REGULAR LANGUAGES AND ASSOCIATIVE LANGUAGE DESCRIPTIONS[†]

Marcella Anselmo[1], Alessandra Cherubini[2]
and Pierluigi San Pietro[3]

[1]*Dip. Informatica ed Applicazioni, Università di Salerno, 84084 Fisciano (Salerno), Italia.*
email: anselmo@dia.unisa.it.
[2]*Dip. di Matematica, Politecnico di Milano, P.za Leonardo da Vinci 32, 20133 Milano, Italia.*
email: alessandra.cherubini@polimi.it.
[3]*Dip. di Elettronica e Informazione, Politecnico di Milano, P.za Leonardo da Vinci 32, 20133 Milano, Italia. email: sanpietr@elet.polimi.it.*

received 15 Jan 2005, revised 10 Jun 2005, accepted 1 Jul 2005.

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 known classes of regular languages are ALD: threshold locally testable languages, group languages, positive commutative languages and commutative languages on 2-letter alphabet. Moreover, we show that there is an ALD language in each level of restricted star height hierarchy. These results seem to show that ALD languages are "well-distributed" over the class of regular languages.

Keywords: Formal languages, Regular languages, Associative descriptions, Commutative languages, Group languages, Threshold locally testable languages

---

1 Introduction

The Associative Language Description model (ALD) (6) 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

[†]Work partially supported by MIUR grants PRIN 2005015419-002: "Linguaggi Formali e Automi: Metodi, Modelli e Applicazioni" and FIRB RBAU01MCAC: "Applicazioni della Teoria degli Automi all'Analisi, alla Compilazione e alla Verifica di Software Critico e in Tempo Reale".
()

information processing in the brain (5). A similar model was already proposed in (7). 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 (1) and in (2), 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 (8).

The main open problem about the ALD family 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 they do not seem to generalize to all regular languages.

The objective of this paper is to prove some inclusion results of subfamilies of regular languages into ALD family. More in details, we show that regular threshold locally testable languages (3), regular positive commutative languages, regular 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 (restricted) 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 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

We recall here the following definitions from (2).

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 labelled 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 labelled 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* $\mathrm{left}(K_i, T) = \mathrm{left}(T_i, T) = s$*; the* right context *of K_i in T and of T_i in T is* $\mathrm{right}(K_i, T) = \mathrm{right}(T_i, T) = t$.

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),*

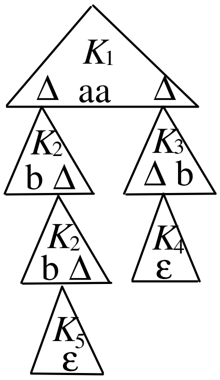

Fig. 1: A valid stencil tree for Example 2.5. Each K_i denotes a constituent matched by a rule.

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*.*

An *ALD* defines a set of constraints or test conditions that a stencil tree must satisfy, in the following sense.

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* $\perp left(K_i, T)$*, and 3)* y *is a prefix of* $\mathrm{right}(K_i, T) \perp$*. 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* $\{x \in \Sigma^* \mid x = \tau(T)$ *for some tree* $T \in \mathcal{T}(A)\}$*.*

Example 2.5 *Consider the language* L *on the alphabet* $\Sigma = \{a, b\}$ *defined by the regular expression* b^*aab^**. An ALD A for L is:*

$$\{(\perp, \Delta aa\Delta, \perp), (\epsilon, b\Delta, a), (a, \Delta b, \epsilon), (a, \epsilon, \epsilon), (\epsilon, \epsilon, a)\}$$

A valid stencil tree for the word bbaab is shown in Figure 1.

Note that when a left/right context is ϵ then the actual context is irrelevant for a pattern.

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 *ALD*: $(\perp, \Delta\Delta, \perp), (\perp, a, b), (a, b, \perp)$, defining the finite language ab. One cannot define an equivalent derivation relation $\rightarrow$: $\Delta \rightarrow \Delta\Delta \rightarrow a\Delta \rightarrow ab$ would not be correct, since each of the two rules $(\perp, a, b), (a, b, \perp)$ would require that the other one has already been applied.

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 (1) and in (2).

A few shorthands may be introduced to make the notation simpler and more compact. Given two 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 $\Lambda = \{\epsilon, \Delta\}$. For instance, the ALD A of Example 2.5 may be rewritten as: $\{(\perp, \Lambda aa\Lambda, \perp)$, $(\epsilon, b\Lambda, a)$, $(a, \Lambda b, \epsilon)\}$. Moreover, for $w = a_1 \cdots a_n$, with $a_1, \cdots, a_n \in \Sigma$, $\Lambda(w)$ denotes the language $\Lambda(w) = \Lambda a_1 \Lambda \cdots \Lambda a_n \Lambda$.

ALD languages have the following properties:

- Every ALD language is a CF-language (2).
- The ALD family is not closed under any of the following operations (2): 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 (1).
- Every 1-letter regular language is ALD (1).
- All regular languages of restricted star-height one are ALD (4).

3 Relations between regular languages and the ALD family

Since no proof has ever been found that every regular language is ALD, a natural approach is trying to prove that there are regular languages which are not ALD. However, this is not an immediate task. Before illustrating in the following subsections various new families of regular languages which are ALD, the next subsection illustrates a few techniques which might be used to solve the ALD vs. Regular problem.

3.1 Comparing ALD and regular languages

The nonclosure properties presented in Section 2 have been proven in (2) using nonregular ALD languages, and therefore do not help in ruling out that some regular language is not 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).

The similarities of CF and ALD languages (due to the their structural equivalence) break down when considering regular languages. For example, it is well-known that all regular languages can be defined by a right-linear grammar. However, it is immediate to prove that this is not the case for right-linear ALD, as defined next.

Definition 3.1 *An ALD is* right-linear *if all its rules are of the form* $(x, y\Lambda, \perp)$*, for* $x \in (\perp \cup \epsilon)\Sigma^*$*,* $y \in \Sigma^+$.

Proposition 3.2 *There exist regular languages that cannot be defined by a right-linear ALD.*

Proof: Consider $L = \{w \in \{a,b\}^* \mid |w|_a = 1\}$, where $|w|_a$ gives the number of occurrences of the letter a in the word w. Assume that there is a right-linear ALD A for L of degree $k > 0$ and width $m > 0$. Consider a word $zb^{2m+k}ab^{2m+k}$ corresponding to a stencil tree Z. Since the ALD is right-linear, there are two constituents K, K' of Z such that $\tau(K) \in b^+\Delta$, $\tau(K') \in b^*ab^*\Delta$ and $left(K,Z) \in b^*b^k = left(K',Z') \in b^*b^k$, with $right(K,Z) = right(K',Z') =\perp$. Hence, the stencil tree Z' obtained by replacing K' in Z with K is valid for A, but $\tau(Z') \in b^+$ and hence is not in L. □

The proof above can be generalized to provide a general tool to prove that a language is not ALD, namely the Swapping Lemma (2), which allows the exchange of two subtrees of two ALD valid trees, provided they have similar profiles. Before presenting it, a few definitions are needed.

Definition 3.3 ($first_k, last_k$,left$_k$, right$_k$)

For every $w \in \Sigma\cup\perp^*$*,* $first_k(w)$ *denotes the prefix of length* k *of* w *if* $|w| \geq k$*, otherwise it denotes* w*;* $last_k(w)$ *is the suffix of* w *of length* k *if* $|w| \geq k$*, otherwise* w*. The two operators are extended to every stencil tree* T*:* $first_k(T) = first_k(\tau(T))$*,* $last_k(T) = last_k(\tau(T))$.

Let Z *be a stencil tree and* T *be one of its maximal subtrees:* $\mathrm{left}_k(T,Z) = last_k(left(T,Z))$*, that is the left context of length* k *of* T *in* Z*. Symmetrically,* $\mathrm{right}_k(T,Z) = first_k(right(T,Z))$*, that is the right context of length* k *of* T *in* Z.

Lemma 3.4 (Swapping Lemma)

Let A *be an ALD of degree* $k \geq 0$ *and let* Z *and* Z' *be two valid trees of* A*, not necessarily distinct. Suppose that there exist two maximal subtrees* T *of* Z *and* T' *of* Z' *such that:*

$left_k(T,Z) = left_k(T',Z')$*,* $right_k(T,Z) = right_k(T',Z')$*,* $last_k(T) = last_k(T')$*,* $first_k(T) = first_k(T')$.

Then also the stencil tree Z'' *obtained from* Z *by replacing the subtree* T *with the subtree* T' *is a valid tree of* A.

The crux of Lemma 3.4 for proving that a regular language R is not ALD is that it requires identifying valid stencil trees for a word of R: unfortunately, in general, one does not have much information about their structure. The lemma is however particularly useful to prove that certain *nonregular* languages are not ALD, by combining it with traditional pumping lemmata of CF languages (that allow the identification of valid ALD trees for a word), as shown in Prop. 3.8 below (which is a more precise version of Corollary 3.7 of (2)). We first recall a traditional periodicity result for CF languages, namely the Ogden Lemma. A few preliminary definitions are useful. The symbols in a word z may be characterized by their positions: $1, 2, \ldots, |z|$ in z, starting from the leftmost and going to the rightmost. A *m-marked* word $m - mk(z)$ is a word z with a set of at least $m \geq 0$ positions of z.

Definition 3.5 *Given a language L, an integer $m \geq 0$ and a m-marked word $m - mk(z)$, with $z \in L$ and $|z| \geq m$, an* Ogden factorization *of z is a 5-tuple (x, u, w, v, y) such that:*

1. *$z = xuwvy$;*
2. *u and v together correspond to at least one marked position in $m - mk(z)$;*
3. *uwv corresponds to at most m marked positions in $m - mk(z)$;*
4. *for every $i \geq 0$, $xu^i wv^i y \in L$.*

Lemma 3.6 (Ogden's Lemma) *For an infinite CF language L there exists an integer $m > 0$ such that for all m-marked words $m - mk(z)$, with $z \in L$ and $|z| > m$, there exists an Ogden factorization.*

We remind the reader that if all positions are marked, then Ogden's Lemma is reduced to what is typically called the Pumping Lemma for CF languages.

