the definition of M , while the if" part follows from the fact that L is a tlt: L is testable by checking a boolean expression, whose atoms can be checked for truth by looking at the encoding $[\bar{g}(v)]_n$ (each $F(x, h)$ can be checked since $h \leq n$).

Let $Q_1 = \{(\perp, x, \perp) \mid x \in \Sigma^*, |x| \leq k + \max(M), x \in L\}$. Q_1 is the ALD defining exactly all the (finite number of) words of L of length not greater than $k + \max(M)$ (thus, Q_1 can always be constructed by checking all words of length up to $k + \max(M)$).

Let $Q_2 = \{(y, a\Delta x, z \perp) \mid y \in \Sigma^k \cup \perp \Sigma^{<k}, a \in \Sigma, x \in \Sigma^*, z \in \Sigma^+, |x| = [|z| \cdot \bar{g}(ya)/\bar{g}(y)]_n - |z|\}$. We claim that: (*) if a generalized stencil tree α is valid for Q_2 (i.e., $\alpha \in G_{\mathcal{T}}(Q_2)$), then $\exists v, z \in \Sigma^* : \tau(\alpha) = v\Delta z, [\bar{g}(v)]_n = |z|$. The proof of (*) is by induction on the length i of the frontier of a (obviously linear) generalized stencil tree α valid for Q_2 . The base case $i = 0$ is obvious, since in this case (corresponding to the trivial tree with only one node, labeled Δ) $z = v = \epsilon$ and $\bar{g}(z) = 0$. Let $\tau(\alpha) = v\Delta z, |v\Delta z| = i$. We want to show that, for every $a \in \Sigma$ and for every $x \in \Sigma^*, \exists \beta \in G_{\mathcal{T}}(Q_2)$ whose frontier is $va\Delta xz$, with $|x| = [\bar{g}(ya) \cdot |z|]_n - |z|$, for some $y \in \Sigma^k \cup \perp \Sigma^{<k}$ that is a suffix of $\perp v$. Let $\perp v = v_1 y$ for some $v_1 \in \Sigma^*$. Hence, by (2), (3) and (4), $[\bar{g}(v_1 ya)]_n = [[\bar{g}(v)]_n \cdot \bar{g}(ya)/\bar{g}(y)]_n$. By induction hypothesis, $[\bar{g}(v)]_n = |z|$. But, by definition of $Q_2, |xz| = |x| + |z| = [|z| \cdot \bar{g}(ya)/\bar{g}(y)]_n = [[\bar{g}(v)]_n \cdot \bar{g}(ya)/\bar{g}(y)]_n = [\bar{g}(va)]_n$ (since y is also a suffix of v).

We also claim that: (+) for every $v, z \in \Sigma^*$ such that $[\bar{g}(v)]_n = |z|$, there exists a generalized stencil tree $\alpha \in G_{\mathcal{T}}(Q_2)$ such that $\tau(\alpha) = v\Delta z$. The proof of (+) is by induction on the length of v : if $v = \epsilon$, then also $z = \epsilon$, and the trivial tree with only one node, labeled Δ is in $G_{\mathcal{T}}(Q_2)$. If $|v| > 0$ then let $v = w'a, w' \in \Sigma^*, a \in \Sigma$. By induction hypothesis, there is a generalized stencil tree $G_{\mathcal{T}}(Q_2)$ with frontier $w'\Delta z'$ for every $z' \mid |z'| = [\bar{g}(w')]_n$. In particular, we can select z' to be a suffix of z ($|z'| = [\bar{g}(w')]_n \leq [\bar{g}(w'a)]_n = |z|$): $z = z''z'$ for one $z'' \in \Sigma^*$. If $|w'| < k$, let $w'' = \perp w'$, otherwise let w'' be the suffix of length k of w' . The claim is proved if $(w'', a\Delta z'', z') \in Q_2$. By definition of Q_2 , it is enough to show that $|z''| = [|z'| \cdot \bar{g}(w''a)/\bar{g}(w'')]_n - |z'|$, that is equivalent to show that $|z| = |z'| + |z''|$ must equal $[|z'| \cdot \bar{g}(w''a)/\bar{g}(w'')]_n$. By induction hypothesis, $|z'| = [\bar{g}(w')]_n$: $[|z'| \cdot \bar{g}(w''a)/\bar{g}(w'')]_n = [\bar{g}(w') \cdot \bar{g}(w''a)/\bar{g}(w'')]_n$, which by (2), is equal to $[\bar{g}(w'a)]_n = [\bar{g}(v)]_n$, which by hypothesis is exactly equal to $|z|$.

Let $Q_3 = \{(y, w, z \perp) \mid y \in \Sigma^k, z, w \in \Sigma^*, |wz| = \max(M) + 1, [|z| \cdot \bar{g}(y wz)/\bar{g}(y)]_n \in M\}$.

