Synchronization of regular automata

Didier Caucal

IGM–CNRS Université Paris-Est
caucal@univ-mlv.fr

Abstract. Functional graph grammars are finite devices which generate the class of regular automata. We recall the notion of synchronization by grammars, and for any given grammar we consider the class of languages recognized by automata generated by all its synchronized grammars. The synchronization is an automaton-related notion: all grammars generating the same automaton synchronize the same languages. When the synchronizing automaton is unambiguous, the class of its synchronized languages forms an effective boolean algebra lying between the classes of regular languages and unambiguous context-free languages. We additionally provide sufficient conditions for such classes to be closed under concatenation and its iteration.

1 Introduction

An automaton over some alphabet can simply be seen as a finite or countable set of labelled arcs together with two sets of initial and final vertices. Such an automaton recognizes the language of all words labelling an accepting path, i.e. a path leading from an initial to a final vertex. It is well-known that finite automata recognize the regular languages. By applying basic constructions to finite automata, we obtain the nice closure properties of regular languages, namely their closure under boolean operations, concatenation and its iteration. For instance the synchronization product and the determinization of finite automata respectively yield the closure of regular languages under intersection and under complement.
This idea can be extended to more general classes of automata. In this paper, we will be interested in the class of regular automata, which recognize context-free languages and are defined as the (generally infinite) automata generated by functional graph grammars [Ca 07]. Regular automata of finite degree are also precisely those automata which can be finitely decomposed by distance, as well as the regular restrictions of transition graphs of pushdown automata [MS 85], [Ca 07]. Even though the class of context-free languages does not enjoy the same closure properties as regular languages, one can define subclasses of context-free languages which do, using the notion of synchronization.
The notion of synchronization was first defined between grammars [CH 08]. A grammar S is synchronized by a grammar R if for any accepting path μ of (the graph generated by) S, there exists an accepting path λ of R with the same label u such that λ and μ are synchronized: for every prefix v of u, the prefixes

of λ and μ labelled by v lead to vertices of the same level (where the level of a vertex is the minimal number of rewriting steps necessary for the grammar to produce it). A language is synchronized by a grammar R if it is recognized by an automaton generated by a grammar synchronized by R. A fundamental result is that two grammars generating the same automaton yield the same class of synchronized languages [Ca 08]. This way, the notion of synchronization can be transferred to the level of automata: for a regular automaton G, the family $Sync(G)$ is the set of languages synchronized by any grammar generating G.
By extending the above-mentioned constructions from finite automata to grammars, one can establish several closure properties of these families of synchronized languages. The sum of two grammars and the synchronization product of a grammar with a finite automaton respectively entail the closure of $Sync(G)$ under union and under intersection with a regular language for any regular automaton G. The (level preserving) synchronization product of two grammars yields the closure under intersection of $Sync(G)$ when G is unambiguous *i.e.* when any two accepting paths of G have distinct labels. Normalizing of grammar into a grammar only containing arcs and then the (level preserving) determinization yields, for any unambiguous automaton G, the closure of $Sync(G)$ under complement relative to $L(G)$. This normalization also allows us to express $Sync(G)$ in the case of an infinite degree automaton G, by performing the e-closure of $Sync(H)$ for some finite degree automaton H using an extra label e.
A final useful normalization only allows the presence of initial and final vertices at level 0. It yields sufficient conditions for the closure of classes of synchronized languages under concatenation and its iteration.
In Section 2, we recall the definition of regular automata. In the next section, we summarize known results on the synchronization of regular automata [Ca 06], [NS 07], [CH 08], [Ca 08]. In the last section, we present a simpler construction for the closure under complement of $Sync(G)$ for unambiguous G [Ca 08] and present new results, especially sufficient conditions for the closure of $Sync(G)$ under concatenation and its iteration.

2 Regular automata

An automaton is a labelled oriented simple graph with input and output vertices. It recognizes the set of words labelling the paths from an input to an output. Finite automata are automata having a finite number of vertices, they recognize the class of regular languages. Regular automata are the automata generated by functional graph grammars, they recognize the class of context-free languages. A key result, originally due to Muller and Schupp, identifies the regular automata of finite degree with the automata finitely generated by distance.

An automaton over an alphabet (finite set of symbols) T of *terminals* is just a set of arcs labelled over T (a simple labelled oriented graph) with initial and final vertices. We use two symbols ι and o to mark respectively the initial and final vertices. More precisely an *automaton* G is defined by $G \subseteq T \times V \times V \cup \{\iota, o\} \times V$

where V is an arbitrary set such that the following set of *vertices*

$$V_G \;=\; \{\; s \in V \mid (\iota, s) \in G \;\vee\; (o, s) \in G \;\vee\; \exists\, a \in T\; \exists\, t \in V\; (a, s, t) \in G \;\vee\; (a, t, s) \in G \;\}$$

is finite or countable. Any triple $(a, s, t) \in G$ is an *arc* labelled by a from *source* s to *goal* t; it is identified with the labelled transition $s \xrightarrow[G]{a} t$ or directly $s \xrightarrow{a} t$ if G is understood. Any pair $(c, s) \in G$ is a *coloured vertex* s by $c \in \{\iota, o\}$ also written $c\,s$. A vertex is *initial* (resp. *final*) if it is coloured by ι (resp. o) *i.e.* $\iota\, s \in G$ (resp. $o\, s \in G$). An example of an automaton is given by

$$G = \{\; n \xrightarrow{a} n+1 \mid n \geq 0 \;\} \cup \{\; n \xrightarrow{b} x^n \mid n > 0 \;\} \cup \{\; n \xrightarrow{b} y^{2n} \mid n > 0 \;\}$$
$$\cup \{\; x^{n+1} \xrightarrow{b} x^n \mid n > 0 \;\} \cup \{\; y^{n+1} \xrightarrow{b} y^n \mid n > 0 \;\}$$
$$\cup \{\iota\, 0\,,\, o\, y\} \qquad \cup \{\; o\, x^n \mid n > 0 \;\} \qquad \cup \{\; \iota\, y^{2n+1} \mid n \geq 0 \;\}$$

and is represented (up to isomorphism) below.

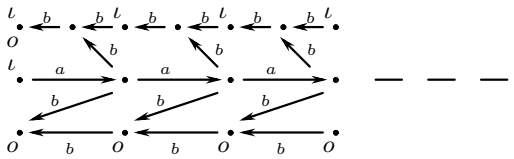


Figure 2.1 *An automaton.*

An automaton G is thus a simple vertex- and arc-labelled graph. G has *finite degree* if for any vertex s, the set $\{\; t \mid \exists\; a\; (s \xrightarrow{a} t \;\vee\; t \xrightarrow{a} s) \;\}$ of its adjacent vertices is finite. Recall that $(s_0, a_1, s_1, \ldots, a_n, s_n)$ for $n \geq 0$ and $s_0 \xrightarrow[G]{a_1} s_1 \;\ldots\; s_{n-1} \xrightarrow[G]{a_n} s_n$ is a *path* from s_0 to s_n labelled by $u = a_1 \ldots a_n$; we write $s_0 \xRightarrow[G]{u} s_n$ or directly $s_0 \xRightarrow{u} s_n$ if G is understood. An *accepting path* is a path from an initial vertex to a final vertex. An automaton is *unambiguous* if two accepting paths have distinct labels. The automaton of Figure 2.1 is unambiguous. The *language recognized* by an automaton G is the set $L(G)$ of all labels of its accepting paths: $L(G) \;=\; \{\; u \in T^* \mid \exists\; s, t\; (s \xRightarrow[G]{u} t \;\wedge\; \iota\, s\,,\; o\, t \in G) \;\}$. Note that $\varepsilon \in L(G)$ if there exists a vertex s which is initial and final: $\iota\, s\,,\; o\, s \in G$. The automaton G of Figure 2.1 recognizes the language

$$L(G) \;=\; \{\; a^m b^n \mid 0 < n \leq m \;\} \;\cup\; \{\; a^n b^{2n} \mid n > 0 \;\} \;\cup\; \{\; b^{2n} \mid n \geq 0 \;\}.$$

