Boolean Algebras from Trace Automata

Alexandre Mansard

Boolean Algebras from Trace Automata

Alexandre Mansard
LIM, University of La Réunion, France
alexandre.mansard@univ-reunion.fr

Abstract

We consider trace automata. Their vertices are Mazurkiewicz traces and they accept finite words. Considering the length of a trace as the length of its Foata normal form, we define the operations of level-length synchronization and of superposition of trace automata. We show that if a family $\mathcal{F}$ of trace automata is closed under these operations, then for any deterministic automaton $H \in \mathcal{F}$, the word languages accepted by the deterministic automata of $\mathcal{F}$ that are length-reducible to H form a Boolean algebra. We show that the family of trace suffix automata with level-regular contexts and the subfamily of vector addition systems satisfy these closure properties. In particular, this yields various Boolean algebras of word languages accepted by deterministic vector addition systems.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Boolean algebras, traces, automata, synchronization, pushdown automata, vector addition systems

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2019.48

1 Introduction

In automatic verification, it is useful to highlight families of languages with good closure properties, as for example Boolean algebras of languages. For example, the fact that the first-order theory of any word-automatic graph[1] is decidable essentially relies on the Boolean closure properties of regular languages: to any relation defined by a first-order formula, there corresponds (in an effective way) a regular language and the problem of deciding whether the graph satisfies a given statement reduces to the problem of deciding emptiness for the corresponding regular language [13].

The regular, context-free, context-sensitive and recursively enumerable languages form a famous increasing hierarchy of formal language families defined by Chomsky in [8] from grammars of increasing complexity. It can also be obtained from families of automata. Indeed, regular languages are accepted by finite automata, context-free languages are accepted by pushdown automata (or more generally by word suffix automata [2]), context-sensitive languages are accepted, for instance, by bounded synchronized automata [20], and recursively enumerable languages are accepted by Turing machines [4]. And if it is well-known that regular languages or context-sensitive languages form a Boolean algebra, it is also well-known that it is not the case of context-free languages. Nevertheless, it was shown that various subclasses of context-free languages form Boolean algebras [17, 18, 5], as for example visibly pushdown languages with respect to a given pushdown alphabet [1]. Besides, pushdown automata model sequential computations. For parallel computations, a relevant family of automata consists of vector addition systems. Hence, the question arises of which Boolean algebras can be obtained from this family of automata.

[1] A word-automatic graph is a graph of which the vertex set is a regular language and each relation is recognized by a finite letter by letter synchronized transducer.

In this paper, we consider trace automata. Their vertices are Mazurkiewicz traces [10] and they accept finite words. Traces bear the advantage of describing executions of concurrent systems [23]. We define the length of a trace as the length of its Foata normal form and we show that we can obtain Boolean algebras from any trace automata family $\mathcal{F}$ closed under level-length synchronization and level-length superposition (Theorem 34). These operations ensure the stability under intersection and difference of the class of recognized word languages. More precisely, we show that for any deterministic automaton $H \in \mathcal{F}$, the class of languages accepted by the deterministic automata in $\mathcal{F}$ length-reducible to H form a Boolean algebra of word languages.

Then, we apply the previous result to the family TrSuffix of trace suffix automata (with level-regular contexts), introduced in [16]. A trace suffix automaton is described by a finite set of rules of the form $\mathcal{W}(u \xrightarrow{a} v)$, where u and v are traces, a is a label and $\mathcal{W}$ is a level-regular trace language (*i.e.*, a language of traces of which the Foata normal forms form a regular word language). We show that this family satisfies the closure conditions stated above.

Lastly, we deduce that the subfamily $\mathrm{TrSuffix}^{\mathrm{VAS}}$ of trace suffix automata over trace monoids of which the dependence alphabet is the equality also satisfies the closure conditions stated above. Since $\mathrm{TrSuffix}^{\mathrm{VAS}}$ essentially corresponds to some vector addition systems (VAS), we obtain various Boolean algebras of word languages accepted by deterministic vector addition systems.

Related works. In [6, 7], Caucal and Rispal adapt Eilenberg's recognizability for languages [11] to infinite automata in order to obtain Boolean algebras. More precisely, in [7], they show how to obtain various Boolean algebras from any family of word automata (*i.e.*, automata of which vertices are words) closed under the operations of length synchronization and superposition. However, vector additions systems, seen as word automata (each trace is encoded by its Foata normal form), are not closed under length superposition.

Besides, in the literature, different types of VAS languages were considered [9, 12, 19, 22]: the labeling function may be free or not, λ-transitions (transitions labeled by the empty word) may be allowed or not, the set of final markings may be finite or equal to all accessible markings. In general, the various investigations focus on closure properties [12] and on the relationship with the other classical formal languages enumerated above [22, 21, 14]. Indeed, it is well known that VAS languages contain regular languages, are incomparable with context-free languages, but are context-sensitive [19]. The regularity (respectively the context-freeness) of some VAS languages is decidable [22] (respectively [21, 14]). But, if some type of VAS languages are closed under union and intersection, the complementation remains, to our knowledge, a challenging problem. In the usual terminology of VAS (or Petri nets), the VAS for which we prove that the languages form a Boolean algebra are labeled (*i.e.*, the labeling function is total and may not be injective), deterministic (from any marking, distinct transitions labeled by a same letter are not allowed) and equipped of a level-regular set of final vertices. Moreover, an action can be performed only with respect to a context that is assumed level-regular also.

2 Preliminaries

In this section, we give some preliminaries about automata and Mazurkiewicz traces. We use standard notations. In particular, the union of two disjoint sets A and B is denoted $A \dot{\cup} B$.

2.1 Automata: definition and generalities

An automaton is given by a set of edges labeled by letters, plus initial and final vertices.

Let V be a set (of vertices), T be a set of symbols called *terminals* and $C = \{\iota, o\}$ be a set of *colors*. A *T-automaton* G over V is a subset of $V \times T \times V \ \cup \ C \times V$ of vertex set

$$V_G := \{v \in V \mid \exists\, a, u \ (u, a, v) \in G \ \vee \ (v, a, u) \in G\} \ \cup \ \{v \in V \mid \exists\, c \ (c, v) \in G\}$$

such that the set $T_G = \{a \in T \mid \exists\, u, v \in V, (u, a, v) \in G\}$ is finite. The automaton G is *finite* if its vertex set V_G is finite. Denote by $I_G := \{v \in V_G \mid (\iota, v) \in G\}$ the set of *initial vertices* of G and by $F_G := \{v \in V_G \mid (o, v) \in G\}$ the set of *final vertices* of G.

An element $(u, a, v) \in G$ is an *edge*. Its *label* is a, its *source* is u and its *target* is v. The notation $u \xrightarrow[G]{a} v$ (or $u \xrightarrow{a} v$ when G is understood) means $(u, a, v) \in G$. Any couple $(c, v) \in G$ is a vertex v *colored* by $c \in C$.

The automaton G is *deterministic* if I_G is reduced to a singleton and for each $a \in T$, if $(u \xrightarrow[G]{a} v$ and $u \xrightarrow[G]{a} v')$ then $v = v'$. Given a family $\mathcal{F}$ of automata, we denote by $\mathcal{F}_{\mathrm{det}}$ the subfamily of $\mathcal{F}$ consisting in deterministic automata.

A *path* in G of *source* u and *goal* v, labeled by a word $a_1 \dots a_k \in T^*$, is a finite sequence of the form $u \xrightarrow[G]{a_1} u_1, \dots, u_{k-1} \xrightarrow[G]{a_k} v$, with $u = v$ for $k = 0$. We denote by $u \xrightarrow[G]{a_1 \dots a_k} v$ the existence of such a path. A path is *accepting* if its source is initial and its goal is final. The *language accepted* by an automaton G is the set $L(G)$ of words that label its accepting paths. The regular languages of T-words are the languages accepted by finite automata.

A *morphism* f from a T-automaton G into a T-automaton H is a mapping $f : V_G \to V_H$ such that

$$u \xrightarrow[G]{a} v \Longrightarrow f(u) \xrightarrow[H]{a} f(v) \quad \text{and} \quad (c, u) \in G \Longrightarrow (c, f(u)) \in H$$

If such a morphism exists, we write $G \xrightarrow{f} H$ (or just $G \to H$) and we say that G is *f-reducible* (or just *reducible*) to H. Moreover, if f is a bijection and $G \xrightarrow{f} H$ and $H \xrightarrow{f^{-1}} G$, then G and H are said to be *f-isomorphic*.