An *ALD* Q for L is $Q_1 \cup Q_2 \cup Q_3$. Suppose $v \in L$. If $|v| \leq k + \max(M)$ then $v \in L(Q_1) \subseteq L(Q)$. If $|v| > k + \max(M)$ then $\exists! x, x' \in \Sigma^+ \mid z = xx'$ and $|x'| = \max(M) + k$ (therefore, $|x| \geq k$). Hence, since $[\bar{g}(v)]_n \leq \max(M)$, $\exists w, z \in \Sigma^+$ such that $x' = wz$ and $|z| = [\bar{g}(v)]_n$. Q_2 defines, by the above property (+), also the valid generalized stencil tree with frontier $x\Delta z$. It enough to show that there is a rule in Q_3 of type $(y, w, z \perp)$, and $v = xwz \in L(Q)$. To this effect, the other conditions being obviously verified, we need to prove that $[|z| \cdot \bar{g}(y wz)/\bar{g}(y)]_n \in M$, with y being the suffix of length k of x ($|x| \geq k$). But $|z| = \bar{g}(x)$. Hence, $[|z| \cdot \bar{g}(y wz)/\bar{g}(y)]_n = [\bar{g}(x) \cdot \bar{g}(y wz)/\bar{g}(y)]_n$, which by (2) (y is a suffix of x) is equal to $[\bar{g}(xwz)]_n \in M$.

Suppose now that $v \in L(Q_2 \cup Q_3)$ (the case $v \in L(Q_1)$ is obvious). Then $v = xwz$, where $x, w, z \in \Sigma^*$ and $x\Delta z$ is the frontier of a generalized stencil tree valid for $Q_2 \subseteq Q$, with $(y, w, z \perp) \in Q_3$, for some suffix y of $\perp x$. Hence, and, by definition of $Q_3, |wz| = \max(M) + 1 > k, [|z| \cdot \bar{g}(y wz)/\bar{g}(y)]_n \in M$. Since, by the above property of $Q_2, [\bar{g}(x)]_n = |z|$, the last term is equal to $[[\bar{g}(x)]_n \cdot \bar{g}(y wz)/\bar{g}(y)]_n = [\bar{g}(x) \cdot \bar{g}(y wz)/\bar{g}(y)]_n$ which is equal (y being a suffix of x) to $[\bar{g}(xwz)]_n \in M$. By definition of $M, v = xwz \in L$. \square

3.2 Relations with the Star-height Hierarchy

To prove that there is an *ALD* language in each level of the star height hierarchy, we need to recall the following definitions from [5].

Definition 3.2 [5] (Contexts of the nonterminals of a *CF* grammar) *Let $G = (V_N, \Sigma, P, S)$ be a *CF* grammar and $k \geq 0$ be an integer. For every $X \in V_N$, $Con_k(X)$ is the set:*

$$\{(x, y) \mid (x, y) \in \Sigma^k \cup \perp \Sigma^{<k} \times \Sigma^k \cup \Sigma^{<k} \perp \wedge \exists u, v \in (\Sigma \cup \perp)^*, \exists z \in \Sigma^* : \perp S \perp \Rightarrow_G^* uxXyv \Rightarrow_G^* uxzyv\}$$

Definition 3.3 [5] (Operator form of CF grammars) Let $G = (V_N, \Sigma, P, S)$ be a CF grammar and $k \geq 0$ be an integer.

- G is said to be in operator form of length k iff for every $X, Y \in V_N$, for every $w \in \Sigma^*$, every production of P with right-hand side of the form $(\Sigma \cup V_N)^* XwY(\Sigma \cup V_N)^*$ is such that $|w| \geq k$.
- G is said to be in disjoint operator form of length k if, and only if:
 1. G is in operator form of length k ;
 2. $Con_k(X) \neq \emptyset$ for all $X \in V_N$;
 3. For all $X, Y \in V_N$, with $X \neq Y$, $Con_k(X) \cap Con_k(Y) = \emptyset$.

Proposition 3.4 There is an ALD language in each level of the star height hierarchy.

PROOF. Let consider for each $i \geq 0$ the language R_i given by a regular expression e_i defined recursively as follows:

- $e_0 = \epsilon$
- $e_{i+1} = (a^{2^i} e_i b^{2^i} e_i)^*$, $i \geq 0$.