Definition 3.7 *Two Ogden factorizations (x, u, w, v, y) and (x', u', w', v', y') are k-*similar *if the following conditions hold for some $i, j \geq 0$:*

1. *$first_k(u^i wv^i) = first_k(u'^j w'v'^j)$,*
2. *$last_k(u^j wv^i) = last_k(u'^j w'v'^j)$,*
3. *$first_k(y) = first_k(y')$,*
4. *$last_k(x) = last_k(x')$.*

Proposition 3.8 *For any ALD language $L \subseteq \Sigma^*$ of degree $k \geq 0$, there exists a constant $m > 0$ such that, given two marked words $m - mk(z), m - mk'(z')$, with $z, z' \in L$ and $|z| > m, |z'| > m$, if all pairs of Ogden factorizations for $m - mk(z)$ and $m - mk'(z')$ are k-similar, then for every pair of Ogden factorizations (x, u, w, v, y) and (x', u', w', v', y') of $m - mk(z)$ and $m - mk'(z')$ (respectively), $xu'^h w'v'^h y \in L$ for every $h \geq 0$.*

To show that a language L is not ALD, one has to suppose that there is an ALD of degree k, select $z, z' \in L$ and a marking for z and a marking for z'; then, find all possible Ogden factorizations of z and z', show they are all pairwise k-similar and then prove that a contradiction ensues for every factorization, by creating for each case a word $z'' \notin L$. For instance, we can show that the (nonregular) language $\{a^n cb^n a^m db^m \mid n, m \geq 0\}$ is not ALD. Apply Ogden's Lemma to a string $z = a^n cb^n a^m db^m$, where n and m are greater than the constant of the Lemma. Mark the leftmost group of a in z: by Ogden's Lemma, for every corresponding Ogden factorization (x, u, w, v, y) of z then $first_k(uwv) = a^k = last_k(x)$, $last_k(uwv) = b^k = first_k(y)$ and uwv is of the form a^+cb^+ (if $|u| < k$ or $|v| < k$ consider $xu^k wv^k y$ instead). Analogously, by marking the rightmost group of a, if (x', u', w', v', y') is an Ogden factorization then $first_k(u'w'v') = a^k = last_k(x')$, $last_k(u'w'v') = b^k = first_k(y')$ and $u'w'v'$ is of the form a^+db^+. By applying Proposition 3.8 to $z' = z$, a word in $a^+db^+a^+cd^+$ should be in the language, a contradiction.

Unfortunately, Proposition 3.8 does not seem very helpful in proving that some regular languages are not ALD. In fact, Ogden's Lemma seems too weak, when applied to regular languages, to restrict enough the possible Ogden factorizations of a word. For instance, the proof above does not work for the (ALD) regular language $L = a^*cb^*a^*db^*$. As above, consider in fact a string $z = z' = a^n cb^n a^m db^m$, where n and m are greater than the constant of Ogden's Lemma. Mark the leftmost group of a in z. Then two sets of Ogden factorizations (x, u, w, v, y) of z are possible: a set o_1 of factorizations such that $x \in a^*, uwv \in a^+, y \in a^*cb^n a^m db^m$, and a set o_2 such that $x \in a^*, uwv \in a^+cb^+, y \in b^+a^m db^m$. Similarly, by marking the rightmost group of a in z, at least two sets of Ogden factorizations (x', u', w', v', y') of z are possible: a set o_1' such that $x' \in a^n cb^n a^*, u'w'v' \in a^+, y \in a^+db^m$, and another set o_2' such that $x' \in a^n cb^n a^*, u'w'v' \in a^+db^+, y \in b^+$. Then the Ogden factorizations in o_1 and o_2' are not even k-similar, and Proposition 3.8 cannot be applied. Also, even if non- k-similar factorizations could be eliminated, the statement of Proposition 3.8 may not lead to a contradiction. For instance, if $(x, u, w, v, y) \in o_1$ and $(x', u', w', v', y') \in o_1'$ are k-similar, then for every $h \geq 0$, $xu'^h w' v'^h y$ is of the form $a^*a^+a^*cb^n a^m db^m$, which is in L.

3.2 Regular threshold locally testable languages

Threshold locally testable languages are a generalization of better known locally testable languages. In this section we prove that the class of regular threshold locally testable languages is contained in the *ALD* family. For any regular threshold locally testable language, an *ALD* can be devised in such a way to simulate the run of a scanner that reading a word from left to right counts the occurrences of meaningful factors.

Let Σ be a finite alphabet. Denote with $\Sigma^{<h}$ the set $\{\epsilon\} \cup \Sigma \cup \Sigma^2 \cup \ldots \Sigma^{h-1}$. For every word $x, w \in \Sigma^+$ let $|w|_x$ be the number of occurrences of x as a factor of w and let $\mathcal{F}(w)$ be the set of factors of w. 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* (3)) if may be described as a boolean composition of a finite set $\mathcal{B}$ of 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 $h = 1$.

To prepare the proof of the next theorem, we need a few more definitions. Given $\mathcal{B}$ as above, there is a finite set $\mathcal{G}_\mathcal{B} \subseteq (\perp \cup \Sigma)^+$ such that: $\mathcal{G}_\mathcal{B} = \{x \perp \mid \Sigma^* x \in \mathcal{B}\} \cup \{\perp x \mid x\Sigma^* \in \mathcal{B}\} \cup \{\perp x \perp \mid \{x\} \in \mathcal{B}\} \cup \{x \mid \exists h \geq 0 : F(x, h) \in \mathcal{B}\}$. We also extend the notion of stencil tree to a *generalized* stencil tree:

Definition 3.9 *A tree α is a* generalized stencil tree *if the internal nodes of α are labelled 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 $L(A)$ is equal to $\{x \in \Sigma^* \mid x = \tau(\alpha)$ for some tree $\alpha \in G_\mathcal{T}(A)\}$.

The proof that every *tlt* L, defined by a boolean function of a set $\mathcal{B}$, is *ALD* is based on the following observation. To decide whether a word w is in L it is possible to use a scanner endowed with one vector $\mathcal{V}$ of monotonic, finite counters, with one counter $\mathcal{V}_x$ for all $x \in \mathcal{G}_\mathcal{B}$: the scanner reads the word from left to right, one letter at a time, and increments one counter $\mathcal{V}_x$ whenever: 1) the current prefix of w ends in the factor x, and 2) the counter $\mathcal{V}_x$ has not reached its threshold. At the end of the scan of w, the scanner may check whether the various counters have reached their threshold, storing the results in a boolean vector. To check if $w \in L$, it is enough to apply a suitable boolean function, depending on L, to this boolean vector. A linear *ALD* A can be devised in such a way that a root-to-frontier visit of the constituents of a valid tree of frontier w, for every long enough $w \in \Sigma^+$, replicates the left-to-right run of the above scanner on w, with the left context l representing the prefix of w already examined by the scanner, and the *length* of the right context as a monotonic encoding of the vector $\mathcal{V}$. An *ALD* for this contains (apart from some suitable initialization and termination rules to be described next) rules of the following form:

$$(y, a\Delta\Sigma^h, \Sigma^j \perp)$$

with $y \in \Sigma^{k-1}$ (k being the length of the largest word in $\mathcal{G_B}$) , $a \in \Sigma$, and h, j defined by a suitable encoding rule to be discussed. Hence, a rule as above simulates a move of the scanner reading a after a factor y (hence, matching a factor ya), with a counter configuration $\mathcal{V}$ encoded by j, and moving to the new configuration $\mathcal{V}'$ encoded by $h + j$.

Initialization rules (i.e., matching only constituents at the root of a valid tree) are quite straightforward. They have the form $(\perp, y\Delta\Sigma^h, \perp)$ with $y \in \Sigma^{k-1}$ and h encoding the factor of y in $\mathcal{G_B}$.

Termination rules are those matching only constituents without children, i.e., constituents at the frontier of a valid tree. The idea is that a generalized stencil tree, valid for A, has a frontier always of the form $v\Delta z$, with $v, z \in \Sigma^*$, and $|z|$ bounded. Hence, a termination rule of the form:

$$(y, \epsilon, z \perp)$$

with $y \in \Sigma^{k-1}, z \in \Sigma^*$ is in A whenever the encoding $|z|$ corresponds to an acceptance configuration of the scanner. More precisely, since $|z|$ only encodes the configuration of the scanner after reading v, and not after reading $w = vz$ itself, one has to add a termination rule $(y, \epsilon, z \perp)$ considering the configuration reached by the scanner after reading vz: there may be other factors of $\mathcal{G_B}$ in z that have not been already counted. However, this test need not know the exact (unbounded) prefix v encountered by the scanner, but only its (bounded) encoding $|z|$, the bounded suffix y of v, and the factors in $\mathcal{G_B}$ of the bounded word yz. Hence, for a *tlt* L there is finite set of possibilities and therefore the termination rules are finite and can be added to A.

Notice that all words whose encoding is longer than the word itself are not dealt with by the above construction. However, since the possible encodings are finite, the set F of words $x \in L$ that are not longer than the largest encoding is finite. Hence, we just add to A a finite set of rules of the form $(\perp, x, \perp)$, for every $x \in F$.

The encoding used to store the vector $\mathcal{V}$ of monotonic counters is based on traditional ideas, loosely inspired by Gödel's famous encoding, based on prime factors, of sentences, proofs and theorems of arithmetic theory in his historical proof of the Incompleteness Theorem. Let $g : \mathcal{G_B} \to \mathcal{P}$, 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 $x \notin \mathcal{G_B}$. Now, given an integer $j \geq 0$ and the decomposition $p_1^{e_1} p_2^{e_2} \ldots$, with $p_i \in \mathcal{P}, e_i \geq 0$ in prime factors of an integer $q > 0$, define $[q]_j = p_1^{\min(j,e_1)} p_2^{\min(j,e_2)} \ldots$. Define $\bar{g}$ to enumerate all relevant factors of each word $v \in (\Sigma \cup \perp)^*$ as follows: $\bar{g}(\epsilon) = 0$ and for $v \neq \epsilon$:

$$\bar{g}(v) = \left[\prod_{w \in \mathcal{G_B}} g(w)^{|v|_w} \right]_n \tag{1}$$

Hence, the encoding $\bar{g}(v)$ of v stores all counter values up to the largest threshold n. As an example, consider $\mathcal{G}_{\mathcal{B}} = \{\perp a, \perp b, aa, ab, ba, abab, abba, b \perp, \perp aba \perp\}$, 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, $\bar{g}(\perp aabab \perp) = 2 \times 3 \times 5^2 \times 7 \times 11 \times 13 \times 19$.

The complete proof of following Prop. 3.10 contains the correct update rules of the encodings used in the right contexts.