▶ **Lemma 1.** *Let G and H be automata. If $G \to H$, then $L(G) \subseteq L(H)$.*

Suppose there exists a length-mapping $|\ | : V \longrightarrow \mathbb{N}$. A morphism f is *length-preserving* if $|f(v)| = |v|$ for any $v \in V_G$. If such a morphism exists, we say that G is *length-reducible* to H and we write $G \xrightarrow{f}_{\ell} H$ (or just $G \to_{\ell} H$). The automata G and H are *length-isomorphic* if there exists a length-preserving morphism f such that G and H are f-isomorphic.

▶ **Example 2.** Let us consider the infinite binary tree $T_2 := \{Ku \xrightarrow{a} Kua \mid u \in \{a, b\}^*\} \cup \{Ku \xrightarrow{b} Kub \mid u \in \{a, b\}^*\} \ \cup \ \{(\iota, K)\} \ \cup \ \{o\} \times K\{a, b\}^*$ and the visibly pushdown automaton $\mathrm{Vis}(\{c\}, \varnothing, \{a, b\}) := \{\bot\top^n \xrightarrow{a,b} \bot\top^{n+1} \mid n \geqslant 0\} \ \cup \ \{\bot\top^n \xrightarrow{c} \bot\top^{n-1} \mid n \geqslant 1\} \ \cup \ \{\bot \xrightarrow{c} \bot\} \ \cup \ \{(\iota, \bot)\} \ \cup \ \{o\} \times \{\bot\top^n \mid n \geqslant 0\}$. The automaton T_2 is length-reducible to $\mathrm{Vis}(\{c\}, \varnothing, \{a, b\})$. See Figure 1.

2.2 Mazurkiewicz traces

Given a (finite) alphabet Σ, recall that Σ^* is the free monoid of finite Σ-words.

A *dependence relation* D on Σ is a reflexive and symmetric binary relation. The pair (Σ, D) is called a *dependence alphabet*. The complement of D is the *independence relation* $I := \Sigma^2 \backslash D$. The (Σ, D)-trace equivalence $\equiv_D$ is the least congruence on Σ^* such that

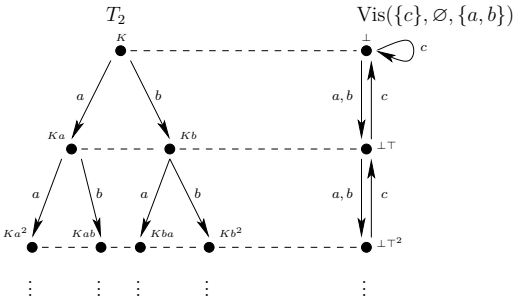

Figure 1 T_2 is length-reducible to $\mathrm{Vis}(\{c\},\varnothing,\{a,b\})$ (Example 2).

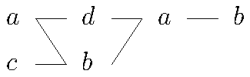

Figure 2 The Foata normal form of $[acbdab]$ (Example 4).

$(a,b) \in I \Rightarrow ab \equiv_D ba$. The (Σ, D)-trace of a word $w \in \Sigma^*$ is its $\equiv_D$-equivalence class. It is denoted $[w]$. Note that $\equiv_D$-equivalent Σ-words have the same length. The quotient monoid $\Sigma^*/\equiv_D$ is called the *trace monoid* of the dependence alphabet (Σ, D) and is denoted by $M(\Sigma, D)$. Its elements are called the *traces* over (Σ, D). The length of a trace t is the length of any word belonging to it and is denoted $|t|$. Denote by $\mathrm{Total}_\Sigma := \Sigma \times \Sigma$ the total dependence relation over Σ and by $\mathrm{Id}_\Sigma := \{(a,a) \mid a \in \Sigma\}$ the equality. Note that in case of $D = \mathrm{Total}_\Sigma$, the trace monoid $M(\Sigma, D)$ coincides with the free monoid Σ^*.

Consider the finite alphabet $I_D := \{A \subseteq \Sigma \mid \forall a_1 \neq a_2 \in A\ (a_1, a_2) \in I\}$ of independent subsets of Σ, and denote by $\Pi_{I_D} : I_D^* \to M(\Sigma, D)$ the canonical morphism defined by $\Pi_{I_D}(\varnothing) = [\varepsilon]$ and $\Pi_{I_D}(\{a_1, \cdots, a_n\}) = [a_1 \dots a_n]$ $(n \geqslant 1)$. Consider the binary relation $\rhd$ on $I_D^- := I_D \setminus \{\varnothing\}$ defined by: $A \rhd B \iff \forall b \in B\ \exists a \in A\ (a,b) \in D$. The set of I_D^{-*}-words of the form $A_1 \dots A_p$ $(p \geqslant 0)$ such that $A_1 \rhd A_2 \rhd \cdots \rhd A_p$ is denoted $\mathbf{F}$. The surjective morphism Π_{I_D} is not injective. Indeed, suppose $\Sigma = \{a,b\}$ and aIb, then $\Pi_{I_D}(\{a,b\}) = \Pi_{I_D}(\{a\}\{b\})$. The following proposition expresses that each trace is encodable by a unique I_D^--word in $\mathbf{F}$.

▶ **Proposition 3** (Foata normal form, [10]). *Let $t \in M(\Sigma, D)$. There exists a unique I_D^--word $\lceil t \rceil_{\mathbf{F}} = A_1 \cdots A_p \in \mathbf{F}$ $(p \geqslant 0)$, the Foata normal form of t, such that $\Pi_{I_D}(A_1 \cdots A_p) = t$.*

▶ **Example 4.** Suppose $\Sigma = \{a,b,c,d\}$ and aIc, bId, cId. The Foata normal form of $t = [acbdab]$ (see Figure 2) is $\lceil t \rceil_{\mathbf{F}} = \{a,c\}\{b,d\}\{a\}\{b\}$.

The following lemma is straightforward.

▶ **Lemma 5** (Level automata). *The set $\mathbf{F}$ of Foata normal forms is a regular word language over I_D^-.*

In general, $\lceil st \rceil_{\mathbf{F}}$ and $\lceil s \rceil_{\mathbf{F}} \lceil t \rceil_{\mathbf{F}}$ may be different. Indeed, suppose $D = \{(a,a),(b,b)\}$. If $s = [a]$ and $t = [ab]$, then $\lceil s \rceil_{\mathbf{F}} = \{a\}$, $\lceil t \rceil_{\mathbf{F}} = \{a,b\}$ and $\lceil st \rceil_{\mathbf{F}} = \{a,b\}\{a\}$. The following lemma expresses some compatibility between concatenation and Foata normal form.

▶ **Lemma 6.** *Let $s,t \in M(\Sigma, D)$ such that $\lceil s \rceil_{\mathbf{F}} = A_1 \cdots A_p$ $(p \geqslant 0)$. Then there exist $B_{i_1}, \dots B_{i_k} \in I_D^-$ $(0 \leqslant k \leqslant p$ and $1 \leqslant i_1 < \cdots < i_k \leqslant p)$ and $C_1, \dots, C_m \in I_D^-$ $(m \geqslant 0)$ such that $\lceil st \rceil_{\mathbf{F}} = A_1 \dots (A_{i_1} \dot\cup B_{i_1}) \dots (A_{i_k} \dot\cup B_{i_k}) \dots A_p C_1 \dots C_m$ and $\Pi_{I_D^-}(B_{i_1} \dots B_{i_k} C_1 \dots C_m) = t$.*

Proof. By induction on the length of t. ◀

In the following, given a trace t, we denote by $\|t\|$ the length of its Foata normal form.

A *trace automaton* is an automaton of which the vertices belong to some trace monoid $M(\Sigma, D)$ and accepts finite words (over an alphabet that may have no relation with Σ).

3 From word (suffix) automata to trace (suffix) automata

In this section, we recall how to obtain various Boolean algebras of deterministic context-free languages from word suffix automata [7]. Then we consider the notion of level-regular trace languages. This allows to define the family TrSuffix of trace suffix automata with level-regular contexts, an extension to traces of word suffix automata. Lastly, we will define (in terms of trace suffix automata) the vector addition systems from which we will obtain Boolean algebras (Section 5).