The language R_i has star height i [10]. It is easy to prove that the grammar $G_i = (\{a, b\}, \{S_1, S_2, \dots, S_i\}, S_i, P_i)$ where P_i is composed of the following productions:

$$\begin{aligned} S_j &\rightarrow S_j a^{2^{j-1}} S_{j-1} b^{2^{j-1}} S_{j-1} \mid \epsilon, 1 < j \leq i \\ S_1 &\rightarrow S_1 a b \mid \epsilon \end{aligned}$$

is a context-free grammar in operator form of length 2 generating R_i for $i \geq 1$. Unfortunately, it is not in disjoint operator form. However it is possible to transform G_i into an equivalent grammar G'_i in disjoint operator form of length $2^{\sum_{0 \leq h < i} 2^h}$, by iterated expansions of non terminal letters in the right-hand side of every production. Then by lemma 3.28 of [5] there exists an ALD A_i structurally equivalent to G'_i . Remarking that the left contexts of A_i are different, A_i can be reduced to the following ALD A'_i with only left contexts:

$$\begin{aligned} &\{(\perp, \Lambda a^{2^{i-1}} \Lambda b^{2^{i-1}} \Lambda, \epsilon), (\perp, \epsilon, \epsilon)\} \\ \cup &\bigcup_{1 \leq h < i-1} \{(\perp a^{2^{i-1}} a^{2^{i-2}} \dots a^{2^{i-h}}, \Lambda a^{2^{i-h-1}} \Lambda b^{2^{i-h-1}} \Lambda, \epsilon)\} \\ \cup &\bigcup_{1 \leq h < i-1} \{(ba^{2^{i-1}} a^{2^{i-2}} \dots a^{2^{i-h}}, \Lambda a^{2^{i-h-1}} \Lambda b^{2^{i-h-1}} \Lambda, \epsilon)\} \\ \cup &\bigcup_{1 \leq h < i-1, 2^{i-h} \leq j < 2^{i-h+1}} \{(ab^j, \Lambda a^{2^{i-h-1}} \Lambda b^{2^{i-h-1}} \Lambda, \epsilon)\} \\ \cup &\{(\perp a^{2^{i-1}} a^{2^{i-2}} \dots a^2, \Lambda ab, \epsilon), (ab^3, \Lambda ab, \epsilon), (ab^2, \Lambda ab, \epsilon), (ba^2, \Lambda ab, \epsilon)\} \end{aligned}$$

which is an ALD for R_i . \square

3.3 Regular group languages

Let $L \subseteq \Sigma^*$ be a regular language. Let us denote: \equiv_L the syntactic congruence on L , $M(L)$ the syntactic monoid of L with identity 1, and $\text{repr}(w)$ the representative in $M(L)$ of $w \in \Sigma^*$. Further let us denote $\text{SYM}(L)$ the *automaton of transitions in $M(L)$* , that is the automaton whose graph is the Cayley graph of $M(L)$, the identity 1 is the initial state and the final states are those ones representing words in L . It is well-known that $\text{SYM}(L)$ recognizes L . We will say that a path (q_1, q_2, \dots, q_n) in a finite automaton has *loop-nesting degree* $d = 0$ if the path is simple (i.e., without loops) ($q_i \neq q_j$ for any $1 \leq i < j \leq n$); it has loop-nesting degree $d \geq 1$ if there exist $1 \leq i_1 < i_2 < \dots < i_{2k-1} < i_{2k} \leq i_n$ such that for any odd $h = 1, \dots, 2k-1$ we have that: $q_{i_h} = q_{i_{h+1}}$, the path $(q_1, \dots, q_{i_1}, q_{i_2+1}, \dots, q_{i_3}, q_{i_4+1}, \dots, q_{i_{2k-1}}, q_{i_{2k}+1}, \dots, q_n)$ has loop-nesting degree $d-1$ and the path $(q_{i_h}, q_{i_h+1}, \dots, q_{i_{h+1}})$ has no repetition of states unless for state q_{i_h} .

Proposition 3.5 *Each regular group language is an ALD language.*

PROOF. Let $L \subseteq \Sigma^*$ be a regular language whose syntactic monoid $M(L)$ is a group. Let y be the label of any loop on a state m in $\text{SYM}(L)$. Hence $my \equiv_L m$. Since $M(L)$ is a group, there exists $m' \in M(L)$ such that $m'm = 1$ and then $\text{repr}(y) = m'm \text{repr}(y) = m'm = 1$. This implies that any loop in $\text{SYM}(L)$ is also a loop on the initial state 1 and that for any $u, v \in \Sigma^*$ we have $uv \in L$ iff $uyv \in L$. An ALD $A(L)$ defining L can be therefore constructed as follows. Define for $w = a_1 \dots a_n$, with $a_1, \dots, a_n \in \Sigma$, $\Lambda(w) = \Lambda a_1 \Lambda \dots \Lambda a_n \Lambda$. The rules of $A(L)$ are the following: $\{(\perp, \Lambda(w), \perp) \mid w \in \text{SSP}\}$ and $\{(\epsilon, \Lambda(w), \epsilon) \mid w \in \text{SL}\}$, where SSP is the set of labels of all successful simple paths (a simple path from an initial to a final state) in $\text{SYM}(L)$, and SL is the set of labels of all simple loops on the initial state of $\text{SYM}(L)$.