Proposition 3.10 *Each regular threshold locally testable language is an ALD language.*

Proof:

Let L be a *tlt*, with $\mathcal{B}, \mathcal{G}_{\mathcal{B}}$ 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 $\mathcal{G}_{\mathcal{B}}$. We claim that there exists an ALD Q such that $L(Q) = L$. The following property of the above-defined enumeration $\bar{g}$ is immediate, for all $x, z \in \Sigma^+, y \in \Sigma^{\leq k}$:

$$\bar{g}(xyz) = [\bar{g}(xy) \cdot \bar{g}(yz)/\bar{g}(y)]_n \tag{2}$$

Let $G = \{h \geq 0 \mid \exists v \in \Sigma^* : v \in \Sigma^* \wedge h = \bar{g}(\perp v \perp)\}$. Clearly, G is finite. Let $M = \{m \geq 0 \mid \exists v \in \Sigma^+ : v \in L \wedge m = \bar{g}(\perp v \perp)\}$. We claim that $v \in L$ iff $\bar{g}(v) \in M$ (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 $\bar{g}(\perp v \perp)$ (each $F(x,h)$ can be checked since $h \leq n$).

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

Let $Q_2 = Q_2' \cup Q_2''$, with:

$$Q_2' = \left\{(\perp, y\Delta\Sigma^h, \perp) \mid y \in \Sigma^{k-1}, h = \bar{g}(\perp y)\right\}$$

$$Q_2'' = \left\{(y, a\Delta\Sigma^h, \Sigma^j \perp) \mid y \in \Sigma^{k-1}, a \in \Sigma, j \in G, h[j \cdot \bar{g}(ya)/\bar{g}(y)]_n - j\right\}$$

We claim that: (*) if a generalized stencil tree α is valid for Q_2 (i.e., $\alpha \in G_{\mathcal{T}}(Q_2)$), then $\exists v, z \in \Sigma^* : \tau(\alpha) = v\Delta z,\ \bar{g}(\perp v) = |z|$. The proof of (*) is by induction on the number i of constituents of a (obviously linear) generalized stencil tree α valid for Q_2. The base case $i = 1$ is immediate from the definition of Q_2'. Let α have $i+1$ constituents for $i > 0$. Then there exist $a \in \Sigma, x, v_1, z \in \Sigma^*, y \in \Sigma^{k-1}, \beta \in G_{\mathcal{T}}(Q_2)$ such that $\tau(\beta) v_1 y \Delta z, \tau(\alpha) = v_1 y a \Delta x z$. Hence, α has one constituent more than β, matched by the rule $(y, a\Delta x, z \perp) \in Q_2''$. Hence, by definition of Q_2'', $|x| = [|z| \cdot \bar{g}(ya)/\bar{g}(y)]_n - |z|$. By (2), $\bar{g}(v_1 y a) = [\bar{g}(v_1 y) \cdot \bar{g}(ya)/\bar{g}(y)]_n$. By induction hypothesis (applied to β), $\bar{g}(\perp v_1 y) = |z|$. Hence, $|xz| = |x| + |z|([|z| \cdot \bar{g}(ya)/\bar{g}(y)]_n - |z|) + |z|[|z| \cdot \bar{g}(ya)/\bar{g}(y)]_n = [\bar{g}(\perp v_1 y) \cdot \bar{g}(ya)/\bar{g}(y)]_n \bar{g}(\perp v_1 ya)$.

We also claim that: (+) for every $v \in \Sigma^{k-1}\Sigma^*, z \in \Sigma^*$ such that $\bar{g}(\perp v) = |z|$, 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. The base case is $v \in \Sigma^{k-1}$: the thesis follows immediately from the definition of Q_2'. If $|v| > 0$ then let $v = w'a, w' \in \Sigma^*, a \in \Sigma$. By induction hypothesis, for all $z' \in \Sigma^+$ such that $|z'| = \bar{g}(\perp w')$ there exists a generalized stencil tree in $G_{\mathcal{T}}(Q_2)$ of frontier $w'\Delta z'$. In particular, we can select z' to be a suffix of z ($|z'| = \bar{g}(\perp w') \leq \bar{g}(\perp w'a) = |z|$): $z = z''z'$ for one $z'' \in \Sigma^*$. Let w'' be the suffix of length $k-1$ 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, $|z'| = \bar{g}(\perp w')$: $[|z'| \cdot \bar{g}(w''a)/\bar{g}(w'')]_n \, [\bar{g}(\perp w') \cdot \bar{g}(w''a)/\bar{g}(w'')]_n$, which by (2), is equal to $\bar{g}(\perp w'a) = \bar{g}(\perp v)$, which by hypothesis is exactly equal to $|z|$.

Let $Q_3 = \{(y, \epsilon, z \perp) \mid y \in \Sigma^{k-1}, z \in \Sigma^*, |z| \leq \max(G), [|z| \cdot \bar{g}(yz \perp)/\bar{g}(y)]_n \in M, \}$.

We claim that an *ALD* Q for L is $Q_1 \cup Q_2 \cup Q_3$. To show that $L \subseteq L(Q)$, suppose $v \in L$. If $|v| \leq k + \max(G)$ then $v \in L(Q_1) \subseteq L(Q)$. If $v > k + \max(G)$ then $\exists! x, x' \in \Sigma^+ \mid v = xx'$ and $|x'| = \max(G)$ (therefore, $|x| \geq k$). Hence, since $\bar{g}(\perp v) \leq \max(G)$, $\exists w, z \in \Sigma^+$ such that $x' = wz$ and $|z| = \bar{g}(\perp v)$. Q_2 defines, by the above property (+), also the valid generalized stencil tree with frontier $x\Delta z$. It is 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 $[|z| \cdot \bar{g}(ywz \perp)/\bar{g}(y)]_n \in M$, with y being the suffix of length $k-1$ of x ($|x| \geq k$). But $|z|\bar{g}(\perp x)$. Hence, $[|z| \cdot \bar{g}(ywz \perp)/\bar{g}(y)]_n = [\bar{g}(\perp x) \cdot \bar{g}(ywz \perp)/\bar{g}(y)]_n$, which by (2) (and since y is a suffix of x) is equal to $\bar{g}(\perp xwz \perp) \in M$. To show that $L(Q) \subseteq L$, suppose that $v \in L(Q_2 \cup Q_3)$ (since $L(Q) = L(Q_1) \cup L(Q_2 \cup Q_3)$ and because we can ignore the obvious case $v \in L(Q_1)$). 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 $(y, w, z \perp) \in Q_3$, where $y \in \Sigma^{k-1}$ is suffix of $\perp x$. By (2), $\bar{g}(\perp xwz \perp) = [\bar{g}(\perp x) \cdot \bar{g}(ywz \perp)/\bar{g}(y)]_n$, which is equal, since $\bar{g}(\perp x) = |z|$ by the above property (*) of Q_2, to $[|z| \cdot \bar{g}(ywz \perp)/\bar{g}(y)]_n$ that by definition of Q_3 is in M. By definition of M, $v = xwz \in L$. □

Example 3.11 *We show here an example of the construction used in the proof of Prop 3.10. Consider the following tlt language L:*

$$\{a\Sigma^*\} \cap F(aa, 2) \cap \neg\{\Sigma^* b\} \cup \neg\{a\Sigma^*\} \cap \neg F(aa, 2) \cap \{\Sigma^* b\}$$

Then $\mathcal{G}_B = \{\perp a, aa, b \perp\}$. *Let* $g(\perp a) = 2, g(aa) = 3, g(b \perp) = 5$. *Hence,* $n = k = 2$ *and* $G = \{h \mid \exists 0 \leq h_1, h_2, h_3 \leq 2 i h = 2^{h_1} 3^{h_2} 5^{h_3}\}$, *with* $\max(G) = 2^2 \cdot 3^2 \cdot 5^2$. *A few of the elements of G are actually useless, since for every word w,the exponent of factors 2 and 5 in* $\bar{g}(\perp w \perp)$ *can never be larger than 1. M, by the definition of the expression of L and of the encoding g, is:* $\{2 \cdot 3^2, 5, 3 \cdot 5\}$. Q_1 *is the finite set* $\{(\perp, x, \perp) \mid x \in \Sigma^*, |x| \leq 2 + 2^2 3^2 5^2, x \in L\}$. $Q_2' = \{(\perp, y\Delta\Sigma^h, \perp) \mid y \in \Sigma^{k-1},\ h = \bar{g}(y)\} = \{(\perp, a\Delta\Sigma^{2^2}, \perp), (\perp, b\Delta, \perp)\}$.

$Q_2'' = \{(y, a\Delta\Sigma^h, \Sigma^j \perp) \mid y \in \Sigma^{k-1}, a \in \Sigma, j \in G, h = [j \cdot \bar{g}(ya)/\bar{g}(y)]_n - j\}$ *is the union of four sets* AA, AB, BA, BB.

$$AA = \{(a, a\Delta\Sigma^h, \Sigma^j \perp) \mid j \in G, h = [j \cdot \bar{g}(aa)/\bar{g}(a)]_2 - j\}$$

By the definition of $\bar{g}$ *and of the operator* $[]_2$, *h in the expression of AA is equal to* $[3j]_2 - j$, *which is equal to 0 if j is a multiple of* 3^2, $2j$ *otherwise. Rules in AA introduce a new occurrence of the factor aa: if the current encoding j has already counted factors aa up to the threshold 2, then the presence of the factor is disregarded. Otherwise, the length of the right context is incremented by twice its amount, i.e., it is multiplied by* $3 = g(aa)$.

$$AB = \{(a, b\Delta\Sigma^h, \Sigma^j \perp) \mid j \in G, h = [j \cdot \bar{g}(ab)/\bar{g}(a)]_2 - j\}$$

By the definition of $\bar{g}$ and of the operator $[]_2$, h in the expression of AB is equal to $[j]_2 - j$, which is equal to 0. Rules in AB introduce a new occurrence of the factor ab, which is not in $\mathcal{G}_{\mathcal{B}}$. Hence, the presence of this factor may be disregarded.

The other two sets can be computed as for the case of AB, obtaining $h = 0$ since they cannot add any factor of $\mathcal{G}_{\mathcal{B}}$:

$$BA = \left\{(b, a\Delta, \Sigma^j \perp) \mid j \in G\right\}, BB = \left\{(b, b\Delta, \Sigma^j \perp) \mid j \in G\right\}$$

being very similar to AB.

Finally, the set Q_3 is too large to be computed here, since one has to check, for every $h \in M$, if every word yz in $\Sigma^h + 1$ is such that $\bar{g}(yz \perp) \in M$. For instance, we must check all 2^{19} words of length $k - 1 + \max(M)$. We just give an example of a rule in Q_3: $(a, \epsilon, b^5 \perp)$ (in Q_3 since $5 \in M$ and $\bar{g}(ab^5 \perp) = 5 \in M$) and another example of a similar rule not in Q_3: $(a, \epsilon, a^5 \perp)$ (since $5 \in M$ and $\bar{g}(aa^5 \perp) = 3^2 \notin M$).