3.1 Boolean algebras from word suffix automata

Given a word language L over T, a non-empty family of languages $\mathcal{F}_L$ is a *Boolean algebra relative to* L if $L_1 \subseteq L$, $L - L_1 \in \mathcal{F}_L$ and $L_1 \cap L_2 \in \mathcal{F}_L$ for any $L_1, L_2 \in \mathcal{F}_L$. Observe that if $\mathcal{F}_L$ is a Boolean algebra relative to L then $\varnothing, L \in \mathcal{F}_L$.

A *word suffix automaton* is a finite union of automata of the form

$$W(u \xrightarrow{a} v) \ \cup \ \{\iota\} \times I \ \cup \ \{o\} \times F$$

where W, I and F are regular word languages (over a finite alphabet), u and v are words, $a \in T$ and $W(u \xrightarrow{a} v) = \{wu \xrightarrow{a} wv \mid w \in W\}$. We denote by Stack the family of word suffix automata. The languages accepted by Stack are the context-free languages [3].

In [7], Caucal and Rispal show how to obtain various Boolean algebras of deterministic context-free languages.

▶ **Theorem 7** ([7]). *Let H be a deterministic word suffix automaton of which the empty word is not a vertex. Then the class of languages $\mathrm{Rec}^{\ell}_{\mathrm{Stack}_{\mathrm{det}}}(H) = \{L(G) \mid G \in \mathrm{Stack}_{\mathrm{det}}, G \to_{\ell} H\}$ is a Boolean algebra relative to $L(H)$.*

In the following, we will consider the family TrSuffix of trace suffix automata with level-regular contexts, introduced in [16] (see Subsection 3.3). This family is an extension to Mazurkiewicz traces of word suffix automata (a word suffix automaton is just a trace suffix automaton with level-regular contexts over a trace monoid for which the dependence relation is total). We will also consider the subfamily $\mathrm{TrSuffix}^{\mathrm{VAS}}$ of vector addition systems and we will show how to obtain Boolean algebras from these, in the same vein as Theorem 7.

3.2 Level-regularity

In order to build the family TrSuffix of trace suffix automata with level-regular contexts, let us give some reminders about the notion of level-regular trace languages [16].

A (Σ, D)-*trace language* is a subset of $M(\Sigma, D)$. If $\mathcal{L}$ is a trace language, then $\bigcup \mathcal{L} = \{w \in \Sigma^* \mid [w] \in \mathcal{L}\}$. If L is a word language, then $[L]$ is the trace language defined by $[L] := \{[w] \in M(\Sigma, D) \mid w \in L\}$. In particular, $[\bigcup \mathcal{L}] = \mathcal{L}$ and $\bigcup [L] \supseteq L$.

Given a trace language $\mathcal{L} \subseteq M(\Sigma, D)$ and a trace $t \in M(\Sigma, D)$, the *right residual* (respectively the *left residual*) of $\mathcal{L}$ by t, $\mathcal{L}t^{-1}$ (respectively $t^{-1}\mathcal{L}$), is $\mathcal{L}t^{-1} := \{s \in M(\Sigma, D) \mid st \in \mathcal{L}\}$ (respectively $t^{-1}\mathcal{L} := \{s \in M(\Sigma, D) \mid ts \in \mathcal{L}\}$). The product of $\mathcal{L}$ by t (respectively the product of t by $\mathcal{L}$), $\mathcal{L}t$ (respectively $t\mathcal{L}$), is $\mathcal{L}t := \{st \in M(\Sigma, D) \mid s \in \mathcal{L}\}$ (respectively $t\mathcal{L} := \{ts \in M(\Sigma, D) \mid s \in \mathcal{L}\}$). If $\mathcal{L}_1, \mathcal{L}_2 \subseteq M(\Sigma, D)$ are trace languages, their product is the trace language $\mathcal{L}_1\mathcal{L}_2 := \{t_1t_2 \mid t_1 \in \mathcal{L}_1, t_2 \in \mathcal{L}_2\}$. In particular, $\mathcal{L}\varnothing = \varnothing\mathcal{L} = \varnothing$.

A trace language $\mathcal{L} \subseteq M(\Sigma, D)$ is *recognizable* if there exists a finite monoid N and a monoid morphism $\phi : M(\Sigma, D) \to N$ such that $\mathcal{L} = \phi^{-1}(\phi(\mathcal{L}))$. The class of recognizable trace languages is denoted by $\mathrm{Rec}(M(\Sigma, D))$.

▶ **Proposition 8** ([10]). *The following are equivalent:*
- *$\mathcal{L}$ is recognizable,*
- *$\bigcup \mathcal{L}$ is a regular word language,*
- *the set $\{t^{-1}\mathcal{L} \mid t \in M(\Sigma, D)\}$ of left residuals of $\mathcal{L}$ is finite,*
- *the set $\{\mathcal{L}t^{-1} \mid t \in M(\Sigma, D)\}$ of right residuals of $\mathcal{L}$ is finite.*

▶ Remark 9. In case of $D = \mathrm{Total}_\Sigma$, $\mathrm{Rec}(M(\Sigma, D)) = \mathrm{Reg}(\Sigma^*)$.

▶ **Proposition 10** ([10]). $\mathrm{Rec}(M(\Sigma, D))$ *is a Boolean algebra closed under concatenation.*

The trace language $[(ab)^*]$ with aIb is not recognizable since its union, the set of words over $\{a, b\}$ with the same number of occurences of a and b, is not regular. Nevertheless, the set $\{a, b\}^*$ of Foata normal forms of its elements is regular. This suggests considering a weaker notion of recognizability.

▶ **Definition 11.** *$\mathcal{L} \subseteq M(\Sigma, D)$ is level-regular if the word language $\lceil \mathcal{L} \rceil_{\mathbf{F}}$ is regular.*

Since $\lceil \mathcal{L} \rceil_{\mathbf{F}} = \Pi_{I_D}^{-1}(\mathcal{L}) \cap \mathbf{F}$, any recognizable trace language is level-regular. Note also that any level-regular trace language is a rational subset of the monoid $M(\Sigma, D)$. Denote by $\mathrm{LevelReg}(M(\Sigma, D))$ the class of level-regular languages of the trace monoid $M(\Sigma, D)$.

▶ **Proposition 12** ([16]). $\mathrm{LevelReg}(M(\Sigma, D))$ *is a Boolean algebra.*

The class $\mathrm{LevelReg}(M(\Sigma, D))$ is not closed under concatenation [16]. Nevertheless, multiplication and residuation of trace languages on the right by a trace preserves the level-regularity.

▶ **Lemma 13.** *Let $\mathcal{L} \in \mathrm{LevelReg}(M(\Sigma, D))$ and $t \in M(\Sigma, D)$. The trace languages $\mathcal{L}t^{-1}$ and $\mathcal{L}t$ are level-regular.*

Given a word language L, we denote by $\mathrm{Pref}(L) := \{u \mid \exists\ v\ uv \in L\}$ the set of prefixes of words in L. From the lemma below, it follows, in particular, that the set of prefixes of Foata normal forms of a level-regular trace language is a regular word language.

▶ **Lemma 14.** *Let $\mathcal{L} \in \mathrm{LevelReg}(M(\Sigma, D))$. Then $\Pi_{I_D}(\mathrm{Pref} \lceil \mathcal{L} \rceil_{\mathbf{F}}) \in \mathrm{LevelReg}(M(\Sigma, D))$.*

3.3 Trace suffix automata with level-regular contexts

We consider trace suffix automata with level-regular contexts [16]. These generalize both word suffix automata (see Subsection 3.1) and vector addition systems (see the following subsection).