We claim that a word (is in L iff it) is recognized by $\text{SYM}(L)$ iff it is defined by $A(L)$. A word w is recognized by $\text{SYM}(L)$ iff there is a successful path in $\text{SYM}(L)$ labelled w . We show by induction that there is a successful path labelled w in $\text{SYM}(L)$ of loop-nesting degree d iff there is a valid stencil tree for $A(L)$ with frontier $\Lambda(w)$ and height h with $h = d$. One has $d = 0$ iff the path is simple iff the rule $(\perp, \Lambda(w), \perp)$ is in $A(L)$, giving raise to a valid stencil tree of height $h = 0$ whose frontier is $\Lambda(w)$. Suppose now $d > 0$. According to the above definition, a successful path of loop-nesting degree $d > 0$ is a successful path $(q_1, \dots, q_{i_1}, q_{i_2+1}, \dots, q_{i_3}, q_{i_4+1}, \dots, q_{i_{2k-1}}, q_{i_{2k}+1}, \dots, q_n)$ having loop-nesting degree $d-1$ and label z , where some simple loops with labels y_1, \dots, y_l (note $l \geq h$) are inserted on some q_{i_1}, \dots, q_{i_h} . By induction, this holds iff there is a valid stencil tree for $A(L)$ of height $h-1 = d-1$ and frontier $\Lambda(z)$, and rules $(\epsilon, \Lambda(y_1), \epsilon), \dots, (\epsilon, \Lambda(y_l), \epsilon)$ are in $A(L)$. Finally this condition holds iff there is a valid stencil tree for $A(L)$ of height $h = d$ and frontier $\Lambda(w)$. \square

Example 3.6 *Let $L \subseteq \{a, b\}^*$ be the language recognized by the finite automaton $(\{1, 2, 3\}, \{a, b\}, \{1\}, \{1\}, \delta)$, where the transition function is given by: $\delta(1, a) = \delta(1, b) = 2$, $\delta(2, a) = 3$, $\delta(2, b) = 1$, $\delta(3, a) = 1$ and $\delta(3, b) = 3$. The syntactic monoid of L is $M(L) = \{1, a, b, a^2, ab, ba\}$ with relations $b^2 = 1$, $a^3 = 1$, $a^2b = ba$, $aba = b$, $ba^2 = ab$, and $bab = a^2$. We have that $M(L)$ is a group and the sets SSP and SL , as defined in the proof of Proposition 3.5, are the following: $\text{SSP} = \{1, ab, ba^2, a^2ba\}$, and $\text{SL} = \{b^2, a^3, abab, baba, baba^2, a^2ba^2b, aba^2ba\}$. According to Proposition 3.5, an ALD defining L is given by the following rules: $(\perp, \Lambda(1), \perp)$, $(\perp, \Lambda(ab), \perp)$, $(\perp, \Lambda(ba^2), \perp)$, $(\perp, \Lambda(a^2ba), \perp)$, and $(\epsilon, \Lambda(b^2), \epsilon)$, $(\epsilon, \Lambda(a^3), \epsilon)$, $(\epsilon, \Lambda(abab), \epsilon)$, $(\epsilon, \Lambda(baba), \epsilon)$, $(\epsilon, \Lambda(baba^2), \epsilon)$, $(\epsilon, \Lambda(a^2ba^2b), \epsilon)$, $(\epsilon, \Lambda(aba^2ba), \epsilon)$.*

3.4 Regular commutative languages on 2-letter alphabets

The family of regular commutative languages on an alphabet Σ is the boolean algebra generated by languages of the form $F(x, k) = \{u \in \Sigma^* \mid |u|_x \geq k\}$ or $F(x, k, n) = \{u \in \Sigma^* \mid |u|_x \equiv k \pmod{n}\}$ with $k < n$, for $x \in \Sigma$. Define $E(x, k) = \{u \in \Sigma^* \mid |u|_x = k\}$ and for $k \leq n$ define $F(x, k, n) = \{u \in \Sigma^* \mid |u|_x \equiv k \pmod{n}, |u|_x \geq k\}$. Then: $\overline{F(x, k)} = \bigcup_{0 \leq i < k} E(x, i)$, $\overline{F(x, k, n)} = \bigcup_{0 \leq i < n, i \neq k} F(x, k, n)$, and $F(x, k) = F(x, k, 1)$. Hence, each regular commutative language belongs to the positive boolean algebra generated by the languages of the form $E(x, k)$ or $F(x, k, n)$ (called in the sequel *literals* $L(x)$ on x) where $k \geq 0$, $n > 0$, $x \in \Sigma$. Moreover, for each $x \in \Sigma$, $L(x) = L(x) \cap F(y, 0, 1)$, $x \neq y$ and $\exists h, t > 0 : F(x, k, n) = \bigcup_{0 \leq i < t} F(x, k + in, hn)$ with t the least integer such that $k + tn \geq hn$.