3.3 Relations with the Star-height Hierarchy

In this section we consider the (restricted) star height hierarchy of regular languages and show that there is an *ALD* language in each level of this hierarchy. This implies that *ALD* languages are well distributed inside the class of regular languages.

Let consider for each $i \geq 0$ the language R_i on a binary alphabet 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 (9). We introduce below, for each i, an *ALD* A_i defining R_i. Assuming that we are able of giving an *ALD* A_{i-1} defining R_{i-1}, the idea of the construction is that we can modify the rules of A_{i-1} to obtain a subset of rules of A_i. In particular, by appending valid trees of A_{i-1} to a placeholder of a generalized valid tree with frontier $(a^{2^i}\Lambda b^{2^i}\Lambda)^n, \ n \geq 0$ we obtain a tree whose frontier is in R_i.

Proposition 3.12 *There is an ALD language in each level of the star height hierarchy.*

Proof: We claim that R_i is defined by the *ALD* A_i whose rules are the following:

$$\{(\perp, \Lambda a^{2^{i-1}} X_{i-1}\Lambda, \epsilon), (\perp, \epsilon, \perp)\} \cup \bigcup_{1 \leq h < i} \left\{(\perp\ a^{2^{i-1}} a^{2^{i-2}} \cdots a^{2^{i-h}}, \Lambda a^{2^{i-h-1}} X_{i-h-1}\Lambda, \epsilon)\right\} \cup$$

$$\bigcup_{1 \leq h < i, i-h \leq k < i} \{(ba^{2^k} a^{2^{k-1}} \cdots a^{2^{i-h}}, \Lambda a^{2^{i-h-1}} X_{i-h-1}\Lambda, \epsilon)\} \cup$$

$$\bigcup_{1 \leq h < i, 2^{i-h} \leq j < 2^{i-h+1}} \{(ab^j, \Lambda a^{2^{i-h-1}} X_{i-h-1}\Lambda, \epsilon)\}$$

where $X_t = \Lambda b^{2^t}$ if $t > 0$ and $X_0 = b$. Let

$$A_i(\perp) = \{(\perp a^{2^{i-1}} a^{2^{i-2}} \cdots a^{2^{i-h}}, \Lambda a^{2^{i-h-1}} X_{i-h-1}\Lambda, \epsilon) \mid 1 \le h < i\},$$
$$A_i(b) = \{(ba^{2^k} a^{2^{k-1}} \cdots a^{2^{i-h}}, \Lambda a^{2^{i-h-1}} X_{i-h-1}\Lambda, \epsilon) \mid 1 \le h < i, i-h \le k < i\}$$
$$A_i(a) = \{(ab^j, \Lambda a^{2^{i-h-1}} X_{i-h-1}\Lambda, \epsilon) \mid 1 \le h < i, 2^{i-h} \le j < 2^{i-h+1}\}$$

let $\sigma_i((x, y, \epsilon)) = (xa^{2^i}, y, \epsilon)$ for all $(x, y, \epsilon) \in A_i(\perp) \cup A_i(b)$, and finally let

$$\sigma_i((\perp, \Lambda a^{2^{i-1}} X_{i-1}\Lambda, \epsilon)) = \{(\perp a^{2^i}, \Lambda a^{2^{i-1}} X_{i-1}\Lambda, \epsilon), (ba^{2^i}, \Lambda a^{2^{i-1}} X_{i-1}\Lambda, \epsilon)\}$$
$$\cup\{xb^{2^i}, \Lambda a^{2^{i-1}} X_{i-1}\Lambda, \epsilon) \mid (x, \Lambda a^{2^{i-2}} X_{i-2}\Lambda, \epsilon) \in A_i(a)\}$$

Then the *ALD*s A_i can be recursively defined as follow:

$$A_1 = \{(\perp, \Lambda ab\Lambda, \epsilon), (\perp, \epsilon, \perp)\}$$
$$A_{i+1} = \{(\perp, \Lambda a^{2^{i-1}} X_{i-1}\Lambda, \epsilon), (\perp, \epsilon, \perp)\} \cup A_i(a) \cup A_i(b) \cup \sigma_i((\perp, \Lambda a^{2^{i-1}} X_{i-1}\Lambda, \epsilon)) \cup$$
$$\bigcup_{(x,y,\epsilon)\in A_i(\perp)\cup A_i(b)} \sigma_i((x, y, \epsilon))$$

We will prove by induction on i that $R_i = L(A_i)$. The case $i = 1$ is trivial ($R_1 = (ab)^*$ is *ALD*). Assume that $R_{i-1} = L(A_{i-1})$. We first show that $R_i \subseteq L(A_i)$. The proof that for all $w \in R_i$ it is the case that $w \in L(A_i)$ is by induction on $|w|$. Both the empty word and the shortest non-empty word in R_i, $a^{2^{i-1}} b^{2^{i-1}}$, belong to R_i and to $L(A_i)$. Then let $w = \prod_{1\le h\le n}(a^{2^{i-1}} u_h b^{2^{i-1}} v_h)$ (where $\prod$ denotes concatenation) with $u_h, v_h \in R_{i-1}, 1 \le h \le n$ and $u_1v_1 \neq \epsilon$ if $n = 1$. First suppose $n = 1$. If $v_1 \neq \epsilon$ then $w_0 = a^{2^{i-1}} u_1 b^{2^{i-1}}$ is a word of R_i shorter than w, so there are both a valid tree $T(w_0)$ of A_i for w_0 and a generalized valid tree T_0 (see Def. 3.9) of A_i, with $\tau(T_0) = w_0\Delta$. Moreover, $v_1 \in R_{i-1}$, and hence there is a valid tree $T(v_1)$ of A_{i-1} for v_1. Since the proof now requires references to individual nodes of a stencil tree, we denote every node by the progressive integer obtained by enumerating the nodes of the tree in a breadth-first, left-to-right visit of the tree starting at the root (Hence, the nodes are numbered 0, 1, 2...). Let j be the leaf of T_0 labelled by Δ. Appending $T(v_1)$ to the node j in T_0 results into a tree T with $\tau(T) = w$. To prove that T is a valid tree of A_i we have to prove that each constituent of the maximal subtree T_j, whose root is j, is matched by a rule of A_i. Since T_j is a copy of $T(v_1)$, each constituent K of T_j has in $T(v_1)$ a copy matched by a rule $r_{i-1}(K) \in A_{i-1}$. Moreover, *left*$(K, T) = w_0x$ with *left*$(K, T(v_1)) = \perp x$ and either $x = \epsilon$ or $first_{2^{i-2}}(x) = a^{2^{i-2}}$. Hence, if $r_{i-1}(K) \in A_{i-1}(a) \cup A_{i-1}(b) \subset A_i$, K is matched by the same rule (of A_i) in T. Then suppose $r_{i-1}(K) \in A_{i-1}(\perp) \cup \{(\perp, \Lambda a^{2^{i-2}} X_{i-2}\Lambda, \epsilon)\}$. If $r_{i-1}(K)$ is the rule $(\perp a^{2^{i-2}} a^{2^{i-3}} \cdots a^{2^{i-h-1}}, \Lambda a^{2^{i-h-2}} X_{i-h-2}\Lambda, \epsilon)$ for some $h, 1 \le h < i-1$, then *left*$(K, T) = w_0 a^{2^{i-2}} a^{2^{i-3}} \cdots a^{2^{i-h-1}}$ and since $last_1(w_0) = b$, K is matched in T by the rule $(ba^{2^{i-2}} a^{2^{i-3}} \cdots a^{2^{i-h-1}}, \Lambda a^{2^{i-h-2}} X_{i-h-2}\Lambda, \epsilon) \in A_{i-1}(b) \subset A_i$. Otherwise, if $r_{i-1}(K)$ is the rule $(\perp, \Lambda a^{2^{i-2}} X_{i-2}\Lambda, \epsilon)$ then *left*$(K, T) = w_0$ and since $last_{2^i}(w_0) = a^t b^{2^i - t}$ for some $t > 0$ such that $2^i - t \ge 2^{i-1}$, then K is matched in T by the rule $(ab^{2^i-t}, \Lambda a^{2^{i-2}} X_{i-2}\Lambda, \epsilon) \in \sigma_{i-1}((\perp, \Lambda a^{2^{i-2}} X_{i-2}\Lambda, \epsilon)) \subset A_i$. Hence, each constituent K of T is matched by some rule of A_i, and therefore T is a valid tree of A_i. If $v_1 = \epsilon$, then $w_0 = a^{2^{i-1}} u_1 b^{2^{i-1}}$ with $u_1 \neq \epsilon$, $u_1 \in R_{i-1}$.