The languages recognized by finite automata are the *regular languages* over T. We generalize finite automata to regular automata using functional graph grammars. To define a graph grammar, we need to extend an arc (resp. a graph) to a hyperarc (resp. a hypergraph). Although such an extension is natural, this may explain why functional graph grammars are not very widespread at the moment. But we will see in the last section that for our purpose, we can restrict to grammars using only arcs.

Let F be a set of symbols ranked by a mapping $\varrho : F \longrightarrow \mathbb{N}$ associating to each $f \in F$ its *arity* $\varrho(f) \geq 0$ such that $F_n = \{\; f \in F \mid \varrho(f) = n \;\}$ is countable for every $n \geq 0$ with $T \subset F_2$ and $\iota, o \in F_1$.

A *hypergraph* G is a subset of $\bigcup_{n \geq 0} F_n \times V^n$ where V is an arbitrary set. Any

tuple $(f, s_1, \ldots, s_{\varrho(f)}) \in G$, also written $fs_1 \ldots s_{\varrho(f)}$, is a *hyperarc* of *label* f and of successive *vertices* $s_1, \ldots, s_{\varrho(f)}$. We add the condition that the set of vertices V_G is finite or countable, and the set of labels F_G is finite. An *arc* is a hyperarc fst labelled by $f \in F_2$ and is also denoted by $s \xrightarrow{f} t$. For $n \geq 2$, a hyperarc $fs_1 \ldots s_n$ is depicted as an arrow labelled f and successively linking $s_1, \ldots, s_n$. For $n = 1$ and $n = 0$, it is respectively depicted as a label f (called a *colour*) on vertex s_1 and as an isolated label f called a *constant*. This is illustrated in the next figures. For instance the following hypergraph:

$G = \{4 \xrightarrow{b} 1\,,\, 5 \xrightarrow{b} 1\,,\, 2 \xrightarrow{a} 5\,,\, 5 \xrightarrow{b} 3\,,\, 6 \xrightarrow{b} 3\,,\, \iota\,4\,,\, o\,6\,,\, A456\}$
with $a, b \in F_2$ and $A \in F_3$, is represented below.

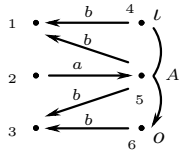


Figure 2.2 *A finite hypergraph.*

A (coloured) *graph* G is a hypergraph whose labels are only of arity 1 or 2: $F_G \subset F_1 \cup F_2$. An automaton G over the alphabet T is a graph with a set of labels $F_G \subseteq T \cup \{\iota, o\}$. We can now introduce functional graph grammars to generate regular automata.

A graph *grammar* R is a finite set of rules of the form $fx_1 \ldots x_{\varrho(f)} \longrightarrow H$ where $fx_1 \ldots x_{\varrho(f)}$ is a hyperarc of label f called *non-terminal* joining pairwise distinct vertices $x_1 \neq \ldots \neq x_{\varrho(f)}$ and H is a finite hypergraph.

We denote by N_R the set of non-terminals of R *i.e.* the labels of the left hand sides, by $T_R = \{\ f \in F - N_R \mid \exists\ H \in Im(R),\ \ f \in F_H\ \}$ the *terminals* of R *i.e.* the labels of R which are not non-terminals, and by $F_R = N_R \cup T_R$ the *labels* of R.

We use grammars to generate automata hence in the following, we may assume that $T_R \subseteq T \cup \{\iota, o\}$. We restrict any hypergraph H to the automaton $[H]$ of its terminal arcs and coloured vertices: $[H] = H \cap (T \times V_H \times V_H \cup \{\iota, o\} \times V_H)$.

Similarly to context-free grammars (on words), a graph grammar has an axiom: an initial finite hypergraph. To indicate this axiom, we assume that any grammar R has a constant non-terminal $Z \in N_R \cap F_0$ which is not a label of any right hand side; the *axiom* of R is the right hand side H of the rule of Z: $Z \longrightarrow H \wedge Z \notin F_K$ for any $K \in Im(R)$.

Starting from the axiom, we want R to generate a unique automaton up to isomorphism. So we finally assume that any grammar R is *functional* meaning that there is only one rule per non-terminal: if $(X, H)\,,\, (Y, K) \in R$ with $X(1) = Y(1)$ then $(X, H) = (Y, K)$.

For any rule $fx_1 \ldots x_{\varrho(f)} \longrightarrow H$, we say that $x_1, \ldots, x_{\varrho(f)}$ are the *inputs* of f, and $V_{H-[H]}$ is the set of *outputs* of f.

To work with these grammars, it is simpler to assume that any grammar R is *terminal-outside* [Ca 07]: any terminal arc or colour in a right hand side links to

at least one non input vertex: $H \cap (T \times V_X \times V_X \cup \{\iota, o\} \times V_X) = \emptyset$ for any rule $(X, H) \in R$. In particular an input is not initial and not final.
We will use upper-case letters $A, B, C, \ldots$ for non-terminals and lower-case letters $a, b, c \ldots$ for terminals. Here is an example of a (functional graph) grammar R:

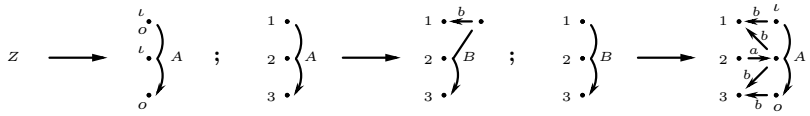


Figure 2.3 *A (functional graph) grammar.*

For the previous grammar R, we have $N_R = \{Z, A, B\}$ with Z the axiom and $\varrho(A) = \varrho(B) = 3$, $T_R = \{a, b, \iota, o\}$ and $1, 2, 3$ are the inputs of A and B.
Given a grammar R, the *rewriting* relation $\underset{R}{\longrightarrow}$ is the binary relation between hypergraphs defined as follows: M rewrites into N, written $M \underset{R}{\longrightarrow} N$, if we can choose a non-terminal hyperarc $X = As_1 \ldots s_p$ in M and a rule $Ax_1 \ldots x_p \longrightarrow H$ in R such that N can be obtained by replacing X by H in M: $N = (M - X) \cup h(H)$ for some function h mapping each x_i to s_i, and the other vertices of H injectively to vertices outside of M; this rewriting is denoted by $M \underset{R,\, X}{\longrightarrow} N$. The rewriting $\underset{R,\, X}{\longrightarrow}$ of a hyperarc X is extended in an obvious way to the rewriting $\underset{R,\, E}{\longrightarrow}$ of any set E of non-terminal hyperarcs. The *complete parallel rewriting* $\underset{R}{\Longrightarrow}$ is a simultaneous rewriting according to the set of all non-terminal hyperarcs: $M \underset{R}{\Longrightarrow} N$ if $M \underset{R,\, E}{\longrightarrow} N$ where E is the set of all non-terminal hyperarcs of M. We depict below the first three steps of the parallel derivation of the previous grammar from its constant non-terminal Z:

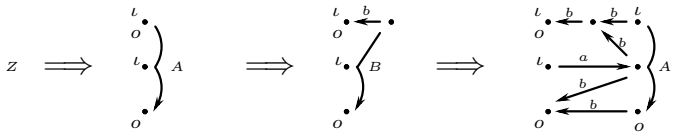
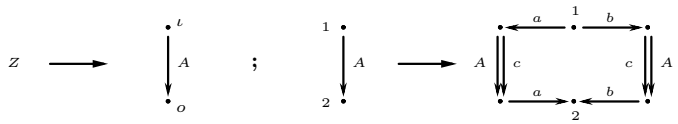


Figure 2.4 *Parallel derivation for the grammar of Figure 2.3.*

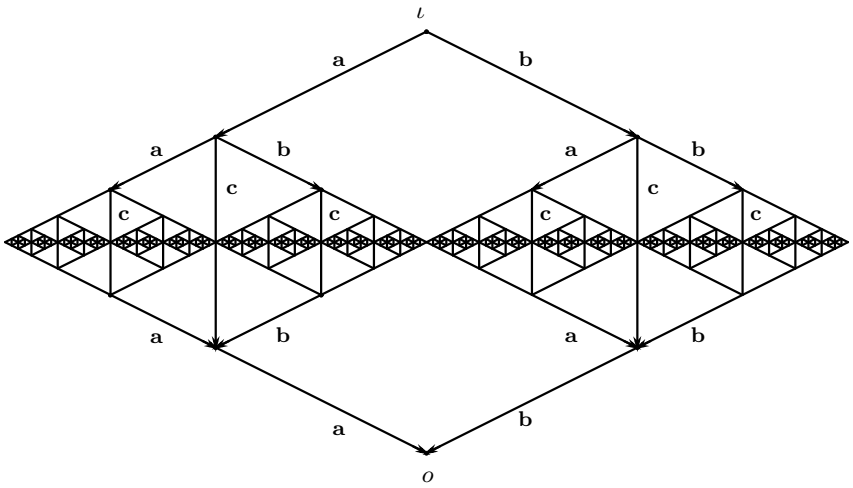
An automaton G is *generated* by R (from its axiom) if G belongs to the following set R^ω of isomorphic automata:

$$R^\omega \;=\; \{\, \bigcup_{n \geq 0} [H_n] \mid Z \underset{R}{\longrightarrow} H_0 \underset{R}{\Longrightarrow} \ldots H_n \underset{R}{\Longrightarrow} H_{n+1} \ldots \,\}.$$

Note that in all generality, we need to consider hypergraphs with multiplicities. However using an appropriate normal form, this technicality can be safely omitted [Ca 07].
For instance the automaton of Figure 2.1 is generated by the grammar of Figure 2.3. A *regular automaton* is an automaton generated by a (functional graph) grammar. Note that a regular automaton has a finite number of non-isomorphic connected components, and has a finite number of distinct vertex degrees.
Another example is given by the following grammar:



which generates the following automaton:



recognizing the language $\{\ uc\widetilde{u} \mid u \in \{a,b\}^+\ \}$ where $\widetilde{u}$ is the mirror of u.
The *language recognized* by a grammar R is the language $L(R)$ recognized by its generated automaton: $L(R) = L(G)$ for (any) $G \in R^\omega$. This language is well-defined since all automata generated by a given grammar are isomorphic.
A grammar R is an *unambiguous grammar* if the automaton it generates is unambiguous.
There is a canonical way to generate the regular automata of finite degree which allows to characterize these automata without the explicit use of grammars. This is the finite decomposition by distance.
The *inverse* G^{-1} of an automaton G is the automaton obtained from G by reversing its arcs and by exchanging initial and final vertices:

$$G^{-1} \ = \ \{\ t \xrightarrow{a} s \mid s \xrightarrow[G]{a} t\ \} \cup \{\ \iota\, s \mid o\, s \in G\ \} \cup \{\ o\, s \mid \iota\, s \in G\ \}.$$

So G^{-1} recognizes the mirror of the words recognized by G. The *restriction* $G_{|I}$ of G to a subset I of vertices is the subgraph of G induced by I:

$$G_{|I} \ = \ G \cap (T{\times}I{\times}I \cup \{\iota, o\}{\times}I).$$

The *distance* $d_I(s)$ of a vertex s to I is the minimal length of the undirected paths between s and I: $d_I(s) = min\{\ |u| \mid \exists\ r \in I,\ r \underset{G \cup G^{-1}}{\overset{u}{\Longrightarrow}} s\ \}$ with $min(\emptyset) = +\infty$.
We take a new colour $\# \in F_1 - \{\iota, o\}$ and define for any integer $n \geq 0$,

$$Dec^{\#}_n(G,I) \ = \ G_{|\{\ s \ \mid\ d_I(s) \geq n\ \}} \ \cup \ \{\ \#\, s \mid d_I(s) = n\ \}\,.$$

In particular $Dec^{\#}_0(G,I) = G \cup \{\ \#\, s \mid s \in I\ \}$. We say that an automaton G is *finitely decomposable by distance* if for each connected component C of G there exists a finite non empty set I of vertices such that $\bigcup_{n\geq 0} Dec^{\#}_n(C,I)$ has a finite number of non-isomorphic connected components. Such a definition allows the

characterization of the class of all automata of finite degree which are regular.

Theorem 2.5 *An automaton of finite degree is regular if and only if it is finitely decomposable by distance and it has only a finite number of non isomorphic connected components.*

The proof is given in [Ca 07] and is a slight extension of [MS 85] (but without using pushdown automata). Regular automata of finite degree are also the transition graphs of pushdown automata restricted to regular sets of configurations and with regular sets of initial and final configurations. In particular, regular automata of finite degree recognize the same languages as pushdown automata.

Proposition 2.6 *The (resp. unambiguous) regular automata recognize exactly the (resp. unambiguous) context-free languages.*

This proposition remains true if we restrict to automata of finite degree. We now use grammars to extend the family of regular languages to boolean algebras of unambiguous context-free languages.

3 Synchronization of regular automata

We introduce the idea of synchronization between grammars. The class of languages synchronized by a grammar R are the languages recognized by grammars synchronized by R. We show that these families of languages are closed under union by applying the sum of grammars, are closed under intersection with a regular language by defining the synchronization product of a grammar with a finite automaton, and are closed under intersection (in the case of grammars generating unambiguous automata) by performing the synchronization product of grammars. Finally we show that all grammars generating the same automaton synchronize the same languages.