Definition 3.7 A regular commutative language L on Σ is in normal form if: $\exists r > 0 : L = \bigcup_{1 \leq i \leq r} C_i$, where $C_i = \bigcap_{x \in \Sigma} L_i(x)$, with either $L_i(x) = E(x, k_{i,x})$ or $L_i(x) = F(x, k_{i,x}, n_x)$ for some nonnegative integer $k_{i,x}$ depending on i and x and a positive integer n_x only depending on x . The terms $C(i)$ are called normal terms of L .

Fact 3.8 Each regular commutative language can be represented in normal form.

Let L be a regular commutative language in normal form on an alphabet $\Sigma : L = \bigcup_{1 \leq i \leq r} C_i$ with $C_i = \bigcap_{x \in \Sigma} L_i(x)$. Let $L_x = \bigcup_{1 \leq i \leq r} L_i(x) = (\bigcup_{i \in I_x} E(x, k_{i,x})) \cup (\bigcup_{j \in J_x} F(x, k_{j,x}, n_x))$ be the union of all literals on x occurring in L , and let $K_x = \max_{i \in I_x} \{k_{i,x}\}$ (assume $K_x = 0$ if I_x is empty and $n_x = 1$ if J_x is empty). The Cayley graph $\Gamma(L_x)$ of the syntactic monoid of L_x has $K_x + n_x + 1$ vertices $X_l, 0 \leq l \leq K_x + n_x$, loops labelled by $y \in \Sigma - x$ are based at each vertex, an edge labelled by x goes from X_i to X_{i+1} for each $i, 0 \leq i \leq K_x + n_x - 1$ and an edge with label x from $X_{K_x+n_x}$ to X_{K_x+1} . The graph $\Gamma(L_x)$ has two parts: the first part contains the $K_x + 1$ vertices $X_l, 0 \leq l \leq K_x$, and no loops whose labels contain x are based at its vertices, the second part contains the n_x vertices $X_l, K_x + 1 \leq l \leq K_x + n_x$, a loop labelled by x^{n_x} is based at each vertex. $SYM(L_x)$ has $\Gamma(L_x)$ as underlying graph, initial state X_0 and final states $\{X_i \mid i \in I_x\} \cup \bigcup_{j \in J_x} \{X_j \mid j \equiv j \pmod{n_x} \wedge j \leq \tilde{j} \leq K_x + n_x\}$. Assume $\Sigma = \{a, b\}$, and consider $\Gamma = \Gamma(L_a) \times \Gamma(L_b)$. Γ is partitioned in four regions $\Gamma_h, 1 \leq h \leq 4$, which are so composed: in Γ_1 there are $(K_a + 1) \times (K_b + 1)$ vertices $V_{i,j}^1 = (A_i, B_j)$ with $0 \leq i \leq K_a, 0 \leq j \leq K_b$ and no loop are based on these vertices; in Γ_2 there are $n_a \times (K_b + 1)$ vertices $V_{i,j}^2 = (A_i, B_j)$ with $K_a + 1 \leq i \leq K_a + n_a, 0 \leq j \leq K_b$ and only loops labelled by a^{n_a} are based at each vertex; in Γ_3 there are $(K_a + 1) \times n_b$ vertices $V_{i,j}^3 = (A_i, B_j)$ with $0 \leq i \leq K_a, K_b + 1 \leq j \leq K_b + n_b$ and only loops labelled by b^{n_b} are based at each vertex; in Γ_4 there are $n_a \times n_b$ vertices $V_{i,j}^4 = (A_i, B_j)$ with $K_a + 1 \leq i \leq K_a + n_a, K_b + 1 \leq j \leq K_b + n_b$ and the same set of loops labelled by anagrams of $a^{tn_a} b^{sn_b}$, for some $t, s > 0$, is based at each vertex. The apex on vertex names indicates the region where the vertex is, and can be omitted.

Definition 3.9 Let L be a regular commutative language in normal form on $\Sigma = \{a, b\}$. The automaton $\mathcal{A} = \langle \Gamma, V(0, 0), F \rangle$, (with Γ the underlying graph, $V_{0,0}$ the initial state, F the set of final state) where a vertex $V_{h,k} \in F$ iff

- $C_i = L_i(a) \cap L_i(b)$ is a normal term of L for some i ,
- either $L_i(a) = E(a, h)$ or $L_i(a) = F(a, t, n_a)$ with $t \equiv h \pmod{n_a}$ and $t \leq h \leq K_a + n_a$
- and either $L_i(b) = E(b, k)$ or $L_i(b) = F(b, s, n_b)$ with $s \equiv k \pmod{n_b}$ and $s \leq k \leq K_b + n_b$

is called the normal automaton of L .