A *trace suffix automaton (with level-regular contexts)* over a trace monoid $M(\Sigma, D)$ is of the form

$$G = \bigcup_{1 \leqslant i \leqslant n} \mathcal{W}_i(u_i \xrightarrow{a_i} v_i) \ \cup\ \{\iota\} \times \mathcal{I}_G \ \cup\ \{o\} \times \mathcal{F}_G$$

where $\mathcal{W}_i, \mathcal{I}_G, \mathcal{F}_G \in \mathrm{LevelReg}(M(\Sigma, D)), u_i, v_i \in M(\Sigma, D)$, $a_i \in T$ and $\mathcal{W}_i(u_i \xrightarrow{a_i} v_i) = \{w_i u_i \xrightarrow{a_i} w_i v_i \mid w_i \in \mathcal{W}_i\}$ $(1 \leqslant i \leqslant n)$. The trace language $\mathcal{W}_i$ is called the *context* of the *rewriting rule* $\mathcal{W}_i(u_i \xrightarrow{a_i} v_i)$ $(1 \leqslant i \leqslant n)$. Denote by TrSuffix the family of trace suffix automata.

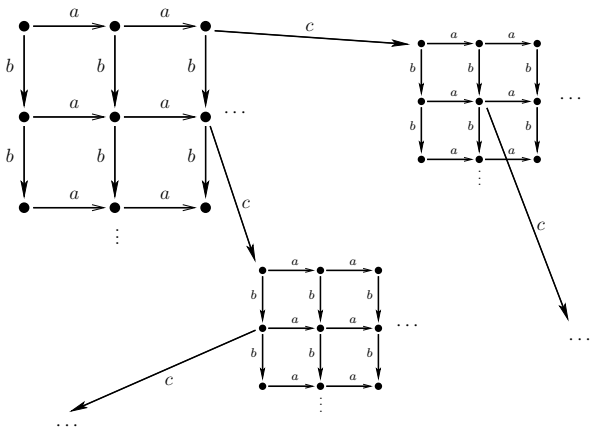

Figure 3 The infinite quarter grid tree is a trace suffix automaton (see Example 15).

▶ **Example 15** (Infinite quarter grid tree)**.** See Figure 3. Consider the following trace suffix automaton with level-regular contexts, where the independence relation is $\{(a,b),(b,a)\}$.

$[\bot\{a,b,c\}^*]([\varepsilon] \xrightarrow{a} [a]) \cup [\bot\{a,b,c\}^*]([\varepsilon] \xrightarrow{b} [b]) \cup [\bot\{a,b,c\}^*]([\varepsilon] \xrightarrow{c} [c])$

In particular, it was shown in [16] that its first-order theory with reachability is decidable though it is not a ground term rewriting graph [15].

The trace suffix automata for total dependence relations are the word suffix automata. Thus regular languages and context-free languages are accepted by trace suffix automata. On the other hand:

▶ **Proposition 16** ([16, 20])**.** *The languages accepted by the trace suffix automata are context-sensitive.*

The previous proposition relies on the fact that trace suffix automata are actually word-automatic automata [16] and the latter accept the context-sensitive languages [20].

3.4 Vector addition systems

Here, a *vector addition system* G over Σ is a trace suffix automaton over the trace monoid $M(\Sigma, \mathrm{Id}_\Sigma)$. Denote by $\mathrm{TrSuffix}^{\mathrm{VAS}}$ the family of vector addition systems. The family $\mathrm{TrSuffix}^{\mathrm{VAS}}$ is a subfamily of TrSuffix.

▶ **Example 17.** Observe that a vector addition system over a singleton alphabet is a word suffix automaton since the equality relation over such an alphabet coincides with the total relation. Let (T_{-1}, T_0, T_1) a triple of disjoint finite alphabets. The trace suffix automaton $\mathrm{Vis}^{\mathrm{VAS} \vee \mathrm{Stack}}(T_{-1}, T_0, T_1)$ over $M(\{\top\}, \{(\top,\top)\})$ defined below is a vector addition system over the singleton alphabet $\{\top\}$. It is length-isomorphic to a word suffix automaton. The Boolean algebra $\mathrm{Rec}^{\ell}_{\mathrm{Stack}_{\mathrm{det}}}(\mathrm{Vis}^{\mathrm{VAS} \vee \mathrm{Stack}}(T_{-1}, T_0, T_1))$ (see Theorem 7) is the family of visibly pushdown languages with respect to (T_{-1}, T_0, T_1) ([1]).

$$\mathrm{Vis}^{\mathrm{VAS} \vee \mathrm{Stack}}(T_{-1}, T_0, T_1) := \bigcup_{\lambda \in T_1} [\top^+]([\varepsilon] \xrightarrow{\lambda} [\top]) \dot\cup \bigcup_{\lambda \in T_0} [\top^+]([\varepsilon] \xrightarrow{\lambda} [\varepsilon])$$
$$\dot\cup \bigcup_{\lambda \in T_{-1}} [\top^+]([\top] \xrightarrow{\lambda} [\varepsilon]) \dot\cup \bigcup_{\lambda \in T_{-1}} [\top]([\varepsilon] \xrightarrow{\lambda} [\varepsilon]) \dot\cup \{\iota\} \times \{[\top]\} \dot\cup \{o\} \times [\top^+]$$

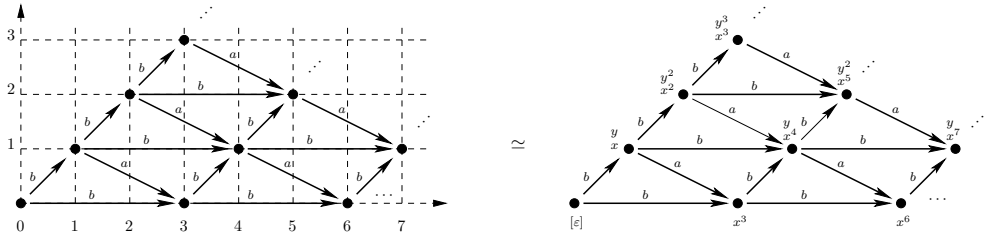

Figure 4 A vector addition system from a finite set of integer vectors.

In the previous example, the length variation of two adjacent vertices is almost 1. It is no more the case for the following example that shows, in particular, how to obtain a vector addition system from a finite set of integer vectors of the same dimension.

▶ **Example 18.** Consider the finite set $A := \left\{\binom{2}{-1}, \binom{1}{1}, \binom{3}{0}\right\}$ of 2-dimensional integer vectors and the automaton G_A over $\mathbb{N}^2$, defined by

$$G_A := \{v \xrightarrow{l(a)} v' \mid v, v' \in \mathbb{N}^2, a \in A, v + a = v'\}$$

where $l : A \longrightarrow T$ is the labeling function that maps $\binom{1}{1}$ and $\binom{3}{0}$ to the terminal b and $\binom{2}{-1}$ to the terminal a. The automaton G_A is isomorphic to the vector addition system over the alphabet $\{x, y\}$ defined by the following rules

$$M([y] \xrightarrow{a} [xx]) \ \cup \ M([\varepsilon] \xrightarrow{b} [xy]) \ \cup \ M([\varepsilon] \xrightarrow{b} [xxx])$$

where M denotes the trace monoid $M(\{x, y\}, \mathrm{Id}_{\{x,y\}})$. See Figure 4.

4 Level-length synchronization and Boolean algebras from trace automata

In this section, we show how to obtain various Boolean algebras of word languages from the family TrSuffix (Theorem 37). Actually, this will be deduced from a more general result, since we give sufficient conditions for any family of trace automata to define Boolean algebras (Theorem 34). After considering a notion of synchronization of two traces belonging to disjoint trace monoids, we define their level-length synchronization. Then we define the level-length synchronization and superposition of two trace automata (Definition 23 and 27) and we show that under some relevant assumptions, they accept respectively the intersection and the difference (Lemmas 25 and 33). As a consequence, we show how to obtain a Boolean algebra of word languages from any trace automata family closed under level-length synchronization and superposition and from a deterministic automaton in this family (Theorem 34). Finally, we show that the family TrSuffix is closed under level-length synchronization and superposition.

Let $M(\Sigma_1, D_1)$ and $M(\Sigma_2, D_2)$ be trace monoids such that $\Sigma_1 \cap \Sigma_2 = \varnothing$.

Let $s \in M(\Sigma_1, D_1)$ and $t \in M(\Sigma_2, D_2)$.