To each vertex s of an automaton $G \in R^\omega$ generated by a grammar R, we associate a non negative integer $\ell(s)$ which is the minimal number of rewritings applied from the axiom necessary to reach s. More precisely for $G = \bigcup_{n\geq 0}[H_n]$ with $Z \underset{R}{\longrightarrow} H_0 \underset{R}{\Longrightarrow} \ldots H_n \underset{R}{\Longrightarrow} H_{n+1} \ldots$, the *level* $\ell(s)$ of $s \in V_G$, also written $\ell_G^R(s)$ to specify G and R, is $\ell(s) = min\{\ n \mid s \in V_{H_n}\ \}$.
We depict below the levels of some vertices of the regular automaton of Figure 2.1 generated by the grammar of Figure 2.3. This automaton is represented by vertices of increasing level: vertices at a same level are aligned vertically.

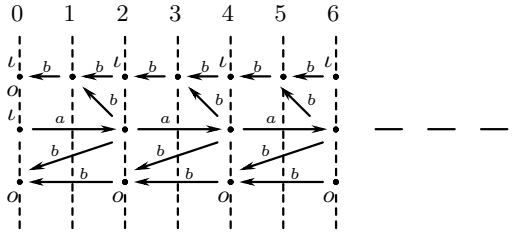


Figure 3.1 *Vertex levels with the grammar of Figure 2.3.*

We say that a grammar S is *synchronized* by a grammar R written $S \lhd R$, or equivalently that R *synchronizes* S written $R \rhd S$, if for any accepting path μ label by u of the automaton generated by S, there is an accepting path λ label by u of the automaton generated by R such that for every prefix v of u, the prefixes of λ and μ labelled by v lead to vertices of the same level: for (any) $G \in R^\omega$ and (any) $H \in S^\omega$ and for any $t_0 \xrightarrow[H]{a_1} t_1 \ldots \xrightarrow[H]{a_n} t_n$ with $\iota\, t_0\,,\, o\, t_n \in H$, there exists

$s_0 \xrightarrow[G]{a_1} s_1 \ldots \xrightarrow[G]{a_n} s_n$ with $\iota\, s_0\,,\, o\, s_n \in G$ and $\ell^R_G(s_i) = \ell^S_H(t_i) \;\; \forall\; i \in [0, n]$.

For instance the grammar of Figure 2.3 synchronizes the following grammar:

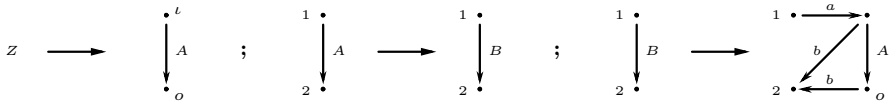


Figure 3.2 *A grammar synchronized by the grammar of Figure 2.3.*

In particular for $S \lhd R$, we have $L(S) \subseteq L(R)$. Note that the *empty grammar* $\{(Z, \emptyset)\}$ is synchronized by any grammar. The synchronization relation $\rhd$ is a reflexive and transitive relation. We denote $\bowtie$ the *bi-synchronization* relation: $R \bowtie S$ if $R \rhd S$ and $S \rhd R$. Note that bi-synchronized grammars $R \bowtie S$ may generate distinct automata: $R^\omega \neq S^\omega$. For any grammar R, the image of R by $\rhd$ is the family $\rhd(R) = \{\; S \mid R \rhd S \;\}$ of grammars synchronized by R and $Sync(R) = \{\; L(S) \mid S \lhd R \;\}$ is the family of languages synchronized by R. Note that $Sync(R)$ is a family of languages included in $L(R)$ and containing the empty language and $L(R)$. Note also that $Sync(R) = Sync(S)$ for $R \bowtie S$.

Standard operations on finite automata are extended to grammars in order to obtain closure properties of $Sync(R)$. For instance the *synchronization product* of finite automata is extended to arbitrary automata G and H by

$$G{\times}H = \{\; (s,p) \xrightarrow{a} (t,q) \mid s \xrightarrow[G]{a} t \;\wedge\; p \xrightarrow[H]{a} q \;\}$$
$$\cup\; \{\; \iota(s,p) \mid \iota\, s \in G \;\wedge\; \iota\, p \in H \;\} \;\cup\; \{\; o(s,p) \mid o\, s \in G \;\wedge\; o\, p \in H \;\}$$

which recognizes $L(G{\times}H) = L(G) \cap L(H)$.

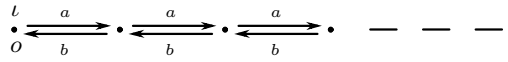
The synchronization product of a regular automaton G, generated by a grammar R, with a finite automaton K remains regular: it is generated by a grammar $R{\times}K$ that we define [CH 08]. Let $\{q_1, \ldots, q_n\}$ be the vertex set of K. To each $A \in N_R$, we associate a new symbol (A, n) of arity $\varrho(A){\times}n$ except that $(Z, 0) =$

Z, and to each hyperarc $Ar_1 \ldots r_m$ with $m = \varrho(A)$, we associate the hyperarc $(Ar_1 \ldots r_m)_K = (A,n)(r_1,q_1)\ldots(r_1,q_n)\ldots(r_m,q_1)\ldots(r_m,q_n)$.
The grammar $R{\times}K$ associates to each rule $(X,H) \in R$ the following rule:

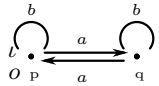
$$X_K \longrightarrow [H]{\times}K \cup \{\ (BY)_K \mid BY \in H \ \wedge\ B \in N_R\ \}\,.$$

Example 3.3 Let us consider the following grammar R:

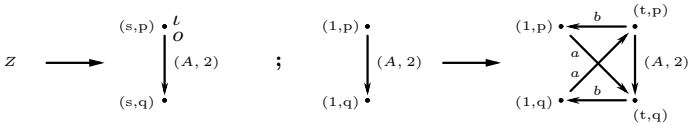
generating the following (regular) automaton G:



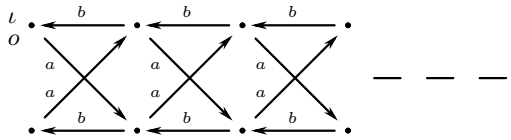
and recognizing the restricted Dyck language $D_1'^*$ over the pair (a,b) [Be 79]: $L(R) = L(G) = D_1'^*$. We consider the following finite automaton K:



recognizing the set of words over $\{a,b\}$ having an even number of a.
So $R{\times}K$ is the following grammar:



generating the automaton $G{\times}K$:



which recognizes $D_1'^*$ restricted to the words with an even number of a (or b). □

The synchronization product of a grammar R with a finite automaton K is synchronized by R *i.e.* $R{\times}K \lhd R$ and recognizes $L(R{\times}K) = L(R) \cap L(K)$.