It is easy to verify that \mathcal{A} recognizes L . Moreover, for each $V_{i,j} \in F$ denote by $\mathcal{A}_{i,j}$ the automaton $\langle \Gamma, V(0,0), \{V_{i,j}\} \rangle$, whose unique final state is $V_{i,j}$. Then, the language $L(\mathcal{A}_{i,j})$ is a normal term of L ; conversely, each normal term of L is a language $L(\mathcal{A}_{i,j})$ for some $V_{i,j} \in F$. If $V_{i,j}$ is in the region Γ_k of Γ , then for shortness we call $L(\mathcal{A}_{i,j})$ a k -normal term of L .

For the following proofs, we introduce a few shorthands. For an automaton \mathcal{A} let $SSP(\mathcal{A})$ be the set of labels of all successful simple paths in \mathcal{A} , let $SSL(\mathcal{A})$ be the set of labels of all simple loops based on the initial vertex of \mathcal{A} containing both a and b , and let $SL(\mathcal{A})$ be the set of labels of all simple loops based on the initial vertex of \mathcal{A} .

Lemma 3.10 *Let L be a regular commutative language in normal form on $\Sigma = \{a, b\}$. Each k -normal term of L is an ALD language.*

PROOF. The proof of the lemma considers four cases:

Case 1: $k = 1$. Then $L(\mathcal{A}_{i,j})$ is finite, and it is obviously defined by the ALD $A_{i,j}^1$ whose rules are $\{(\perp, w, \perp) \mid w \in L(\mathcal{A}_{i,j})\}$.

Case 2: $k = 2$. It is easy to verify that the automaton $\mathcal{A}_{i,j}^2$, obtained by $\mathcal{A}_{i,j}$ adding an edge labelled a going from each vertex $V_{h,k}$, with $h \geq n_a - 1$, to the vertex $V_{h-n_a+1,k}$, recognizes $L(\mathcal{A}_{i,j})$. Then the ALD $A_{i,j}^2 = \{(\perp, \Lambda(w), \perp) \mid w \in SSP(\mathcal{A}_{i,j}^2)\} \cup \{(\epsilon, \Lambda a^{n_a}, \epsilon)\}$ defines $L(\mathcal{A}_{i,j})$. The proof is analogous to the proof of Proposition 3.5 because even if here the syntactic monoid is not a group, the language is recognized by an automaton where the same set of loops (labelled by a^{n_a}) is based on each state.

Case 3: $k = 3$. Denote with $\mathcal{A}_{i,j}^3$ the automaton obtained by $\mathcal{A}_{i,j}$ adding an edge labelled by b from each vertex $V_{h,k}$, with $h \geq n_b - 1$, to the vertex $V_{h,k-n_b+1}$. Analogously to Case 2, the ALD $A_{i,j}^3 = \{(\perp, \Lambda(w), \perp) \mid w \in SSP(\mathcal{A}_{i,j}^3)\} \cup \{(\epsilon, \Lambda b^{n_b}, \epsilon)\}$, defines $L(\mathcal{A}_{i,j})$.

Case 4: $k = 4$. Denote with $\mathcal{A}_{i,j}^4$ the automaton obtained by $\mathcal{A}_{i,j}$ adding an edge labelled by a from each vertex $V_{h,k}$, with $h \geq n_a - 1$, to the vertex $V_{h-n_a+1,k}$ and an edge labelled by b from each vertex $V_{h,k}$, with $h \geq n_b - 1$, to the vertex $V_{h,k-n_b+1}$. Again the same set of loops is based at each vertex of the automaton, so it is easy to prove that the ALD $A_{i,j}^4 = \{(\perp, \Lambda(w), \perp) \mid w \in SSP(\mathcal{A}_{i,j}^4)\}$ and $\{(\epsilon, \Lambda(w), \epsilon) \mid w \in SL(\mathcal{A}_{i,j}^4)\}$, defines $L(\mathcal{A}_{i,j})$. \square

Proposition 3.11 *A regular commutative language on an alphabet Σ with $|\Sigma| \leq 2$ is an ALD language.*