Hence there is a valid tree $T(u_1)$ for u_1 in A_{i-1}. Then let T_0 be the generalized stencil tree of A_i formed by the constituent matched by the initial rule $(\perp, a^{2^{i-1}}\Delta b^{2^{i-1}}, \perp)$. Appending $T(u_1)$ to the placeholder of T_0, we get a tree T with $\tau(T) = w$. Each constituent K of $T(u_1)$ is matched by a rule $r_{i-1}(K) \in A_{i-1}$. Again, we can assume $r_{i-1}(K) \in A_{i-1}(\perp) \cup \{(\perp, \Lambda a^{2^{i-2}} X_{i-2}\Lambda, \epsilon)\}$. If $r_{i-1}(K)$ is the rule $(\perp\ a^{2^{i-2}} a^{2^{i-3}} \cdots a^{2^{i-h-1}}, \Lambda a^{2^{i-h-2}} X_{i-h-2}\Lambda, \epsilon)$ for some $h, 1 \leq h < i-1$, then *left*$(K,T) = a^{2^{i-1}} a^{2^{i-2}} a^{2^{i-3}} \cdots a^{2^{i-h-1}}$ and K is matched by the rule $(\perp\ a^{2^{i-1}} a^{2^{i-2}} a^{2^{i-2}} a^{2^{i-3}} \cdots a^{2^{i-h-1}}, \Lambda a^{2^{i-h-2}} X_{i-h-2}\Lambda, \epsilon) \in \sigma_{i-1}(A_{i-1}(\perp)) \subset A_i$. Otherwise, if $r_{i-1}(K)$ is the rule $(\perp, \Lambda a^{2^{i-2}} X_{i-2}\Lambda, \epsilon)$ then *left*$(K,T) = a^{2^{i-1}}$ and K is matched by the rule $(\perp\ a^{2^{i-1}}, \Lambda a^{2^{i-2}} X_{i-2}\Lambda, \epsilon) \in \sigma_{i-1}((\perp, \Lambda a^{2^{i-2}} X_{i-2}\Lambda, \epsilon)) \subset A_i$. Then, in any case, each constituent K of T is matched by some rule of A_i, and T is again a valid tree of A_i. Then consider the case $n \geq 1$. Both $w_1 = a^{2^{i-1}} u_1 b^{2^{i-1}} v_1, w_2 = \prod_{2 \leq h \leq n}(a^{2^{i-1}} u_h b^{2^{i-1}} v_h)$ are shorter than w and belong to R_i: by induction, there are two valid trees $T(w_1)$ and $T(w_2)$ of A_i such that $\tau(T(w_1)) = w_1,\ \tau(T(w_2)) = w_2$. $T(w_2)$ has a unique constituent K matched by a rule of the form $(\perp, x, \epsilon)$ with $x \in \{a^{2^{i-1}}\Lambda b^{2^{i-1}}\Lambda\}$. Replacing K with a constituent K' matched by the rule $(\perp, \Delta x, \epsilon)$, we get a generalized valid stencil tree T_2 whose frontier is Δw_2. Let j be the unique leaf of T_2 labelled by a placeholder. Appending $T(w_1)$ to the node j of T_2 results in a tree T with $\tau(T) = w_1 w_2 = w$. To prove that T is a valid tree, it is enough to verify that all of its constituents appended to a node $k \geq j$ are matched by some rule of A_i. Let T_1 be the maximal subtree of T of root j. T_1 is a copy of $T(w_1)$, so each constituent C appended to a node of T_1 has the same left context of its copy in $T(w_1)$: it must be matched by some rule of A_i. Now let C be a constituent of T appended to a node $m > j$ not belonging to the set of nodes of T_1. C is a constituent of T_2: it is matched in T_2 by a rule $r_i(C)$ of A_i. Obviously, *left*$(C,T) = w_1 x$ where *left*$(C,T_2) =\perp x$ and $first_{2^{i-1}}(x) = a^{2^{i-1}}$. So if $r_i(C) \in A_i(a) \cup A_i(b)$ then C is matched by $r_i(C)$ also in T. Then let $r_i(C) \in A_i(\perp) \cup \{(\perp, \Lambda a^{2^{i-1}} X_{i-1}\Lambda, \epsilon)\}$. If $r_i(C)$ is the rule $(\perp\ a^{2^{i-1}} a^{2^{i-2}} \cdots a^{2^{i-h}}, \Lambda a^{2^{i-h-1}} X_{i-h-1}\Lambda, \epsilon)$ for some $h, 1 \leq h < i$, then *left*$(C,T) = w_1 a^{2^{i-1}} a^{2^{i-2}} \cdots a^{2^{i-h}}$ and since $last_1(w_1) = b$, K is matched in T by the rule $(ba^{2^{i-1}} a^{2^{i-2}} \cdots a^{2^{i-h}}, \Lambda a^{2^{i-h-1}} X_{i-h-1}\Lambda, \epsilon) \in (A_i(b)) \subset A_i$. Otherwise, if $r_i(C)$ is the rule $(\perp, \Lambda a^{2^{i-1}} X_{i-1}\Lambda, \epsilon)$, then *left*$(C,T) = w_1$ and since $last_{2^i}(w_0) = a^t b^{2^i - t}$ for some $t > 0$ such that $2^i - t \geq 2^{i-1}$, C must be matched in T by the rule $(ab^{2^i - t}, \Lambda a^{2^{i-1}} X_{i-1}\Lambda, \epsilon) \in A_i(a) \subset A_i$. Hence, each constituent C of T is matched by some rule of A_i: T is a valid tree of A_i. So $R_i \subseteq L(A_i)$ for all positive integers i.

To prove the opposite inclusion, $R_i \supseteq L(A_i)$, we have to show that the frontier of each valid tree of A_i belongs to R_i for each $i > 0$. Again, the proof is by induction on i. Let T be a valid tree of A_i. Of course, if T has height 1, $\tau(T) = a^{2^{i-1}} b^{2^{i-1}} \in R_i$. Assume $\tau(T') \in R_i$ for each valid tree T' of A_i of height $m < n$ and let T a valid tree of height n. The constituent K_0 of T with the same root 0 of T is matched by a rule $(\perp, \Lambda a^{2^{i-1}}\Lambda b^{2^{i-1}}\Lambda, \perp)$. Enumerate all the nodes in T also when they are labelled by $\Lambda = \epsilon$, and denote by T_i the maximal subtree of T appended to the node i. T_1 (if non empty) is a valid tree of A_i of height $m \leq n-1$, hence $w_1 = \tau(T_1) \in R_i$. $T_{2+2^{i-1}},\ T_{3+2^i}$ (if non empty) are valid trees of A_{i-1}, shortly called T_2 and T_3, hence $w_2 = \tau(T_2),\ w_3 = \tau(T_3) \in R_{i-1}$. So the frontier of T is $w_1 a^{2^{i-1}} w_2 b^{2^{i-1} w_3}$ which obviously is in R_i. □

3.4 Regular group languages

A group language is a language whose syntactic monoid is a group. We will show that any regular group language can be defined by an ALD. The proof is based on the analysis of the loops in a particular automaton recognizing the language that is related to its syntactic monoid.

Let us firstly introduce some notations. 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 $[w]_L$ the $\equiv_L$-congruence class of $w \in \Sigma^*$. Further, let us denote $SYM(L)$ 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 $SYM(L)$ recognizes L. We will say that a path $(q_1, q_2, \cdots, q_n)$ 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 $1 \leq i_1 < i_2 < \cdots < i_{2k-1} < i_{2k} \leq i_n$ such that for any odd $h = 1, \cdots, 2k-1$ we have that: $q_{i_h} = q_{i_{h+1}}$, the path $(q_1, \cdots, q_{i_1}, q_{i_2+1}, \cdots, q_{i_3}, q_{i_4+1}, \cdots, q_{i_{2k-1}}, q_{i_{2k}+1}, \cdots, q_n)$ has loop-nesting degree $d-1$ and the path $(q_{i_h}, q_{i_h+1}, \cdots, q_{i_{h+1}})$ has no repetition of states unless for state q_{i_h}. Finally, given an automaton $\mathcal{A}$, we denote by $SSP(\mathcal{A})$, the set of labels of all successful simple paths (a simple path from an initial to a final state) in $\mathcal{A}$, and by $SL(\mathcal{A})$, the set of labels of all simple loops on the initial state of $\mathcal{A}$. The reference to the automaton is dropped when there is no ambiguity on the automaton to be considered.

Remark that the automaton of transitions in $M(L)$ is a deterministic automaton, and that in general it is bigger than the minimal automaton. Nevertheless its introduction here is useful because in the case of group languages its loops are very special: they can be inserted in any position of any word of the language, regardless to the contexts. This property allows us to construct an *ALD* defining the language.

Proposition 3.13 *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. Consider the automaton $SYM(L)$ of transitions in $M(L)$ and let y be the label of any loop on a state m in $SYM(L)$. Hence $my \equiv_L m$. Since $M(L)$ is a group, there exists $m' \in M(L)$ such that $m'm = 1$ and then $[y]_L = m'm\ [y]_L = m'm = 1$. This implies that any loop in $SYM(L)$ is also a loop on any state and that for any $u, v \in \Sigma^*$ we have $uv \in L$ iff $uyv \in L$. An *ALD* $A(L)$ defining L can therefore be constructed as follows. The rules of $A(L)$ are the following: $\{(\perp, \Lambda(w), \perp) \mid w \in SSP\}$ and $\{(\epsilon, \Lambda(w), \epsilon) \mid w \in SL\}$.

We claim that a word (is in L iff it) is recognized by $SYM(L)$ iff it is defined by $A(L)$. Remark that since w is in $\Lambda(w)$, and because of the special kind of rules in $A(L)$, we have that there is a valid stencil tree in $A(L)$ with frontier w iff there is a valid generalized stencil tree (see Def. 3.9) in $A(L)$ with frontier w' for any w' in $\Lambda(w)$. A word w is recognized by $SYM(L)$ iff there is a successful path in $SYM(L)$ labelled w. We show by induction that there is a successful path labelled w in $SYM(L)$ of loop-nesting degree d iff there is a valid generalized stencil tree for $A(L)$ with frontier w' for any w' in $\Lambda(w)$ and height h with $h = d$. One has $d = 0$ iff the path is simple iff the rules $(\perp, \Lambda(w), \perp)$ are in $A(L)$, giving raise to a generalized valid stencil tree of height $h = 0$ whose frontier is w' for any w' in $\Lambda(w)$. Suppose now $d > 0$. According to the above definition, a successful path of loop-nesting degree $d > 0$ is a successful path $(q_1, \cdots, q_{i_1}, q_{i_2+1}, \cdots, q_{i_3}, q_{i_4+1}, \cdots, q_{i_{2k-1}}, q_{i_{2k}+1}, \cdots, q_n)$ having loop-nesting degree $d-1$ and

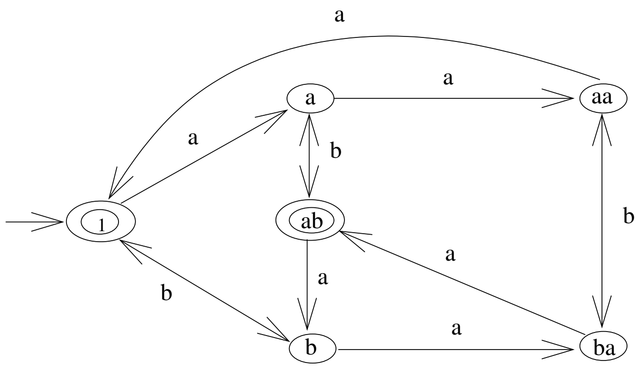

Fig. 2: The automaton $SYM(L)$ for the language L in Example 3.14.

label z, where some simple loops with labels $y_1, \cdots, y_l$ are inserted on $q_{i_1}, q_{i_3}, \cdots, q_{i_{2k+1}}$ (note $l \geq k+1$). By induction, this holds iff there is a valid generalized stencil tree for $A(L)$ of height $h-1 = d-1$ and frontier z' for any z' in $\Lambda(z)$, and the rules $(\epsilon, \Lambda(y_1), \epsilon), \cdots, (\epsilon, \Lambda(y_l), \epsilon)$ are in $A(L)$. Finally this condition holds iff there is a valid generalized stencil tree for $A(L)$ of height $h = d$ and frontier w' for any w' in $\Lambda(w)$. □