Proposition 3.4 *For any grammar R, the family $Sync(R)$ is closed under intersection with a regular language.*

Propositions 2.6 and 3.4 imply the well-known closure property of the family of context-free languages under intersection with a regular language. As $R{\times}K$ is unambiguous for R unambiguous and K deterministic, it also follows Theorem 6.4.1 of [Ha 78]: the family of unambiguous context-free languages is closed

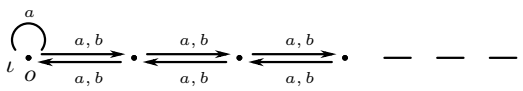
under intersection with a regular language.
Another basic operation on finite automata is the disjoint union. This operation is extended to any grammars R_1 and R_2. For any $i \in \{1,2\}$, we denote $R'_i = R_i \times \big(\{\, i \xrightarrow{a} i \mid a \in T \,\} \cup \{\iota\, i\,,\, o\, i\}\big)$ in order to distinguish the vertices of R_1 and R_2. For $(Z, H_1) \in R'_1$ and $(Z, H_2) \in R'_2$, the *sum* of R_1 and R_2 is the grammar

$$R_1 + R_2 \;=\; \{(Z\,,\, H_1 \cup H_2)\} \;\cup\; (R'_1 - \{(Z, H_1)\}) \;\cup\; (R'_2 - \{(Z, H_2)\})\,.$$

So $(R_1 + R_2)^\omega = \{\, G_1 \cup G_2 \mid G_1 \in R_1^\omega \;\wedge\; G_2 \in R_2^\omega \;\wedge\; V_{G_1} \cap V_{G_2} = \emptyset \,\}$ hence $L(R_1 + R_2) = L(R_1) \cup L(R_2)$. In particular if $S_1 \lhd R_1$ and $S_2 \lhd R_2$ then $S_1 + S_2 \lhd R_1 + R_2$.

Proposition 3.5 *For any grammar R, $Sync(R)$ is closed under union.*

The synchronization product of regular automata can be non regular. Furthermore for the regular automaton G:

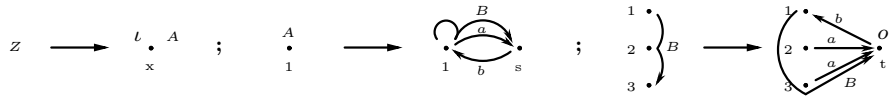


the languages $\{\, a^m b^m a^n \mid m, n \geq 0 \,\}$ and $\{\, a^m b^n a^n \mid m, n \geq 0 \,\}$ are in $Sync(G)$ but their intersection $\{\, a^n b^n a^n \mid n \geq 0 \,\}$ is not a context-free language.
The synchronization product of a grammar with a finite automaton is extended for two grammars R and S for generating the *level synchronization product* $G{\times_{R,S}}H$ of their generated automata $G \in R^\omega$ and $H \in S^\omega$ which is the restriction of $G{\times}H$ to pairs of vertices with same level: $G{\times_{R,S}}H = (G{\times}H)_{|P}$ for $P = \{\, (s,p) \in V_G{\times}V_H \mid \ell^R_G(s) = \ell^S_H(p) \,\}$. This product can be generated by a grammar $R{\times}S$ that we define. Let $(A, B) \in N_R{\times}N_S$ be any pair of non-terminals and $E \subseteq [1, \varrho(A)]{\times}[1, \varrho(B)]$ be a binary relation over inputs such that for all $i, j \in [1, \varrho(A)]$, if $E(i) \cap E(j) \neq \emptyset$ then $E(i) = E(j)$, where $E(i) = \{j \mid (i,j) \in E\}$ denotes the *image* of $i \in [1, \varrho(A)]$ by E. Intuitively for a pair $(A, B) \in N_R{\times}N_S$ of non-terminals, a relation $E \subseteq [1, \varrho(A)]{\times}[1, \varrho(B)]$ is used to memorize which entries of A and B are being synchronized.
To any such A, B and E, we associate a new symbol $[A, B, E]$ of arity $|E|$ (where $[Z, Z, \emptyset]$ is assimilated to Z). To each non-terminal hyperarc $Ar_1 \ldots r_m$ of R ($A \in N_R$ and $m = \varrho(A)$) and each non-terminal hyperarc $Bs_1 \ldots s_n$ of S ($B \in N_S$ and $n = \varrho(B)$), we associate the hyperarc
$[Ar_1 \ldots r_m, Bs_1 \ldots s_n, E] = [A, B, E](r_1, s_1)_{_E} \ldots (r_1, s_n)_{_E} \ldots (r_m, s_1)_{_E} \ldots (r_m, s_n)_{_E}$
with $(r_i, s_j)_{_E} = (r_i, s_j)$ if $(i,j) \in E$, and ε otherwise. The grammar $R{\times}S$ is then defined by associating to each $(AX, P) \in R$, each $(BY, Q) \in S$, and each $E \subseteq [\varrho(A)]{\times}[\varrho(B)]$, the rule of left hand side $[AX, BY, E]$ and of right hand side

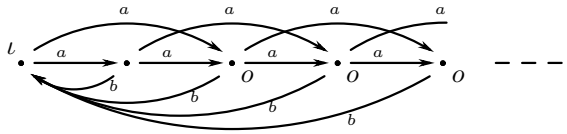
$$\big([P]{\times}[Q]\big)_{|\overline{E}} \;\cup\; \{[CU, DV, E'] \mid CU \in P \wedge C \in N_R \wedge DV \in Q \wedge D \in N_S\}$$

with $\overline{E} = \{\, (X(i), Y(j)) \mid (i,j) \in E \,\} \cup \big(V_P - V_X\big){\times}\big(V_Q - V_Y\big)$ and
$E' = \{\, (i,j) \in [\varrho(C)]{\times}[\varrho(D)] \mid (U(i), V(j)) \in \overline{E} \,\}$.

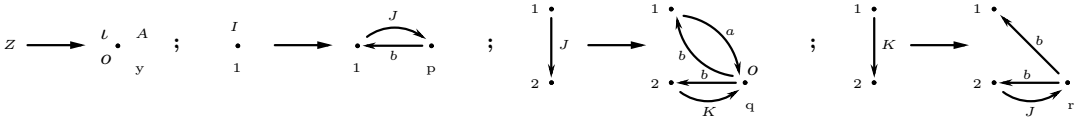
Example 3.6 Let us illustrate the level synchronization product of two grammars.
We take a first grammar R:



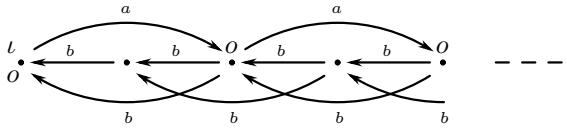
generating a graph G:



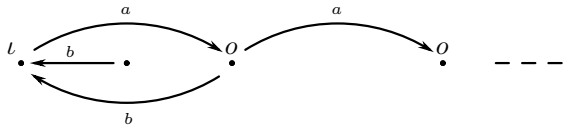
A second grammar S is the following:



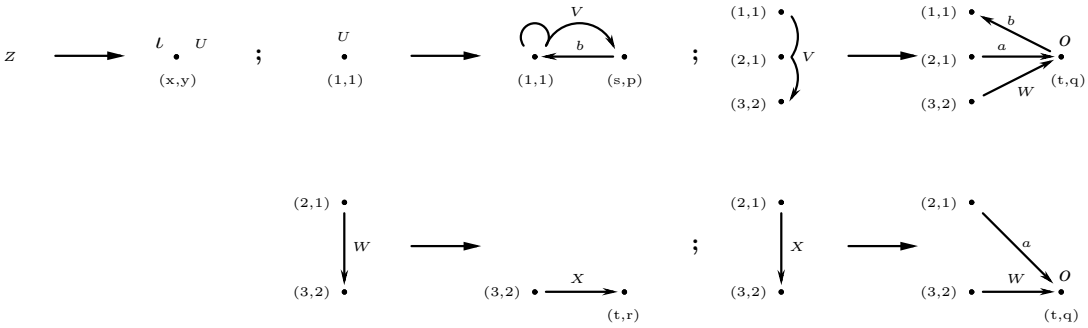
generating a graph H:



The level synchronization product $G\times_{R,S}H$ of the previous two graphs is the graph:



This graph is generated by the following grammar $R\times S$ restricted to the rules accessible from Z:



with $U = [A, I, \{(1,1)\}]$
$V = [B, J, \{(1,1),(2,1),(3,2)\}]$
$W = [B, K, \{(2,1),(3,2)\}]$
$X = [B, J, \{(2,1),(3,2)\}]$.