▶ **Definition 19.** *The synchronization $s \parallel t$ of s and t is the product st in the trace monoid $M(\Sigma_1 \dot\cup \Sigma_2, D_1 \ \cup \ D_2)$.*

▶ **Definition 20.** *The level-length synchronization $s \parallel_= t$ of s and t is $s \parallel t$ if $\|s\| = \|t\|$ and is not defined otherwise.*

We also define $s \parallel_{\geqslant} t := s \parallel t$ if $\|s\| \geqslant \|t\|$ and $s \parallel_{\leqslant} t := s \parallel t$ if $\|s\| \leqslant \|t\|$.

Given $\mathcal{L}_1 \subseteq M(\Sigma_1, D_1)$ and $\mathcal{L}_2 \subseteq M(\Sigma_2, D_2)$, we define
$\mathcal{L}_1 \parallel \mathcal{L}_2 := \{t_1 \parallel t_2 \mid t_1 \in \mathcal{L}_1, t_2 \in \mathcal{L}_2\}$ and $\mathcal{L}_1 \parallel_{=} \mathcal{L}_2 := \{t_1 \parallel_{=} t_2 \mid t_1 \in \mathcal{L}_1, t_2 \in \mathcal{L}_2\}$,
$\mathcal{L}_1 \parallel_{\geqslant} \mathcal{L}_2 := \{t_1 \parallel_{\geqslant} t_2 \mid t_1 \in \mathcal{L}_1, t_2 \in \mathcal{L}_2\}$ and $\mathcal{L}_1 \parallel_{\leqslant} \mathcal{L}_2 := \{t_1 \parallel_{\leqslant} t_2 \mid t_1 \in \mathcal{L}_1, t_2 \in \mathcal{L}_2\}$.
These synchronized trace languages remain level-regular if $\mathcal{L}_1$ and $\mathcal{L}_2$ are.

▶ **Lemma 21.** *Let $\mathcal{L}_1 \in \mathrm{LevelReg}(M(\Sigma_1, D_1))$ and $\mathcal{L}_2 \in \mathrm{LevelReg}(M(\Sigma_2, D_2))$. The following trace languages are level-regular: $\mathcal{L}_1 \parallel \mathcal{L}_2$, $\mathcal{L}_1 \parallel_{\geqslant} \mathcal{L}_2$, $\mathcal{L}_1 \parallel_{\leqslant} \mathcal{L}_2$, $\mathcal{L}_1 \parallel_{=} \mathcal{L}_2$.*

If a synchronized trace language is level-regular, then so are its projections. Let us make explicit what such a projection is. For $i \in \{1, 2\}$, denoting $\bar{i} := 3 - i$, we consider the morphism π_i from $M(\Sigma_1 \dot\cup \Sigma_2, D_1 \dot\cup D_2)$ into $M(\Sigma_i, D_i)$ defined by $\pi_i([a]) = [a]$ if $a \in \Sigma_i$ and $\pi_i([a]) = [\varepsilon]$ if $a \in \Sigma_{\bar{i}}$. Given $t \in M(\Sigma_1 \dot\cup \Sigma_2, D_1 \dot\cup D_2)$, there exists a unique couple $(t_1, t_2) \in M(\Sigma_1, D_1) \times M(\Sigma_2, D_2)$ such that $t = t_1 t_2$. This couple is given by $t_1 = \pi_1(t)$ and $t_2 = \pi_2(t)$. The morphism π_i naturally extends to trace languages.

▶ **Lemma 22.** *Let $\mathcal{L} \in \mathrm{LevelReg}(M(\Sigma_1 \dot\cup \Sigma_2, D_1 \dot\cup D_2))$. The trace languages $\pi_1(\mathcal{L})$ and $\pi_2(\mathcal{L})$ are level-regular.*

Now, we consider the level-length synchronization of two automata. It is an automaton over the level-length synchronization of vertices of these automata.

▶ **Definition 23.** *Let G_1 and G_2 be trace automata over respectively $M(\Sigma_1, D_1)$ and $M(\Sigma_2, D_2)$. Their level-length synchronization $G_1 \parallel_{=} G_2$ is the trace automaton over $M(\Sigma_1 \dot\cup \Sigma_2, D_1 \dot\cup D_2)$ defined by*

$$G_1 \parallel_{=} G_2 := \{p_1 \parallel_{=} p_2 \xrightarrow{a} q_1 \parallel_{=} q_2 \mid p_1 \xrightarrow[G_1]{a} q_1, p_2 \xrightarrow[G_2]{a} q_2\} \cup \{\iota\} \times (I_{G_1} \parallel_{=} I_{G_2}) \cup \{o\} \times (F_{G_1} \parallel_{=} F_{G_2})$$

▶ **Lemma 24.** $G_1 \parallel_{=} G_2 \xrightarrow{\pi_i}_{\ell} G_i$ $(i \in \{1, 2\})$.

If two automata are length-reducible to a same deterministic one, then the intersection of their languages is accepted by their level-length synchronization.

▶ **Lemma 25.** *If H is a deterministic trace automaton such that $G_1 \to_{\ell} H$ and $G_2 \to_{\ell} H$, then $L(G_1 \parallel_{=} G_2) = L(G_1) \cap L(G_2)$.*

Proof. Let $u \in L(G_1) \cap L(G_2)$ be a word. By Lemma 1 and since $G_1 \to_{\ell} H$, there exists an accepting path labeled by u in H. Since H is deterministic, this path is unique. Then, for any two paths in G_1 and G_2 accepting u, the sequences of lengths of vertices of these paths are same, because they are the same as the sequence of lengths of the path accepting u in H. Hence, u labels an accepting path in $G_1 \parallel_{=} G_2$. The other inclusion is straightforward. ◀

If two deterministic automata are length-reducible to a same deterministic one, then their level-length synchronization is also a deterministic automaton.

▶ **Lemma 26.** *If G_1, G_2 and H are deterministic trace automata such that $G_i \to_{\ell} H$ $(i \in \{1, 2\})$, then $G_1 \parallel_{=} G_2$ is a deterministic automaton.*

We are now introducing the level-length superposition $G \mathbin{/\!\!/^{\sharp}} H$ of trace automata. When restricted to deterministic automata and if $G \to_{\ell} H$, this accepts the difference language $L(H) - L(G)$ (Lemma 33). Note that to accept the difference, we may consider simpler operations. The level-length superposition bears the advantage of preserving the boundedness

of the degree. Indeed, we will consider families of automata of bounded degree and in order to obtain Boolean algebras, we will require these families to be closed under level-length superposition.

Given a word language L and a word u, we write $u \sqsubseteq L$ if u is a prefix of a word in L. A trace automaton is $[\varepsilon]$-*free* if $[\varepsilon]$ is not a vertex.

▶ **Definition 27.** *Let G and H be $[\varepsilon]$-free trace automata over respectively $M(\Sigma_1, D_1)$ and $M(\Sigma_2, D_2)$ and $\sharp \notin \Sigma_1 \dot\cup \Sigma_2$. The level-length superposition of G on H is the trace automaton $G /\!\!/^\sharp H$ over $M(\Sigma_1 \dot\cup \{\sharp\} \dot\cup \Sigma_2, D_1 \dot\cup \{(\sharp,\sharp)\} \dot\cup D_2)$ defined by*