Example 3.14 *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. The automaton* $SYM(L)$ *of transitions in* $M(L)$ *is given in Fig. 2. The sets* SSP *and* SL *for* $SYM(L)$ *are the following:* $SSP = \{1, ab, ba^2, a^2ba\}$*, and* $SL = \{b^2, a^3, abab, baba, ba^2ba^2, a^2ba^2b, aba^2ba\}$*. According to Proposition 3.13, an ALD defining language* L *is given by the set of the following rules:* $(\perp, \epsilon, \perp)$*,* $(\perp, \Lambda(ab), \perp)$*,* $(\perp, \Lambda(ba^2), \perp)$*,* $(\perp, \Lambda(a^2ba), \perp)$*, and* $(\epsilon, \Lambda(b^2), \epsilon)$*,* $(\epsilon, \Lambda(a^3), \epsilon)$*,* $(\epsilon, \Lambda(abab), \epsilon)$*,* $(\epsilon, \Lambda(baba), \epsilon)$*,* $(\epsilon, \Lambda(ba^2ba^2), \epsilon)$*,* $(\epsilon, \Lambda(a^2ba^2b), \epsilon)$*,* $(\epsilon, \Lambda(aba^2ba), \epsilon)$*.*

3.5 Relations with regular commutative languages

In this section we consider regular commutative languages. The family of regular commutative languages on an alphabet Σ is the boolean algebra generated by languages of the form $F(x, k) = \{u \in \Sigma^* \mid |u|_x \geq k \geq 0\}$ or $F(x, k, n) = \{u \in \Sigma^* \mid |u|_x \equiv k \ (mod\ n)\}$ with $0 < k < n$, for $x \in \Sigma$. Any regular commutative language has a normal form and is recognized by a normal automaton. From the normal automaton we are able to construct an ALD for two classes of regular commutative languages: regular languages in Com^+, the positive boolean algebra generated by languages of the form $F(x, k)$, $F(x, k, n)$, and regular commutative languages on an alphabet of at most two letters.

To introduce the normal form we need some notations. Define $E(x, k) = \{u \in \Sigma^* \mid |u|_x = k\}$ and for every $k \geq 0$ define $E(x, k, n) = \{u \in \Sigma^* \mid |u|_x \equiv k(mod\ n),\ |u|_x \geq k\}$. Then:

$\overline{F(x,k)} = \bigcup_{0\le i<k} E(x,i)$, $\overline{F(x,k,n)} = \bigcup_{0\le i<n, i\ne k} F(x,i,n)$, $F(x,k) = E(x,k,1)$, and, for $k<n$, $E(x,k,n) = F(x,k,n)$. Hence, each regular commutative language belongs to the positive (i.e., negation-free) boolean algebra generated by the languages of the form $E(x,k)$ or $E(x,k,n)$ (called in the sequel *literals* $L(x)$ on x) where $k\ge 0,\ n>0,\ x\in\Sigma$. Moreover, for each $x\in\Sigma,\ L(x) = L(x)\cap E(y,0,1), x\ne y$ and $\forall h>0:\ E(x,k,n) = \bigcup_{0\le i<t} E(x,k+in,hn)$ where t is the least integer such that $k+tn\ge hn$.

Definition 3.15 *A regular commutative language L on Σ is representable in* normal form *if: $\exists r>0 : L = \bigcup_{1\le j\le r} C_j$, where $C_j = \bigcap_{x\in\Sigma} L_j(x)$, with either $L_j(x) = E(x,k_{j,x})$ or $L_j(x) = E(x,k_{j,x},n_x)$ for some nonnegative integer $k_{j,x}$ depending on j and x and a positive integer n_x only depending on x. The terms C_j are called* normal terms *of L.*

From previous remarks, using standard set operation properties, it is immediate to prove the following result.

Fact 3.16 *Each regular commutative language can be represented in normal form.*

Now we show how to construct a normal automaton recognizing a regular commutative language, starting from its normal form. The normal automaton is divided in some regions characterized by the allowed letters in the loops. Further the loops in a region can be inserted on any state in that region. This allows us to construct an *ALD* for any language of labels of successful paths ending in a same region.

In the sequel L will denote a regular commutative language in normal form on the alphabet $\Sigma = \{a_1, a_2, ..., a_n\}$, and then $L = \bigcup_{1\le j\le r} C_j$ for some positive integer r, with $C_j = \bigcap_{i=1,\cdots,n} L_j(a_i)$. Let $L_i = \bigcup_{1\le j\le r} L_j(a_i) = (\bigcup_{j'\in J'_i} E(a_i, k_{j',a_i})) \cup (\bigcup_{j''\in J''_i} E(x, k_{j'',a_i}, n_{a_i}))$ be the union of all literals on a_i occurring in L. Denote $n_i = n_{a_i}$, $k_{j,a_i} = k_{j,i}$ and let $K_i = \ max_{j'\in J'_i}\{k_{j',i}\}$, (where $K_i = 0$ if $J'_i = \emptyset$ and $n_i = 1$ if $J''_i = \emptyset$).
The Cayley graph $\Gamma(L_i)$ of the syntactic monoid of L_i has $K_i + n_i + 1$ vertices $X_{i,l}$, $0\le l\le K_i+n_i$, loops labelled by a_j with $j\ne i$ are based at each vertex, an edge labelled by a_i goes from $X_{i,t}$ to $X_{i,t+1}$ for each t, $0\le t\le K_i+n_i-1$ and an edge with label a_i from X_{i,K_i+n_i} to X_{i,K_i+1}. The graph $\Gamma(L_i)$ has two parts: the first part contains the K_i+1 vertices $X_{i,l}, 0\le l\le K_i$, and no loops whose labels contain a_i are based at its vertices, the second part contains the n_i vertices $X_{i,l}, K_i+1\le l\le K_i+n_i$ and a loop labelled by $a_i^{n_i}$ is based at each vertex. The automaton $SYM(L_i)$ has $\Gamma(L_i)$ as underlying graph, initial state $X_{i,0}$ and final states $\{X_{i,k_{j',i}} \mid j'\in J'_i\} \cup \bigcup_{j''\in J''_i} \{X_{i,\tilde{k}_{j'',i}} \mid \tilde{k}_{j'',i} \equiv k_{j'',i}\ (mod\ n_i) \wedge k_{j'',i} \le \tilde{k}_{j'',i} \le K_i+n_i\}$.
Let $\Gamma = \prod_{i=1,n}\Gamma(L_i)$. Γ is partitioned in 2^n regions denoted by $\Gamma_{(\eta_1,\dots,\eta_n)}$, with $\eta_i = 0,1$, composed as follows. For each region $\Gamma_{(\eta_1,\dots,\eta_n)}$ let $I'_{(\eta_1,\dots,\eta_n)} = \{i \mid 1\le i\le n\ \wedge\ \eta_i = 1\}$, $I''_{(\eta_1,\dots,\eta_n)} = \{i \mid 1\le i\le n \wedge \eta_i = 0\}$. Then $\Gamma_{(\eta_1,\dots,\eta_n)}$ has $\prod_{i'\in I'_{(\eta_1,\dots,\eta_n)}} n_{i'} \times \prod_{i''\in I''_{(\eta_1,\dots,\eta_n)}} (K_{i''}+1)$ vertices $V^{(\eta_1,\dots,\eta_n)}_{(m_1,m_2,\dots,m_n)} = (X_{1,m_1}, ..., X_{n,m_n})$ with $K_i+1\le m_i\le K_i+n_i$ for $i\in I'_{(\eta_1,\dots,\eta_n)}$ and $0\le m_i\le K_i$ for $i\in I''_{(\eta_1,\dots,\eta_n)}$. The same set of loops are based at each vertex, the loops are labelled by anagrams of $\prod_{i\in I'_{(\eta_1,\dots,\eta_n)}} a_i^{s_i\cdot n_i}$, $s_i\ge 0$. In particular $\Gamma_{(0,0,\dots,0)}$ has $\prod_{i=1,n}(K_i+1)$ vertices $V^{(0,0,\dots,0)}_{(m_1,m_2,\dots,m_n)} = (X_{1,m_1}, ..., X_{n,m_n})$ with $0\le m_i\le K_i$, and no loop is based on its vertices. The apices on the vertex names indicate the region of Γ which the vertex belongs to and can be omitted.

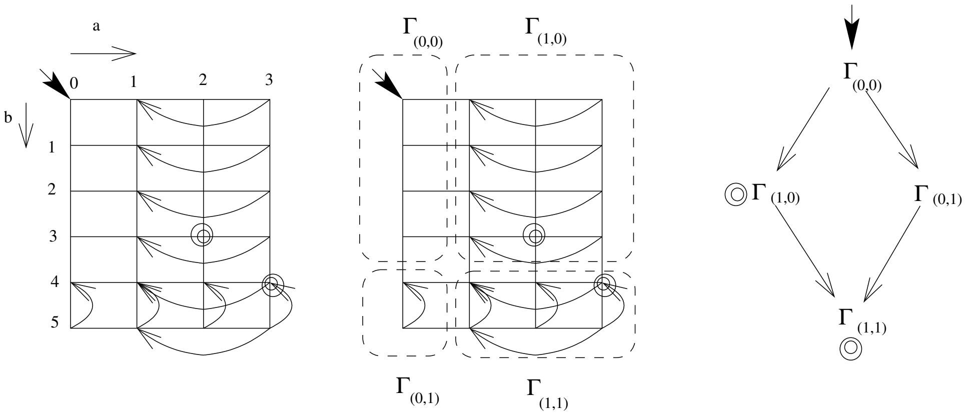

Fig. 3: The normal automaton for language L of Example 3.18.