□

Note that $R{\times}S$ is synchronized by R and S, and is bi-synchrnonized with S for $S \lhd R$. Furthermore $R{\times}S$ generates $G{\times}_{R,S}H$ for $G \in R^\omega$ and $H \in S^\omega$ hence recognizes a subset of $L(R) \cap L(S)$. However for grammars S and S' synchronized by an unambiguous grammar R, we have $L(S{\times}S') = L(S) \cap L(S')$.

Proposition 3.7 *For any unambiguous grammar R, the family $Sync(R)$ is closed under intersection.*

By extending basic operations on finite automata to grammars, it appears that graph grammars are to context-free languages what finite automata are to regular languages. We will continue these extensions in the next section. Let us present a fundamental result concerning grammar synchronization, which states that $Sync(R)$ is independent of the way the automaton R^ω is generated.

Theorem 3.8 *For any grammars R and S such that $R^\omega = S^\omega$, we have*
$$Sync(R) = Sync(S).$$

Proof sketch.
By symmetry of R and S, it is sufficient to show that $Sync(R) \subseteq Sync(S)$.
Let $R' \lhd R$. We want to show that $L(R') \in Sync(S)$.
We have to show the existence of $S' \lhd S$ such that $L(S') = L(R')$.
Note that it is possible that there is no grammar S' synchronized by S and generating the same automaton as R' (*i.e.* $S' \lhd S$ and $S'^\omega = R'^\omega$).
Let $G \in R^\omega = S^\omega$. Any vertex s of G has a level $\ell^R_G(s)$ according to R and a level $\ell^S_G(s)$ according to S.
Let $H \in R'^\omega$ and let $K = (G{\times}_\ell H)_{|P}$ be the automaton obtained by level synchronization product of G with H and restricted to the set P of vertices accessible from ι and co-accessible from o.
The restriction by accessibility from ι and co-accessibility from o can de done by a bi-synchronized grammar [Ca 08]. By definition of $R{\times}R'$, the automaton K can be generated by a grammar R'' bi-synchronized to R' with
$$\ell^{R''}_K(s,p) = \ell^R_G(s) = \ell^{R'}_H(p) \quad \text{for every} \quad (s,p) \in V_K\,.$$
In particular $L(K) = L(R')$.
Let us show that K is generated by a grammar synchronized by S.
We give the proof for R^ω of finite degree. In that case and for $\| \varrho \| = \sum_{A \in N_R} \varrho(A)$,
$$|\ell^R_G(s) - \ell^R_G(t)| \;\leq\; \| \varrho \|.d_G(s,t) \quad \text{for every} \quad s,t \in V_G\,.$$
Furthermore K is also of finite degree.
We show that K is finitely decomposable not by distance but according to $\ell^S_K(s)$ for the vertices (s,p) of K.

Let $n \geq 0$ and C be a connected component of $K_{|\{\,(s,p)\in V_K \mid \ell^S_G(s)\geq n\,\}}$.
So C is fully determined by

its *frontier* : $Fr_K(C) = V_C \cap V_{K-C}$

its *interface* : $Int_K(C) = \{\ s \xrightarrow[C]{a} t \mid \{s,t\} \cap Fr_K(C) \neq \emptyset\ \}$.

Let $(s_0,p_0) \in Fr_K(C)$ and D be the connected component of $G_{\{\,s \mid \ell^S_G(s)\geq n\,\}}$ containing s_0. It remains to find a bound b independent of n such that

$$|\ell^{R''}_K(s,p) - \ell^{R''}_K(t,q)| \leq b \quad \text{for every} \quad (s,p)\,,\,(t,q) \in Fr_K(C).$$

For any $(s,p)\,,\,(t,q) \in Fr_K(C)$, we have $s,t \in Fr_G(D)$ hence $d_D(s,t)$ is bounded by the integer

$$c \;=\; max\{\ d_{S^\omega(A)}(i,j) < +\infty \mid A \in N_S \ \wedge\ i,j \in [1,\varrho(A)]\ \}$$

whose $S^\omega(A) \;=\; \{\ \bigcup_{n\geq 0}[H_n] \mid A1\ldots\varrho(A) = H_0 \underset{S}{\Longrightarrow} \ldots\ H_n \underset{S}{\Longrightarrow} H_{n+1}\ \ldots\ \}$

thus it follows that

$$|\ell^{R''}_K(s,p) - \ell^{R''}_K(t,q)| = |\ell^R_G(s) - \ell^R_G(t)| \leq \|\,\varrho\,\| d_G(s,t) \leq \|\,\varrho\,\| d_D(s,t) \leq \|\,\varrho\,\| c\,.$$

For G of infinite degree and by Proposition 4.9, we can express $Sync(G)$ as an ε-closure of $Sync(H)$ for some regular automaton H of finite degree using ε-transitions.
□

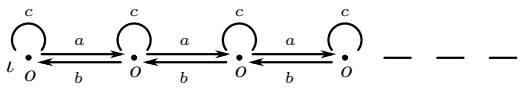
Theorem 3.8 allows to transfer the concept of grammar synchronization to the level of regular automata: for any regular automaton G, we can define

$$Sync(G) = Sync(R) \ \text{ for (any) } R \text{ such that } G \in R^\omega.$$

The synchronization relation is also extended between regular automata. A regular automaton H is synchronized by a regular automaton G, and we write $H \lhd G$ or $G \rhd H$, if there exists a grammar S generating H which is synchronized by a grammar R generating G: $S \lhd R$, $H \in S^\omega$ and $G \in R^\omega$.
Let us illustrate these ideas by presenting some examples of well-known subfamilies of context-free languages obtained by synchronization.

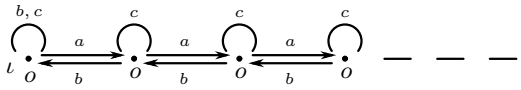
Example 3.9 For any finite automaton G, $Sync(G)$ is the family of regular languages included in $L(G)$.