$$
\begin{aligned}
& G /\!\!/^\sharp H := \\
(1)\quad & \{p \parallel_= s \xrightarrow{a} q \parallel_= t \mid p \xrightarrow[G]{a} q, s \xrightarrow[H]{a} t\} \\
(2)\quad & \cup \{\iota\} \times (I_G \parallel_= I_H) \\
(3)\quad & \cup \{o\} \times ((V_G - F_G) \parallel_= F_H) \\
(4)\quad & \cup \{p \parallel_= s \xrightarrow{a} p[\sharp] \parallel t \mid s \xrightarrow[H]{a} t, \|s\| \leqslant \|t\|, p \in V_G, \neg\exists p'(p \xrightarrow[G]{a} p' \ \wedge \|p'\| = \|t\|)\} \\
(5)\quad & \cup \{p \parallel_= s \xrightarrow{a} q[\sharp] \parallel_= t \mid s \xrightarrow[H]{a} t, \|s\| > \|t\|, p \in V_G, \neg\exists p'(p \xrightarrow[G]{a} p' \ \wedge \|p'\| = \|t\|), \lceil q\rceil_{\mathbf{F}} \sqsubseteq \lceil p\rceil_{\mathbf{F}}\} \\
(6)\quad & \cup \{p[\sharp] \parallel_\leqslant s \xrightarrow{a} p[\sharp] \parallel t \mid s \xrightarrow[H]{a} t, \|s\| \leqslant \|t\|, \lceil p\rceil_{\mathbf{F}} \sqsubseteq \lceil V_G\rceil_{\mathbf{F}}\} \\
(7)\quad & \cup \{p[\sharp] \parallel_\leqslant s \xrightarrow{a} p[\sharp] \parallel_\leqslant t \mid s \xrightarrow[H]{a} t, \|s\| > \|t\|, \lceil p\rceil_{\mathbf{F}} \sqsubseteq \lceil V_G\rceil_{\mathbf{F}}\} \\
(8)\quad & \cup \{p[\sharp] \parallel_\leqslant s \xrightarrow{a} q[\sharp] \parallel_= t \mid s \xrightarrow[H]{a} t, \|s\| > \|t\|, \|t\| < \|p[\sharp]\|, \lceil p\rceil_{\mathbf{F}} \sqsubseteq \lceil V_G\rceil_{\mathbf{F}}, \lceil q\rceil_{\mathbf{F}} \sqsubseteq \lceil p\rceil_{\mathbf{F}}\} \\
(9)\quad & \cup \{o\} \times (\{p[\sharp] \parallel_\leqslant s \mid \lceil p\rceil_{\mathbf{F}} \sqsubseteq \lceil V_G\rceil_{\mathbf{F}}, s \in F_H\}) \\
(10)\quad & \cup \{\iota\} \times (\{[\sharp] \parallel s \mid s \in I_H, \neg(\exists p \in I_G \ \|p\| = \|s\|)\})
\end{aligned}
$$

Let us give some more explanations about this definition. First, observe that there are two kinds of vertices: those in which $\sharp$ occurs and the other ones. Then, observe that the edges between non $\sharp$-vertices are those of $G \parallel_= H$. Note that the edges at lines (4) and (5) are the only ones between $\sharp$-vertices and non $\sharp$-vertices, and that they leave the set $V_{G\parallel_= H}$ of non $\sharp$-vertices. Thus, once a path leaves $V_{G\parallel_= H}$, it never returns.

Given $t \in M(\Sigma_1 \dot\cup \{\sharp\} \dot\cup \Sigma_2, D_1 \dot\cup D_2)$, we denote by $\pi_1(t)$ (respectively $\pi_2(t)$) the unique trace in $M(\Sigma_1 \dot\cup \{\sharp\}, D_1)$ (respectively $M(\Sigma_2, D_2)$) such that $t = \pi_1(t)\pi_2(t)$.

For each edge $s \xrightarrow[H]{a} t$ and each vertex x of $G /\!\!/^\sharp H$, H-projecting on s (*i.e.*, $\pi_2(x) = s$), there exists an edge $x \xrightarrow[G/\!\!/^\sharp H]{a} y$ with y H-projecting on t (*i.e.*, $\pi_2(y) = t$). Since on the other hand, any initial vertex of H lifts to an initial vertex of $G /\!\!/^\sharp H$ (lines (2) and (10)), it follows that any initial path in H (an *initial path* is a path of which the source is an initial vertex) lifts to an initial path of $G /\!\!/^\sharp H$.

▶ **Lemma 28.** *Any word that labels an initial path in H, also labels an initial path in $G /\!\!/^\sharp H$.*

Lastly, note that we preserve the fact that the level-length of any vertex is given by the level-length of its H-component (see Lemma 30). In particular, the first component is not longer than the second one.

It will be relevant to discuss about the final vertices of $G /\!\!/^\sharp H$ only under some more assumptions about G and H (Lemma 30, Corollary 32 and Lemma 33).

▶ **Example 29.** Let us consider the automata described in Example 2. The unique initial vertex of $T_2 /\!\!/^\sharp \mathrm{Vis}(\{c\}, \varnothing, \{a, b\})$ is $K \parallel_= \bot$. The final vertices of $T_2 /\!\!/^\sharp \mathrm{Vis}(\{c\}, \varnothing, \{a, b\})$ are these for which the number of occurrences of $\sharp$ is 1. See Figure 5 for the restriction of $T_2 /\!\!/^\sharp \mathrm{Vis}(\{c\}, \varnothing, \{a, b\})$ to the vertices accessible from the initial vertex and co-accessible from final vertices.

▶ **Lemma 30.** $G /\!\!/^\sharp H \xrightarrow{\pi_2}_\ell H$.

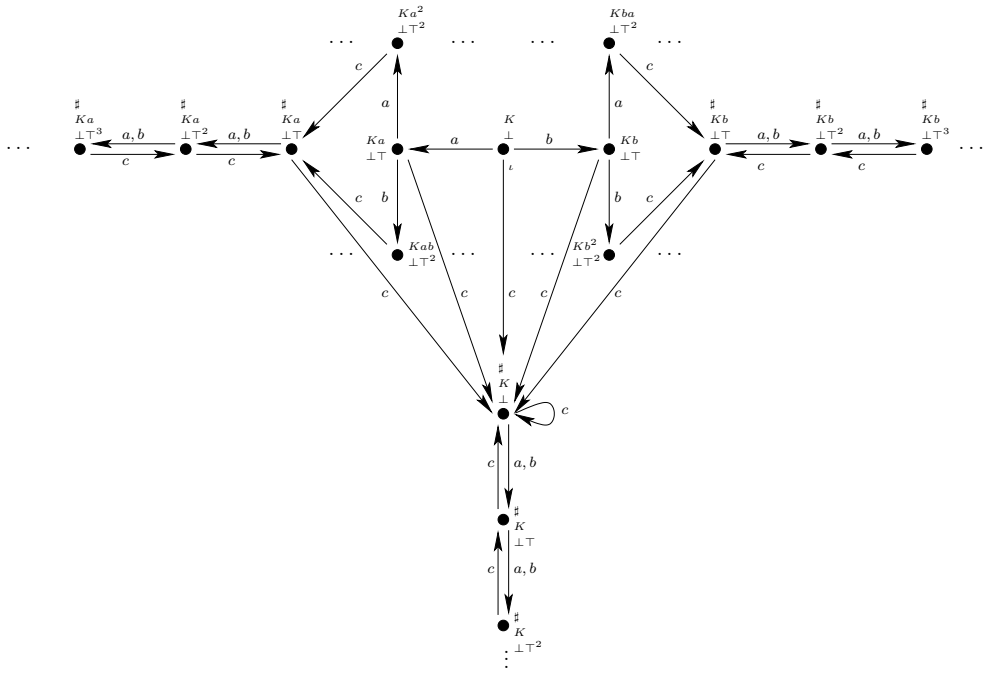

Figure 5 Trimmed level-length superposition of T_2 on $\mathrm{Vis}(\{c\}, \varnothing, \{a, b\})$ (see Example 29).

By Lemma 1, $L(G \mathbin{/\!\!/^\sharp} H) \subseteq L(H)$. That is, if a word labels an accepting path in $G \mathbin{/\!\!/^\sharp} H$ then it also labels an accepting path in H. The converse is not true in general, as we will see in Lemma 33.

The level-length superposition preserves determinism.

▶ **Lemma 31.** *If G and H are deterministic, then $G \mathbin{/\!\!/^\sharp} H$ is deterministic.*

Observe that the automaton described at lines (1), (2) and (3) (Definition 27) is the level-length synchronization of H and the automaton obtained from G by declaring final the non final vertices and *vice-versa*. By Lemma 25 and since obviously $H \to_\ell H$, we deduce the following corollary.

▶ **Corollary 32.** *If G and H are deterministic, $G \to_\ell H$ and w is a word that labels an initial path in $G \parallel_= H$, then $w \in L(G) \cap L(H) \iff w \notin L(G \mathbin{/\!\!/^\sharp} H)$.*

The level-length superposition accepts the difference.