PROOF. In [5], it was proved that each unary regular language is an ALD language. Then, let L be a regular commutative language in normal form on $\Sigma = \{a, b\}$ and let $\mathcal{A} = \langle \Gamma, V_{0,0}, F \rangle$ be the normal automaton recognizing L . Obviously $L = \bigcup_{i,j:V_{i,j} \in F} L(\mathcal{A}_{i,j})$. Let $F_k = \{V_{i,j}^k \mid V_{i,j}^k \in F\}$. If $F = F_k$ the ALD $A = \bigcup_{i,j:V_{i,j} \in F} A_{i,j}^k$ defines L , since only the initial rules of each $A_{i,j}^k$ are different. Obviously, if $F = F_1 \cup F_k$ for $k = 2, 3, 4$ the ALD $A = \bigcup_{i,j:V_{i,j} \in F_1} A_{i,j}^1 \cup \bigcup_{i,j:V_{i,j} \in F_k} A_{i,j}^k$ also defines L , because each $A_{i,j}^k$ has only initial rules. In the other cases the union A of the ALDs defining the normal terms of L does not define L because the contexts of non initial rules of an ALD $A_{i,j}^k$, defining a k -normal term ($k \neq 1$) of L , cannot be distinguished by the context of non initial rules of an ALD $A_{i',j'}^{k'}$, defining a k' -normal term ($k' \neq 1, k$) of L : some words not belonging to L can be the frontier of a valid tree of the ALD A . Hence, we modify the ALDs defining k -normal terms of L and the ALDs defining k' -normal terms of L so that if $k \neq k'$ the corresponding non initial rules may have the same contexts only if they also have equal patterns. For each $u, v \in \Sigma^+$ denote $Sh(u, v) = \{u_0 v_1 u_1 \dots v_s u_s v_{s+1} \mid u = u_0 u_1 \dots u_s, v = v_1 v_2 \dots v_{s+1}, s \leq$

0, $u_0, v_{s+1} \in \Sigma^*$, $u_i, v_i \in \Sigma^+$ for $0 < i < s$).

We have four cases according to the value of k , but Case $k = 1$ does not require modifications to the *ALDs*.

Case $k = 2$: Let $L(\mathcal{A}^2)$ be a 2-normal term. For $w \in \Sigma^*$ define $EXP_2(w) = \Lambda(\{Sh(w, a^{hn_a}) \mid h \leq 2(|w| + 1)\}) \cap (b^* a^{n_a} a^* \Lambda b^*)^*$. Then the *ALD* \tilde{A}^2 with the following rules: $\{(\perp, EXP_2(w), \perp) \mid w \in SSP(\mathcal{A}^2)\} \cup \{(a^{n_a}, \Lambda a^{n_a}, \epsilon)\}$, defines $L(\mathcal{A}^2)$. In fact, the initial rules allow the introduction, by means of a valid tree of height 1, of all the words belonging to L in the form $a^{m_0} b^{n_1} a^{m_2} b^{n_2} \dots a^{m_q} b^{n_q}$ with $q > 0$, $0 \leq m_i \leq 2n_a$. The non initial rules allow to pump a^{n_a} .

Case $k = 3$: Let $L(\mathcal{A}^3)$ be a 3-normal term. For $w \in \Sigma^*$ define $EXP_3(w) = \Lambda(\{Sh(w, b^{hn_b}) \mid h \leq 2(|w| + 1)\}) \cap (a^* b^{n_b} b^* \Lambda a^*)^*$. Then the *ALD* \tilde{A}^3 with the following rules: $\{(\perp, EXP_3(w), \perp) \mid w \in SSP(\mathcal{A}^3)\} \cup \{(b^{n_b}, \Lambda b^{n_b}, \epsilon)\}$, defines $L(\mathcal{A}^3)$. The same argument of Case $k = 2$ applies.

Case $k = 4$: Let $L(\mathcal{A}^4)$ be a 4-normal term. For $w \in \Sigma^*$ define $EXP_4(w) = \Lambda(\{Sh(w, Sh(a^{hn_a}, b^{ln_b})) \mid h, l \leq 2(|w| + 1)\})$. Then let \tilde{A}^4 be the *ALD* with the following rules:

$$\begin{aligned} & \{(\perp, EXP_4(w), \perp) \mid w \in SSP(\mathcal{A}^4)\} \cup \\ & \{(ba^t, EXP_4(w), \epsilon) \mid 0 < t < n_a, w \in SL(\mathcal{A}^4)\} \cup \\ & \{(ab^s, EXP_4(w), \epsilon) \mid 0 < s < n_b, w \in SSL(\mathcal{A}^4)\} \cup \\ & \{(a^{n_a}, \Lambda a^{n_a}, \epsilon), (b^{n_b}, \Lambda b^{n_b}, \epsilon)\} \end{aligned}$$

\tilde{A}^4 defines $L(\mathcal{A}^4)$. Of course the frontier of each valid stencil tree is a word in $L(\mathcal{A}^4)$; conversely, a standard proof based on induction on the length of the words gives that a word $w \in L(\mathcal{A}^4)$ belongs to the language defined by \tilde{A}^4 .