Definition 3.17 *Using the above notations, the* normal automaton *of a regular commutative language L on the alphabet Σ is an automaton $\mathcal{A} =< \Gamma, V_{(0,0,...,0)}, F >$, (with Γ the underlying graph, $V_{(0,0,...,0)}$ the initial state, F the set of final states) such that a vertex $V_{(m_1,m_2,...,m_n)} \in F$ iff there exists a normal term $C_j = \bigcap_{1\le i\le n} L_j(a_i)$ of L and either $L_j(a_i) = E(a_i, m_i)$ or $L_j(a_i) = E(a_i, t_i, n_i)$ with $t_i \equiv m_i$ (mod n_i) and $t_i \le m_i \le K_i + n_i$.*

Example 3.18 *Let $L \subseteq \{a, b\}^*$ be the commutative language $L = C_1 \cup C_2$, where $C_1 = E(a, 2, 3) \cap E(b, 3)$ and $C_2 = E(a, 3, 3) \cap E(b, 4, 2)$. Following the above definitions and considering $a = a_1$, $b = a_2$, we have: $n_a = 3$, $n_b = 2$, $J_1' = \emptyset$, $J_1'' = \{1, 2\}$, $J_2' = \{1\}$, $J_2'' = \{2\}$, $K_1 = 0$ and $K_2 = 3$. The normal automaton of L is represented in Fig. 3, where the four regions $\Gamma_{(0,0)}$, $\Gamma_{(1,0)}$, $\Gamma_{(0,1)}$ and $\Gamma_{(1,1)}$ are shown. We find that in $\Gamma_{(0,0)}$ there are no loops on the vertices; in $\Gamma_{(1,0)}$ there are only loops labelled a^3 based on each vertex; in $\Gamma_{(0,1)}$ there are only loops labelled b^2 based on each vertex; and in $\Gamma_{(1,1)}$ there is the same set of loops based on each vertex labelled by any anagram of $a^{3t}b^{2s}$, for some $t, s \ge 0$. Fig. 3 also shows the way one can move from a region of the automaton to another one.*

It is easy to verify that the normal automaton of a language L recognizes L. Moreover, for each $V_{(m_1,m_2,...,m_n)} \in F$, denote by $\mathcal{A}_{(m_1,m_2,...,m_n)}$ the automaton $< \Gamma, V_{(0,0,...,0)}, V_{(m_1,m_2,...,m_n)} >$, whose unique final state is $V_{(m_1,m_2,...,m_n)}$. Then, the language $L(\mathcal{A}_{(m_1,m_2,...,m_n)})$ is a normal term of L; conversely, each normal term of L is a language $L(\mathcal{A}_{(m_1,m_2,...,m_n)})$ for some $V_{(m_1,m_2,...,m_n)} \in F$. If $V_{(m_1,m_2,...,m_n)}$ is in the region $\Gamma_{(\eta_1,...,\eta_n)}$ of Γ, $L(\mathcal{A}_{(m_1,m_2,...,m_n)})$ will be called for shortness an $(\eta_1, ..., \eta_n)$-normal term of L. A word w belonging to an $(\eta_1, ..., \eta_n)$-normal term of L, has m_i occurrences of a_i for $i \in I''_{(\eta_1,...,\eta_n)}$ while no upper bound is given for the occurrences of a_i with $i \in I'_{(\eta_1,...,\eta_n)}$.

For the definitions of SSP and SL, see Section 3.4.

Lemma 3.19 *With the above notations, each $(\eta_1, ..., \eta_n)$-normal term of a regular commutative language L on the alphabet Σ is an ALD language.*

Proof: Obviously a $(0, 0, ..., 0)$- normal term of L is a finite language $L(\mathcal{A}_{(m_1,m_2,...,m_n)})$ (and it is obviously defined by the *ALD* $A^{(0,0,...,0)}_{(m_1,m_2,...,m_n)}$ whose rules are $\{(\perp, w, \perp) \mid w \in L(\mathcal{A}_{(m_1,m_2,...,m_n)})\})$. Then let $L(\mathcal{A}_{(m_1,m_2,...,m_n)})$ be a $(\eta_1,...,\eta_n)$-normal term of L with $(\eta_1,...,\eta_n) \neq (0,0,...,0)$. It is easy to verify that the automaton $\mathcal{A}^{(\eta_1,...,\eta_n)}_{(m_1,m_2,...,m_n)}$, obtained by $\mathcal{A}_{(m_1,...,m_n)}$ adding an edge labelled by a_i going from each vertex $V_{(h_1,...,h_i,...,h_n)}$, with $h_i \geq n_{a_i} - 1$, to the vertex $V_{h_1,...,h_i - n_{a_i}+1,...,h_n}$ for all $i \in I'_{(\eta_1,\eta_2,...,\eta_n)}$ recognizes $L(\mathcal{A}_{(m_1,m_2,...,m_n)})$. The same set of loops is based at each vertex of the automaton $\mathcal{A}^{(\eta_1,...,\eta_n)}_{(m_1,m_2,...,m_n)}$, so using an argument similar to the one used in the proof of of Proposition 3.13, it is easy to prove that the *ALD* $A^{(\eta_1,...,\eta_n)}_{(m_1,m_2,...,m_n)}$ whose rules are $\{(\perp, \Lambda(w), \perp) \mid w \in SSP(\mathcal{A}^{(\eta_1,...,\eta_n)}_{(m_1,m_2,...,m_n)})\}$ and $\{(\epsilon, \Lambda(w), \epsilon) \mid w \in SL(\mathcal{A}^{(\eta_1,...,\eta_n)}_{(m_1,m_2,...,m_n)})\}$, defines $L(\mathcal{A}_{(m_1,m_2,...,m_n)})$. □

Remark 3.20 For each commutative regular language L on the alphabet Σ, denote by $F_{(\eta_1,\eta_2,...,\eta_n)}$ the set of final states of its normal automaton belonging to the region $\Gamma_{(\eta_1,...,\eta_n)}$. If either $F = F_{(\eta_1,\eta_2,...,\eta_n)}$ or $F = F_{(0,0,...,0)} \cup F_{(\eta_1,\eta_2,...,\eta_n)}$ for some region $\Gamma_{(\eta_1,...,\eta_n)}$ then L is an *ALD*-language. In fact, in such cases, all normal terms of L are either $(\eta_1, \eta_2, ..., \eta_n)$ or $(0,0,...0)$-normal terms. The *ALD*s defining $(\eta_1, \eta_2, ..., \eta_n)$-normal terms differ only for their initial rules (i.e., rules of the form $(\perp\ u, v, w\ \perp)$ for some $u, w \in \Sigma^*$, $v \in (\Sigma \cup \Delta)^*$) and the *ALD*s defining $(0,0,...,0)$-normal terms have only initial rules. Then the *ALD* defining L is obtained by taking all the rules of the *ALD*s defining the terms of L. In the other cases the union A of the *ALD*s defining the normal terms of L does not define L because the pattern of non initial rules of an *ALD* defining an $(\eta_1, \eta_2, ..., \eta_n)$-normal term $((\eta_1, \eta_2, ..., \eta_n) \neq (0,0,...,0))$ of L, are different from the patterns of non initial rules of an *ALD* defining an $(\eta'_1, \eta'_2, ..., \eta'_n)$-normal term $((\eta'_1, \eta'_2, ..., \eta'_n) \neq (\eta_1, \eta_2, ..., \eta_n)$, $(\eta'_1, \eta'_2, ..., \eta'_n) \neq (0,0,...,0))$ of L but these rules are not distinguished by means of their contexts, hence some words not belonging to L can be the frontier of a valid tree of the *ALD* A.

We have shown how to construct an *ALD* for any normal term of a regular commutative language. Unfortunately we are not able to construct an *ALD* in the general case of a regular commutative language, since the union of the *ALD* for each normal term does not always define the language (see Remark 3.20). Now we show how to solve the problem in two particular cases: regular languages in Com^+ (Proposition 3.21); and regular commutative languages on alphabets with two letters at most (Proposition 3.22).

Consider a language L belonging to $Com^+(\Sigma)$, i.e., to the positive boolean algebra generated by the languages of the form $F(x,k)$ or $F(x,k,n)$ with $k < n$ and $x \in \Sigma$. In this case all final states of L are in the region $\Gamma_{(1,...,1)}$ of the normal automaton of L. Hence by Lemma 3.19 and Remark 3.20 the following Proposition immediately follows.

Proposition 3.21 *Let Σ be a finite alphabet. Each language in $Com^+(\Sigma)$ is an ALD language.*

The following result for regular commutative languages on small alphabets can also be proved. Let $Sh(u,v) = \{u_0v_1u_1...v_su_sv_{s+1} \mid u = u_0u_1...u_s,\ v = v_1v_2...v_{s+1},\ s \geq 0,\ u_0, v_{s+1} \in \Sigma^*,\ u_i, v_i \in \Sigma^+\ for\ 0 < i < s\}$ denote the shuffle of some $u, v \in \Sigma^+$.

Proposition 3.22 *A regular commutative language on an alphabet Σ with $|\Sigma| \leq 2$ is an ALD language.*

Proof: In (2), it was proved that each unary regular language is an ALD language. Then, let L be a regular commutative language in normal form on $\Sigma = \{a_1, a_2\}$ and let $\mathcal{A} =< \Gamma, V_{(0,0)}, F >$ be the normal automaton recognizing L. Obviously $\mathcal{A}$ has four regions. By Remark 3.20, it follows that the ALDs $A^{(\eta_1,\eta_2)}_{(i,j)}$ defining (η_1, η_2)-normal terms of L have to be modified in ALDs $\widetilde{A}^{(\eta_1,\eta_2)}_{(i,j)}$ so that if $(\eta_1, \eta_2) \neq (\eta'_1, \eta'_2)$ the non initial rules of the ALD defining an (η_1, η_2)-normal term have the same contexts of a non initial rule of the ALD defining an (η'_1, η'_2)-normal term only if they also have equal patterns.
The regions of the normal automaton of L are four, and for each region (η_1, η_2) a corresponding type of ALD $A^{(\eta_1,\eta_2)}$ was defined in Lemma 3.19. The ALDs $A^{(0,0)}$ defining $(0, 0)$-normal terms contain only initial rules, so they do not require modifications, hence $A^{(0,0)}_{(i,j)} = \widetilde{A}^{(0,0)}_{(i,j)}$ for each (i, j) such that $L(\mathcal{A}^{(0,0)}_{(i,j)})$ is a $(0, 0)$-normal term of L. So we have to deal with the following three regions.
Region $(1, 0)$. Let $L(\mathcal{A}^{(1,0)}_{(i,j)})$ be a $(1, 0)$-normal term of L. For $w \in \Sigma^*$ define $EXP_{(1,0)}(w) = \Lambda(\{Sh(w, a_1^{hn_1}) \mid h \leq 2(|w| + 1)\}) \cap (a_2^* a_1^{n_1} a_1^* \Lambda a_2^*)^*$. Then the ALD $\widetilde{A}^{(1,0)}_{(i,j)}$ with the following rules: $\{(\perp, EXP_{(1,0)}(w), \perp) \mid w \in SSP(\mathcal{A}^{(1,0)}_{(i,j)})\} \cup \{(a_1^{n_1}, \Lambda a_1^{n_1}, \epsilon)\}$, defines $L(\mathcal{A}^{(1,0)}_{(i,j)})$. In fact, the initial rules allow the introduction, by means of a valid tree of height 1, of all the words belonging to $L(\mathcal{A}^{(1,0)}_{(i,j)})$ in the form $a_1^{m_0} a_2^{h_1} a_1^{m_2} a_2^{h_2} ... a_1^{m_q} a_2^{h_q}$ with $q > 0,\ 0 \leq m_i \leq 2n_1$. The non initial rules allow to pump $a_1^{n_1}$.
Region $(0, 1)$. Let $L(\mathcal{A}^{(0,1)}_{(i,j)})$ be a $(0, 1)$-normal term. For every $w \in \Sigma^*$, define $EXP_{(0,1)}(w) = \Lambda(\{Sh(w, a_2^{hn_2}) \mid h \leq 2(|w| + 1)\}) \cap (a_1^* a_2^{n_2} a_2^* \Lambda a_1^*)^*$. Then the ALD $\widetilde{A}^{(0,1)}_{(i,j)}$ with the following rules: $\{(\perp, EXP_{(0,1)}(w), \perp) \mid w \in SSP(\mathcal{A}^{(0,1)}_{(i,j)})\} \cup \{(a_2^{n_2}, \Lambda a_2^{n_2}, \epsilon)\}$, defines $L(\mathcal{A}^{(0,1}_{(i,j)})$. The same argument of Region $(1, 0)$ applies.
Region $(1, 1)$. Let $L(\mathcal{A}^{(1,1)}_{(i,j)})$ be a $(1, 1)$-normal term. For every $w \in \Sigma^*$, define $EXP_{(1,1)}(w) = \Lambda(\{Sh(w, Sh(a_1^{hn_1}, a_2^{ln_2})) \mid\ h, l \leq 2(|w| + 1)\})$. Then let $\widetilde{A}^{(1,1)}_{(i,j)}$ be the ALD with the following rules:

$$\begin{aligned}
&\{(\perp, EXP_{(1,1)}(w), \perp) \mid w \in SSP(\mathcal{A}^{(1,1)}_{(i,j)})\} \cup \\
&\{(a_2 a_1^t, EXP_{(1,1)}(w), \epsilon) \mid 0 < t < n_1,\ w \in SL(\mathcal{A}^{(1,1)}_{(i,j)})\} \cup \\
&\{(a_1 a_2^s, EXP_{(1,1)}(w), \epsilon) \mid 0 < s < n_2,\ w \in SL(\mathcal{A}^{(1,1)}_{(i,j)})\} \cup \\
&\{(a_1^{n_1}, \Lambda a_1^{n_1}, \epsilon),\ (a_2^{n_2}, \Lambda a_2^{n_2}, \epsilon)\}
\end{aligned}$$

$\widetilde{A}^{(1,1)}_{(i,j)}$ defines $L(\mathcal{A}^{(1,1)}_{(i,j)})$. Of course the frontier of each valid stencil tree is a word in $L(\mathcal{A}^{(1,1)}_{(i,j)})$; conversely, a standard proof based on induction on the length of the words gives that a word $w \in L(\mathcal{A}^{(1,1)}_{(i,j)})$ belongs to the language defined by $\widetilde{A}^{(1,1)}_{(i,j)}$.
Now in the initial rules of ALDs $\widetilde{A}^{(1,0)}$ defining the $(1, 0)$-normal terms of L, each placeholder has a left context of the form $va_1^{n_1}$ for some $v \in \Sigma^*$; hence, only the rule $(a_1^{n_1}, \Lambda a_1^{n_1}, \epsilon)$ matches a constituent and does not affect the left context of the placeholder. Analogously, in the initial rules of ALDs $\widetilde{A}^{(0,1)}$, defining the $(0, 1)$-normal terms of L, each placeholder has a left context

of the form $va_2^{n_2}$, for some $v \in \Sigma^*$; hence, only the rule $(a_2^{n_2}, \Lambda a_2^{n_2}, \epsilon)$ matches a constituent and does not affect the left context of the placeholder. So the non initial rules ALDs $\widetilde{A}^{(1,1)}$, whose left context belong to the set $\{(a_2 a_1^t \mid 0 < t < n_1\} \cup \{(a_1 a_2^s \mid 0 < s < n_2\}$, which are the only rules whose patterns can contain both a_1, a_2 can be applied only when the rule used as initial rule belongs to an ALD $\widetilde{A}^{(1,1)}$ defining a $(1,1)$-normal term of L. □

Example 3.23 *Let $L \subseteq \{a,b\}^*$ be the commutative language $L = C_1 \cup C_2$, where $C_1 = E(a,2,3) \cap E(b,3)$ and $C_2 = E(a,3,3) \cap E(b,4,2)$, as in Example 3.18.*

Let us construct an ALD defining L. From Lemma 3.19, the language C_1 is defined by the ALD $A^{(1,0)}_{(2,3)}$ containing the rules: $(\perp, \Lambda(w), \perp)$, for any $w \in SSP(\mathcal{A}^{(1,0)}_{(2,3)})$, that is the set of all anagrams of a^2b^3; and $(\epsilon, \Lambda(a^3), \epsilon)$. Again from Lemma 3.19, the language C_2 is defined by the ALD $A^{(1,1)}_{(3,4)}$ containing the rules: $(\perp, \Lambda(w), \perp)$, for any $w \in SSP(\mathcal{A}^{(1,1)}_{(3,4)})$, that is the set of all anagrams of a^3b^4; and $(\epsilon, \Lambda(w), \epsilon)$, for any $w \in SL(\mathcal{A}^{(1,1)}_{(3,4)})$, that is the set containing a^2, b^3 and all anagrams of a^3b^2.

Unfortunately the ALD $A^{(1,0)}_{(2,3)} \cup A^{(1,1)}_{(3,4)}$ does not define L; indeed it also defines some word not belonging to L: as an example it defines a^2b^5. Following Proposition 3.22, let us construct another ALD $\widetilde{A}^{(1,0)}_{(2,3)}$ defining C_1, and another ALD $\widetilde{A}^{(1,1)}_{(3,4)}$ defining C_2, whose union defines L.

The ALD $\widetilde{A}^{(1,0)}_{(2,3)}$ has the following rules:

$$\{(\perp, EXP_{(1,0)}(w), \perp) \mid w \in SSP(\mathcal{A}^{(1,0)}_{(2,3)})\} \cup \{(a^3, \Lambda a^3, \epsilon)\}$$

where $EXP_{(1,0)}(w) = \Lambda(\{Sh(w, a^{3h}) \mid h \leq 2(|w|+1)\}) \cap (b^* a^3 a^* \Lambda b^*)^*$. That is the initial rules are obtained from some x that is the shuffle of an anagram of a^2b^3 with some a^{3h} with $h \leq 12$ where we can put Λ only just after an a^3. As an example for $h = 1$ we obtain as x: $a^5\Lambda b^3$, $ba^5\Lambda b^2$, $b^2a^5\Lambda b$, $b^3a^5\Lambda$; for $h = 2$ we obtain: $a^3\Lambda a^5\Lambda b^3$, $ba^3\Lambda ba^5\Lambda b$, and so on.

The ALD $\widetilde{A}^{(1,1)}_{(3,4)}$ has the following rules: $\{(\perp, EXP_{(1,1)}(w), \perp) \mid w \in SSP(\mathcal{A}^{(1,1)}_{(3,4)})\} \cup$ $\{(ba^t, EXP_{(1,1)}(w), \epsilon) \mid t = 1,2,\ w \in SL(\mathcal{A}^{(1,1)}_{(3,4)})\} \cup \{(ab, EXP_{(1,1)}(w), \epsilon) \mid w \in SSL(\mathcal{A}^{(1,1)}_{(3,4)})\} \cup$ $\{(a^3, \Lambda a^3, \epsilon),\ (b^2, \Lambda b^2, \epsilon)\}$, where $EXP_{(1,1)}(w) = \Lambda(\{Sh(w, Sh(a^{3h}, b^{2l})) \mid\ h, l \leq 2(|w|+1)\})$ and $SSL(\mathcal{A}^{(1,1)}_{(3,4)})$ is the set of all anagrams of a^3b^2. As an example the initial rules are $(\perp, \Lambda(w), \perp)$ for any w anagram of $a^{3+3h}b^{4+2l}$ with $h, l \leq 16$.

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 3.12 says that ALD languages are "well-distributed" inside the class of regular languages, and that may be all regular languages are indeed ALD

On the other hand, the techniques used to capture these families in this paper do not seem to be generalizable to all regular languages. For instance, the method used here for showing that threshold locally testable (*tlt*) languages are ALD does not work for showing that regular languages in $Com^{+}(\Sigma)$ are ALD; but the method used to prove the latter, does not work for many commutative languages that are not in $Com^{+}(\Sigma)$ but are actually *tlt* (and hence ALD).

Other open questions for the ALD family 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] A.Cherubini, S.Crespi Reghizzi, and P.San Pietro. Some structural properties of associative language descriptions. volume 2202 of *Lecture Notes in Computer Science*, pages 172–183, Berlin, 2001. Springer-Verlag.

[2] A.Cherubini, S.Crespi Reghizzi, and P.San Pietro. Associative language descriptions. *Theoretical Computer Science*, 270:463–491, 2002.

[3] D. Beauquier and J.E. Pin. Factors of words. volume 372 of *Lecture Notes in Computer Science*, pages 63–79, Berlin, 1989. Springer-Verlag.

[4] E. Bertsch and M.J. Nederhof. Personal communication. Feb 2002.

[5] V. Braitenberg and F. Pulvermüller. Entwurf einer neurologischen Theorie der Sprache. *Naturwissenschaften*, (79):103–117, 1992.

[6] A. Cherubini, S. Crespi Reghizzi, and P. San Pietro. Languages based on structural local testability. In C. S. Calude and M.J. Dinnen, editors, *Combinatorics, Computation and Logic: Proceedings of DMTCS99,Auckland, New Zealand, 18-21 Jan 1999*, pages 159–174.

[7] A.K. Joshi, L.S.Levy, and K. Yueh. Local constraints in the syntax and semantics of programming languages.

[8] S. Crespi Reghizzi, M. Pradella, and P. San Pietro. Associative definitions of programming languages. *Computer Languages*, 26:105–123, 2000.

[9] A. Salomaa. *Theory of Automata*. Pergamon Press, Oxford, 1969.