▶ **Lemma 33.** *If G and H are deterministic and $G \to_\ell H$, then $L(G \mathbin{/\!\!/^\sharp} H) = L(H) - L(G)$.*

Proof. By Lemmas 30, 31 and Corollary 32, it remains to show that any word $w \in L(H) - L(G)$ that does not label any initial path in $G \parallel_= H$ is accepted by $G \mathbin{/\!\!/^\sharp} H$. The unique initial path in the deterministic automaton $G \mathbin{/\!\!/^\sharp} H$ labeled by w is accepting since its goal is of the form $p[\sharp] \parallel_\leqslant s$ $(s \in F_H)$ and any such vertex is final in $G \mathbin{/\!\!/^\sharp} H$ (line (9) in Definition 27). ◀

Let us apply Lemmas 25 and 33. From any trace automata family closed under level-length synchronization and superposition, we obtain various Boolean algebras of word languages from the deterministic automata of this family.

▶ **Theorem 34.** *Let $\mathcal{F}$ be a family of trace automata closed under level-length synchronization and superposition and $H \in \mathcal{F}_{\mathrm{det}}$ $[\varepsilon]$-free. Then the class of languages $\mathrm{Rec}^\ell_{\mathcal{F}_{\mathrm{det}}}(H) = \{L(G) \mid G \in \mathcal{F}_{\mathrm{det}}, G \to_\ell H\}$ is a Boolean algebra relative to $L(H)$.*

Let us show that Theorem 34 applies to the family TrSuffix.

▶ **Proposition 35.** *The family* TrSuffix *is closed under level-length synchronization.*

Proof (Sketch). First, observe that given $G_1, G_2 \in \mathrm{TrSuffix}$, the sets of initial vertices and final vertices of $G_1 \parallel_= G_2$ are level-regular by Lemma 21. Indeed, the level-length synchronization of level-regular languages remains level-regular. Then, since the operation $\parallel_=$ is distributive over union, it suffices to suppose that G_1 (respectively G_2) is of the form $\mathcal{W}_1(u_1 \xrightarrow{a} v_1)$ over $M(\Sigma_1, D_1)$ (respectively $\mathcal{W}_2(u_2 \xrightarrow{b} v_2)$ over $M(\Sigma_2, D_2)$) with $\mathcal{W}_i \in \mathrm{LevelReg}(M(\Sigma_i, D_i)), u_i, v_i \in M(\Sigma_i, D_i)$ $(i \in \{1, 2\})$. If $a \neq b$, then $G_1 \parallel_= G_2 = \varnothing$. Suppose $a = b$. We show that $G_1 \parallel_= G_2 := \{p_1 \parallel_= p_2 \xrightarrow{a} q_1 \parallel_= q_2 \mid p_1 \xrightarrow[G_1]{a} q_1, p_2 \xrightarrow[G_2]{a} q_2\}$ is equal to $\big((\mathcal{W}_1 u_1 \parallel_= \mathcal{W}_2 u_2)(u_1 \parallel u_2)^{-1} \cap (\mathcal{W}_1 v_1 \parallel_= \mathcal{W}_2 v_2)(v_1 \parallel v_2)^{-1}\big)(u_1 \parallel u_2 \xrightarrow{a} v_1 \parallel v_2)$. Since $(\mathcal{W}_1 u_1 \parallel_= \mathcal{W}_2 u_2)(u_1 \parallel u_2)^{-1} \cap (\mathcal{W}_1 v_1 \parallel_= \mathcal{W}_2 v_2)(v_1 \parallel v_2)^{-1}$ is level-regular by Lemmas 13, 21 and Proposition 12, this proves the proposition. ◀

▶ **Proposition 36.** *The family* TrSuffix *is closed under level-length superposition.*

Proof (Sketch). Since $G /\!\!/^\sharp (H_1 \cup H_2) = G /\!\!/^\sharp H_1 \cup G /\!\!/^\sharp H_2$, it suffices to consider $G = \bigcup_{1 \leqslant i \leqslant n} \mathcal{W}_i(u_i \xrightarrow{a_i} v_i) \cup \{\iota\} \times \mathcal{I}_G \cup \{o\} \times \mathcal{F}_G$ over $M(\Sigma_1, D_1)$ and $H = \mathcal{Z}(x \xrightarrow{a} y) \cup \{\iota\} \times \mathcal{I}_H \cup \{o\} \times \mathcal{F}_H$ over $M(\Sigma_2, D_2)$. We may assume that the level-length variation of the rules $\mathcal{W}_i(u_i \xrightarrow{a_i} v_i)$ $(1 \leqslant i \leqslant n)$ and $\mathcal{Z}(x \xrightarrow{a} y)$ is constant and is equal respectively to δ_i and δ_H. We show that the sets of initial and final vertices of $G /\!\!/^\sharp H$ correspond to level-regular trace languages. Then, by distinguishing the cases $\delta_H \geqslant 0$ and $\delta_H < 0$, we show that the sets of edges constituting $G /\!\!/^\sharp H$ (Definition 27) can be described as trace suffix automata with level-regular contexts. In each case, the level-regularity of the trace languages considered (for initial and final vertices and for the contexts of the rewriting rules) will be ensured by Proposition 12 and Lemmas 13, 14, 21, 22. ◀

Let us apply Theorem 34 and Propositions 35 and 36.

▶ **Theorem 37.** *Let H be an $[\varepsilon]$-free deterministic trace suffix automaton. Then $\mathrm{Rec}^{\ell}_{\mathrm{TrSuffix}_{\mathrm{det}}}(H) = \{L(G) \mid G \in \mathrm{TrSuffix}_{\mathrm{det}}, G \to_\ell H\}$ is a Boolean algebra relative to $L(H)$.*

5 Vector addition systems

By Theorem 34 and since we will prove that $\mathrm{TrSuffix}^{\mathrm{VAS}}$ is closed under level-length synchronization and superposition, we obtain various Boolean algebras of word languages accepted by deterministic vector addition systems.

▶ **Theorem 38.** *Let $H \in \mathrm{TrSuffix}^{\mathrm{VAS}}_{\mathrm{det}}$ $[\varepsilon]$-free. Then $\mathrm{Rec}^{\ell}_{\mathrm{TrSuffix}^{\mathrm{VAS}}_{\mathrm{det}}}(H) = \{L(G) \mid G \in \mathrm{TrSuffix}^{\mathrm{VAS}}_{\mathrm{det}}, G \to_\ell H\}$ is a Boolean algebra relative to $L(H)$.*

Proof. Given Theorem 34, it suffices to show that $\mathrm{TrSuffix}^{\mathrm{VAS}}$ is closed under level-length synchronization and superposition. Consider $G_1, G_2 \in \mathrm{TrSuffix}^{\mathrm{VAS}}$ with G_1 over Σ_1, G_2 over Σ_2, $\Sigma_1 \cap \Sigma_2 = \varnothing$ and $\sharp \notin \Sigma_1 \dot\cup \Sigma_2$. Then observe that $G_1 \parallel_= G_2$ (respectively $G_1 /\!\!/^\sharp G_2$) is a trace suffix automaton over $\Sigma_1 \dot\cup \Sigma_2$ (respectively over $\Sigma_1 \dot\cup \{\sharp\} \dot\cup \Sigma_2$). ◀

▶ **Example 39.** The Boolean algebra $\mathrm{Rec}^{\ell}_{\mathrm{TrSuffix}^{\mathrm{VAS}}_{\mathrm{det}}}(\mathrm{Vis}^{\mathrm{VAS} \vee \mathrm{Stack}}(\varnothing, \varnothing, \{a\}))$ is the Boolean algebra of regular languages over the singleton alphabet $\{a\}$.

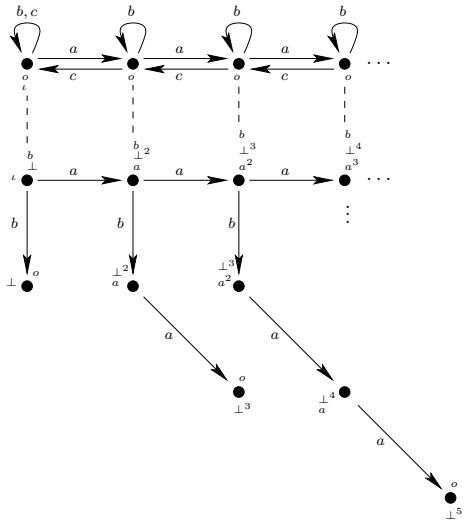

Figure 6 A trace suffix automaton that accepts $\{a^n b a^n \mid n \geqslant 0\}$ (Example 40).