Now *ALD* $A = \bigcup_{i,j:V_{i,j} \in F} \tilde{A}_{i,j}^k$ (where $\tilde{A}^1 = A^1$) defines L , since in the initial rules of *ALDs* \tilde{A}^2 defining the 2-normal terms of L , each placeholder has a left context of the form va^{n_a} for some $v \in \Sigma^*$; hence, only the rule $(a^{n_a}, \Lambda a^{n_a}, \epsilon)$ matches a constituent and does not affect the left context of the placeholder. Analogously, in the initial rules of *ALDs* \tilde{A}^3 , defining the 3-normal terms of L , each placeholder has a left context of the form vb^{n_b} , for some $v \in \Sigma^*$; hence, only the rule $(b^{n_b}, \Lambda b^{n_b}, \epsilon)$ matches a constituent and does not affect the left context of the placeholder. So the non initial rules of the forms $\{(ba^t, EXP_4(w), \epsilon) \mid 0 < t < n_a, w \in SSL(\mathcal{A}_{i,j}^4)\}$, $\{(ab^s, EXP_4(w), \epsilon) \mid 0 < s < n_b, w \in SL(\mathcal{A}_{i,j}^4)\}$ can be applied only when the rule used as initial rule belongs to an *ALD* \tilde{A}^4 defining a 4-normal term of L . \square

The construction of the normal automaton of a regular commutative language on $\{a, b\}$ can be easily generalized to regular commutative languages on alphabet of any finite cardinality; analogously, it is possible to define the normal terms of any regular commutative language with an *ALD*. In case we consider a language L belonging to $Com^+(\Sigma)$, i.e., to the positive boolean algebra generated by the languages of the form $F(x, k) = \{u \in \Sigma^* \mid |u|_x \geq k\}$ or $F(x, k, n) = \{u \in \Sigma^* \mid |u|_x \equiv k \pmod{n}\}$ with $k < n$, with $x \in \Sigma$, all final states of L are in the same region of the normal automaton; hence, the union of the rules of the *ALDs* defining the terms of L is an *ALD* defining L . Therefore:

Corollary 3.12 *Let Σ be a finite alphabet. Each language in $Com^+(\Sigma)$ is an *ALD* language.*

4 Conclusions

In this paper, we partially tackled the inclusion of regular languages in the *ALD* family. The classes of regular languages known to be *ALD* do not cover all regular languages, but exhibit a remarkable

variedness. For instance, group and locally testable languages have completely different syntactic monoids (groups and aperiodic monoids, respectively). Star-height one languages have non-empty intersection with both classes, and Proposition 3.4 says that *ALD* languages are "well-distributed" inside the class of regular languages. These considerations support the conjecture of the inclusion of regular languages in the *ALD* family.

Other open questions concern the hierarchy with respect to the number of placeholders, various decidability properties, minimization in the length of contexts and of patterns of the rules.

Acknowledgements: We are indebted with many people for long discussions on the issue of inclusion of regular languages in *ALD*. Among them, we gratefully thank Eberhard Bertsch and Mark-Jan Nederhof, who proposed many candidate counterexamples and found the first proof that star-height one languages are *ALD*, and M. Volkov, who suggested some possible extensions to the results presented here.

References

- [1] E. Bertsch and M.J. Nederhof, personal communication, Feb. 2002.
- [2] D. Beauquier, J.E. Pin, Factors of words, Lect. Notes in Comp. Sci, Springer, Berlin, 372(1989), 63-79.
- [3] V. Braitenberg and F. Pulvermüller, *Entwurf einer neurologischen Theorie der Sprache*, Naturwissenschaften **79** (1992), 103-117.
- [4] A. Cherubini, S. Crespi Reghizzi, P. San Pietro, *Languages Based on Structural Local Testability*, in C. S. Calude and M.J. Dinnen (eds.), *Combinatorics, Computation and Logic*, Proceedings of DMTCS99, Auckland, New Zealand, pp.18-21 Jan. 1999, pp. 159-174, Springer-Verlag.
- [5] A.Cherubini, S.Crespi Reghizzi, P.San Pietro, *Associative Language Descriptions*, Theoretical Computer Science, 270 (2002), pp.463-491.
- [6] A.Cherubini, S.Crespi Reghizzi, P.San Pietro, *Some structural properties of Associative Language Descriptions*, LNCS 2202, 2001, pp 172-183, Springer Verlag. New York.
- [7] S. Crespi Reghizzi, M. Pradella, P. San Pietro, *Associative definitions of programming languages*, Computer Languages, 26(2000) 105-123.
- [8] S. Crespi Reghizzi and V. Braitenberg, *Towards a Brain Compatible Theory of Language Based on Local Testability*, in V. Braitenberg and F.J. Rädermacher (eds), In C. Martin-Vide and V. Mitrana (eds), *Grammars and Automata for String Processing: from Mathematics and Computer Science to Biology, and Back* (1998), published by Taylor and Francis, Topics in Computer Mathematics 9, London, 2003, p. 17-32.
- [9] A.K. Joshi, L.S.Levy, K. Yueh, *Local constraints in the syntax and semantics of programming languages*, Procs. Fifth POPL (Annual ACM Symposium on the Principles of Programming Languages), Tucson, Arizona, January, 1978.
- [10] A. Salomaa, *Theory of Automata*, Pergamon Press, Oxford, 1969.