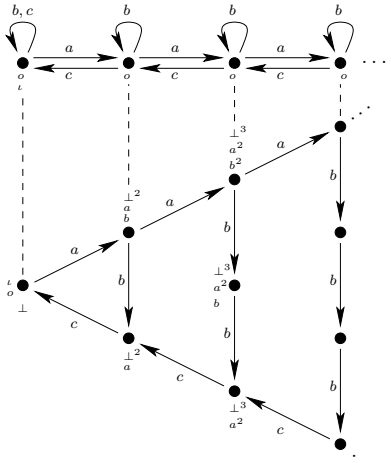

Figure 7 A trace suffix automaton that accepts $\{a^n b^n c^n \mid n \geqslant 0\}^*$ (Example 41).

▶ **Example 40.** It is well-known that the language $L = \{a^n b a^n \mid n \geqslant 0\}$, while being context-free, is not a visibly pushdown language [1]. Consider the following vector addition system G over $\{\bot, a, b\}$ defined by

$$G := [(\bot a)^* \bot]([b] \xrightarrow{b} [\varepsilon]) \cup [b(\bot a)^* \bot]([\varepsilon] \xrightarrow{a} [\bot a]) \cup [(\bot a)^* \bot^+]([a] \xrightarrow{a} [\bot]) \cup \{\iota\} \times \{[\bot b]\} \cup \{o\} \times [\bot^*]$$

We have $L = L(G) \in \mathrm{Rec}^{\ell}_{\mathrm{TrSuffix}^{\mathrm{VAS}}_{\mathrm{det}}}(\mathrm{Vis}^{\mathrm{VAS} \vee \mathrm{Stack}}(\varnothing, \{b\}, \{a\}))$. See Figure 6.

▶ **Example 41.** It is well-known that the language $L = \{a^n b^n c^n \mid n \geqslant 0\}^*$ is not context-free. Consider the following vector addition system G over $\{\bot, a, b\}$ defined by

$$G := [(\bot ab)^* \bot]([\varepsilon] \xrightarrow{a} [\bot ab]) \cup [(\bot ab)^* (\bot a)^+ \bot]([b] \xrightarrow{b} [\varepsilon]) \cup [(\bot a)^* \bot]([\bot a] \xrightarrow{c} [\varepsilon]) \cup \{\iota\} \times \{[\bot]\} \cup \{o\} \times \{[\bot]\}$$

We have $L = L(G) \in \mathrm{Rec}^{\ell}_{\mathrm{TrSuffix}^{\mathrm{VAS}}_{\mathrm{det}}}(\mathrm{Vis}^{\mathrm{VAS} \vee \mathrm{Stack}}(\{c\}, \{b\}, \{a\}))$. See Figure 7.

6 Conclusion

Summing up, we have shown how to obtain various Boolean algebras from any family of trace automata closed under level-length synchronization and superposition. The family TrSuffix of trace suffix automata with level-regular contexts and the subfamily $\mathrm{TrSuffix}^{\mathrm{VAS}}$ of vector addition systems satisfy these closure conditions. In particular, we obtain various Boolean algebras of context-sensitive languages accepted by deterministic vector addition systems. However, a better understanding of these languages seems to be a challenging problem. Moreover, it would be interesting to show that some other subfamilies of $\mathrm{TrSuffix}^{\mathrm{VAS}}$ also satisfy the closure conditions stated above.

References

1 R. Alur and P. Madhusudan. Visibly pushdown languages. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 202–211, 2004. doi:10.1145/1007352.1007390.

2 D. Caucal. On the Regular Structure of Prefix Rewriting. *Theor. Comput. Sci.*, 106(1):61–86, 1992. doi:10.1016/0304-3975(92)90278-N.

3 D. Caucal. On Infinite Transition Graphs Having a Decidable Monadic Theory. In *Automata, Languages and Programming, 23rd International Colloquium, ICALP96, Paderborn, Germany, 8-12 July 1996, Proceedings*, pages 194–205, 1996. doi:10.1007/3-540-61440-0_128.

4 D. Caucal. On the transition graphs of turing machines. *Theor. Comput. Sci.*, 296(2):195–223, 2003. doi:10.1016/S0304-3975(02)00655-2.

5 D. Caucal. Boolean algebras of unambiguous context-free languages. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2008, December 9-11, 2008, Bangalore, India*, pages 83–94, 2008. doi:10.4230/LIPIcs.FSTTCS.2008.1743.

6 D. Caucal and C. Rispal. Recognizability for Automata. In *Developments in Language Theory - 22nd International Conference, DLT 2018, Tokyo, Japan, September 10-14, 2018, Proceedings*, pages 206–218, 2018. doi:10.1007/978-3-319-98654-8_17.

7 D. Caucal and C. Rispal. Boolean Algebras by Length Recognizability. In Tiziana Margaria, Susanne Graf, and Kim G. Larsen, editors, *Models, Mindsets, Meta: The What, the How, and the Why Not? Essays Dedicated to Bernhard Steffen on the Occasion of His 60th Birthday*, pages 169–185. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-030-22348-9_11.

8 N. Chomsky. Three models for the description of languages. *IRE Transactions on Information Theory*, 2(3):113–124, September 1956. doi:10.1109/TIT.1956.1056813.

9 S. Crespi-reghizzi and D. Mandrioli. Petri nets and szilard languages. *Information and Control*, 33(2):177–192, 1977. doi:10.1016/S0019-9958(77)90558-7.

10 V. Diekert and G. Rozenberg. *The Book of Traces*. WORLD SCIENTIFIC, 1995. doi:10.1142/2563.

11 S. Eilenberg. *Automata, Languages, and Machines*. Academic Press, Inc., Orlando, FL, USA, 1974.

12 M. Hack. *Decidability questions for Petri nets*. PhD thesis, MIT, Dept. Electrical Engineering, Cambridge, Mass., USA, 1975.

13 B. Hodgson. On Direct Products of Automaton Decidable Theories. *Theor. Comput. Sci.*, 19:331–335, 1982. doi:10.1016/0304-3975(82)90042-1.

14 J. Leroux, V. Penelle, and G. Sutre. On the Context-Freeness Problem for Vector Addition Systems. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 43–52, 2013. doi:10.1109/LICS.2013.9.

15 C. Löding. Model-Checking Infinite Systems Generated by Ground Tree Rewriting. In *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8-12, 2002, Proceedings*, pages 280–294, 2002. doi:10.1007/3-540-45931-6_20.

16 A. Mansard. Unfolding of Finite Concurrent Automata. In *Proceedings 11th Interaction and Concurrency Experience, ICE 2018, Madrid, Spain, June 20-21, 2018.*, pages 68–84, 2018. doi:10.4204/EPTCS.279.8.

17 K. Mehlhorn. Pebbling Mountain Ranges and its Application of DCFL-Recognition. In *Automata, Languages and Programming, 7th Colloquium, Noordweijkerhout, The Netherlands, July 14-18, 1980, Proceedings*, pages 422–435, 1980. doi:10.1007/3-540-10003-2_89.

18 D. Nowotka and J. Srba. Height-Deterministic Pushdown Automata. In *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Český Krumlov, Czech Republic, August 26-31, 2007, Proceedings*, pages 125–134, 2007. doi:10.1007/978-3-540-74456-6_13.

19 J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

20 C. Rispal. The synchronized graphs trace the context-sensitive languages. *Electr. Notes Theor. Comput. Sci.*, 68(6):55–70, 2002. doi:10.1016/S1571-0661(04)80533-4.

21 S. R. Schwer. The context-freeness of the languages associated with vector addition systems is decidable. *Theoretical Computer Science*, 98(2):199–247, 1992. doi:10.1016/0304-3975(92)90002-W.

22 R. Valk and G. Vidal-Naquet. Petri nets and regular languages. *Journal of Computer and System Sciences*, 23(3):299–325, 1981. doi:10.1016/0022-0000(81)90067-2.

23 G. Winskel and M. Nielsen. Models for Concurrency. In S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science (Vol. 4)*, pages 1–148. Oxford University Press, Inc., New York, NY, USA, 1995. URL: http://dl.acm.org/citation.cfm?id=218623.218630.