Example 3.10 For the following regular automaton G:



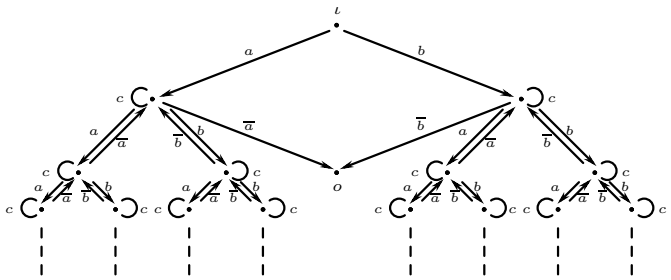
$Sync(G)$ is the family of *input-driven languages* [Me 80] with a pushing, b popping and c internal. As the initial vertex is not source of an arc labelled by b, $Sync(G)$ does not contain all the regular languages.

Example 3.11 We complete the previous automaton by adding an b-loop on the initial vertex to obtain the following automaton G:



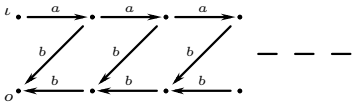
The set $Sync(G)$ is the family of *visibly pushdown languages* [AM 04] with a pushing, b popping and c internal.

Example 3.12 For the following regular automaton G:



the set $Sync(G)$ is the family of *balanced languages* [BB 02] with a, b pushing with their corresponding popping letters $\overline{a}, \overline{b}$, and c is internal.

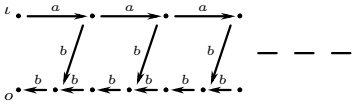
Example 3.13 For the following regular automaton G_1:



the family $Sync(G_1)$ is the set of languages generated from I by the following linear context-free grammars:

$$I = P \ + \ a^m A b^m \text{ with } m \geq 0 \ \text{ and } \ P \subseteq \{ab, \ldots, a^m b^m\}$$
$$A = Q \ + \ a^n A b^n \ \text{ with } \ n > 0 \ \text{ and } \ Q \subseteq \{ab, \ldots, a^n b^n\}\,.$$

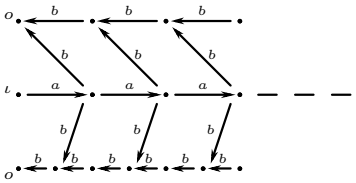
Example 3.14 For the following regular automaton G_2:



the family $Sync(G_2)$ is the set of languages generated from I by the following linear context-free grammars:

$$I = P \ + \ a^m A b^{2m} \ \text{ with } \ m \geq 0 \ \text{ and } \ P \subseteq \{abb, \ldots, a^m b^{2m}\}$$
$$A = Q \ + \ a^n A b^{2n} \ \text{ with } \ n > 0 \ \text{ and } \ Q \subseteq \{abb, \ldots, a^n b^{2n}\}\,.$$

Example 3.15 For the following unambiguous regular automaton G:

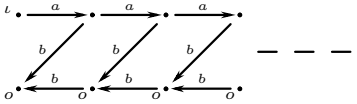


we have

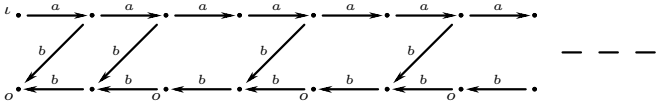
$$Sync(G) \;=\; \{\; L_1 \cup L_2 \mid L_1 \in Sync(G_1) \;\wedge\; L_2 \in Sync(G_2) \;\}$$

for the regular automata G_1 and G_2 of the previous Examples 3.13 and 3.14.

Example 3.16 The regular automaton G:



synchronizes the regular automaton:



which recognizes the language generated by the following context-free grammar:

$$\begin{aligned} I &= ab + aA \;\; + aBb \\ A &= \quad\; aaA + aaBb \\ B &= ab \quad\;\; + aaBbb \end{aligned}$$

More generally $Sync(G)$ is the family of languages generated by the linear context-free grammars:

$$\begin{aligned} I &= L_0 + a^{n_0}A + a^{n_0}BM_0 \\ A &= L_1 + a^{n_1}A + a^{n_1}BM_1 \\ B &= L \quad\;\; + a^{n_1}Bb^{n_1} \end{aligned}$$

defined for $n_0 \geq 0$ and $n_1 > 0$, and for $I_0, J_0, K_0 \subseteq [0, n_0[$ and $I_1, J_1, K_1 \subseteq [0, n_1[$ such that for every $k \in \{0, 1\}$,

$$\begin{aligned} L_k &= \{\; a^{i+1}b^{i+1-j} \mid i \in I_k \;\wedge\; j \in J_k \;\wedge\; j \leq i \;\wedge\; [j, i[\cap K_k = \emptyset \;\} \\ M_k &= \{\; b^{n_k - j} \mid j \in J_k \;\wedge\; [j, n_k[\cap K_k = \emptyset \;\} \\ L &= \{\; a^{i+1}b^{i+1} \mid i \in I_1 \;\wedge\; [0, i[\cap K_1 = \emptyset \;\}\,. \end{aligned}$$

Intuitively, the integer n_0 (resp. n_1) is the length of the 'base' (resp. of the 'period') and for any $k \in \{0, 1\}$, I_k, J_k, K_k are the subsets of $[0, n_k[$ such that I_k is the set of the goals of the b-diagonals, J_k is the set of the positions of the outputs, and K_k is the set of the non allowed positions: there are no goal of a b-horizontal.
□

For each regular automaton G among the previous examples, $Sync(G)$ is a boolean algebra according to $L(G)$ and, for the Examples 3.9, 3.10 and 3.11, is also closed under concatenation and its iteration. We now consider new closure properties of synchronized languages for regular automata.

4 Closure properties

We have seen that the family $Sync(G)$ of languages synchronized by a regular automaton G is closed under union and under intersection with a regular language, and under intersection when G is unambiguous. In this section, we consider the closure of $Sync(G)$ under complement relative to $L(G)$ and under concatenation and its transitive closure. To obtain these closure properties, we first apply grammar normalizations preserving the synchronized languages. These normalizations also allow us to add ε-arcs to any regular automaton to get a regular automaton of finite degree with the same synchronized languages.

First we put any grammar in an *equivalent* normal form with the same set of synchronized languages. As in the case of finite automata, we transform any automaton G into the *pointed automaton* $G^{\top}_{\bot}$ which is language equivalent $L(G^{\top}_{\bot}) = L(G)$, with a unique initial vertex $\top \notin V_G$ which is goal of no arc and can be final, and with a unique non initial and final vertex $\bot \notin V_G$ which is source of no arc:

$$\begin{aligned} G^{\top}_{\bot} = (G - \{\iota, o\}\times V_G) \;\cup\; \{\iota\top\,,\, o\bot\} \;\cup\; \{\; o\top \mid \exists\; s\; (\iota\, s\,,\, o\, s \in G)\;\} \\ \cup\; \{\; \top \xrightarrow{a} t \mid \exists\; s\; (s \xrightarrow[G]{a} t \;\wedge\; \iota\, s \in G)\,\} \\ \cup\; \{\; s \xrightarrow{a} \bot \mid \exists\; t\; (s \xrightarrow[G]{a} t \;\wedge\; o\, t \in G)\,\} \\ \cup\; \{\; \top \xrightarrow{a} \bot \mid \exists\; s,t\; (s \xrightarrow[G]{a} t \;\wedge\; \iota\, s\,,\, o\, t \in G)\,\}\,. \end{aligned}$$

For instance, the finite degree regular automaton G of Figure 2.1 is transformed into the following infinite degree regular automaton $G^{\top}_{\bot}$:

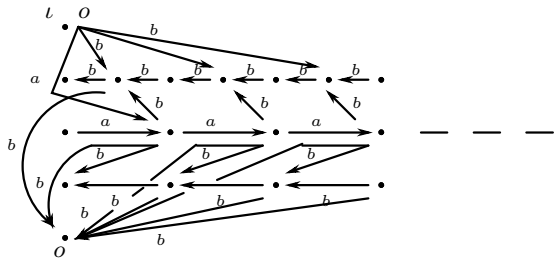


Figure 4.1 *A pointed regular automaton.*

Note that if G is unambiguous, $G^{\top}_{\bot}$ remains unambiguous. The pointed transformation of a regular automaton remains a regular automaton which can be generated by an 0-*grammar*: only the axiom has initial and final vertices. Let R be any grammar and $\top, \bot$ be two symbols which are not vertices of R. Let $G \in R^{\omega}$ with $\top, \bot \notin V_G$. We define an 0-grammar $R^{\top}_{\bot}$ generating $G^{\top}_{\bot}$ and preserving the synchronized languages: $Sync(R^{\top}_{\bot}) = Sync(R)$.

First we transform R into a grammar $\widehat{R}$ in which we memorize in the nonterminals the input vertices which are linked to initial or final vertices of the generated automaton. More precisely to any $A \in N_R$ and $I, J \subseteq [1, \varrho(A)]$, we associate a new symbol $A_{I,J}$ of arity $\varrho(A)$ with $Z = Z_{\emptyset,\emptyset}$. We define the grammar

$\widehat{R}$ assciating to each $(AX, H) \in R$ and $I, J \subseteq [1, \varrho(A)]$ the following rule:

$$A_{I,J}X \;\longrightarrow\; [H] \cup \{\; B_{I',J'}Y \mid BY \in H \;\wedge\; B \in N_R \;\}$$

with $I' = \{\; i \mid Y(i) \in I \;\vee\; \iota\, Y(i) \in H \;\}$ and $J' = \{\; j \mid Y(j) \in J \;\vee\; o\, Y(j) \in H \;\}$ and we restrict the rules of $\widehat{R}$ to the non-terminals accessible from Z.

Note that the set $L(R) \cap T$ of letters recognized by R can be determined as

$$\{\; a \mid \exists\, (A_{I,J}X, H) \in \widehat{R} \;\; (\exists\, i \in I\; \exists\, t,\; X(i) \xrightarrow[[H]]{a} t \;\wedge\; o\, t \in H)$$
$$\vee\; (\exists\, j \in J\; \exists\, s,\; s \xrightarrow[[H]]{a} X(j) \;\wedge\; \iota\, s \in H) \;\vee\; (\exists\, s, t,\; s \xrightarrow[[H]]{a} t \;\wedge\; \iota\, s\,, o\, t \in H) \;\}$$

and $\varepsilon \in L(R) \iff \exists H \in Im(\widehat{R})\; \exists\, s\; (\iota\, s\,, o\, s \in H)$.

To any $A \in N_R - \{Z\}$ and any $I, J \subseteq [1, \varrho(A)]$, we associate a new symbol $A'_{I,J}$ of arity $\varrho(A) + 2$, and we define the grammar $R^{\top}_{\perp}$ containing the axiom rule

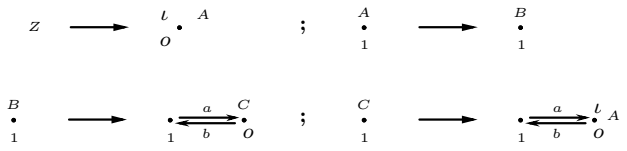
$$Z \longrightarrow H_{\emptyset,\emptyset} \cup \{\iota\top\,, o\perp\} \cup \{\; o\top \mid \varepsilon \in L(R) \;\} \cup \{\; \top \xrightarrow{a} \perp \mid a \in L(R) \cap T \;\}$$

for $(Z, H) \in \widehat{R}$, and for any $(A_{I,J}X, H) \in \widehat{R}$ with $A \neq Z$, we take in $R^{\top}_{\perp}$ the rule $A'_{I,J}\top X \perp \longrightarrow H_{I,J}$ such that $H_{I,J}$ is the following hypergraph:

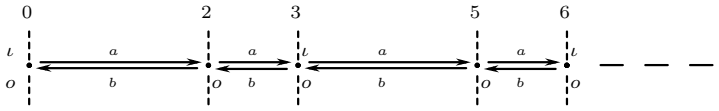
$$H_{I,J} = ([H] - \{\iota, o\}) \times V_H) \;\cup\; \{\; B'_{P,Q}\top X \perp \mid B_{P,Q}X \in H \;\wedge\; B_{P,Q} \in N_{\widehat{R}} \;\}$$
$$\cup\; \{\; \top \xrightarrow{a} t \mid \exists\, i \in I\, (X(i) \xrightarrow[[H]]{a} t) \;\vee\; \exists\, s\, (\iota\, s \in H \;\wedge\; s \xrightarrow[[H]]{a} t) \;\}$$
$$\cup\; \{\; s \xrightarrow{a} \perp \mid \exists\, j \in J\, (s \xrightarrow[[H]]{a} X(j)) \;\vee\; \exists\, t\, (o\, t \in H \;\wedge\; s \xrightarrow[[H]]{a} t) \;\}$$

and we put $R^{\top}_{\perp}$ into a terminal-outside form [Ca 07].

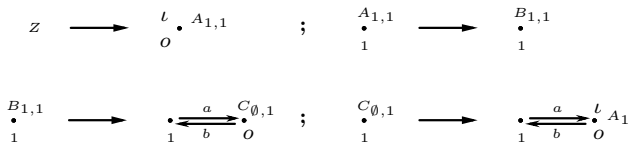
Example 4.2 Let us consider the following grammar R:



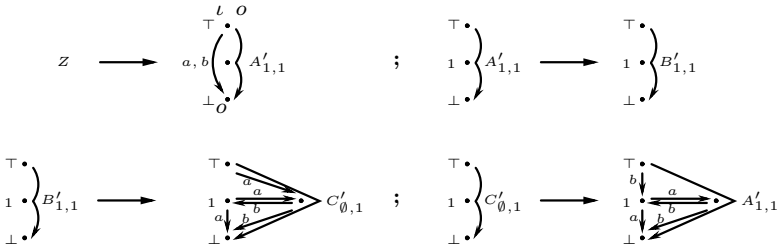
generating the following automaton G (with vertex levels):



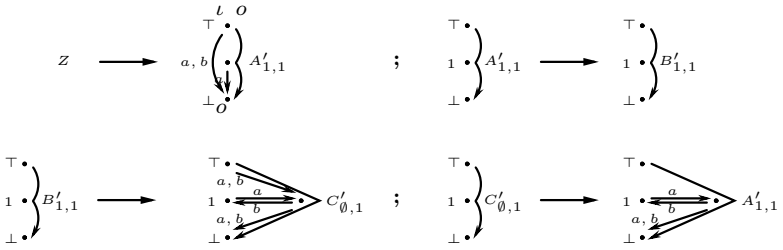
First this grammar is transformed into the following grammar $\widehat{R}$:



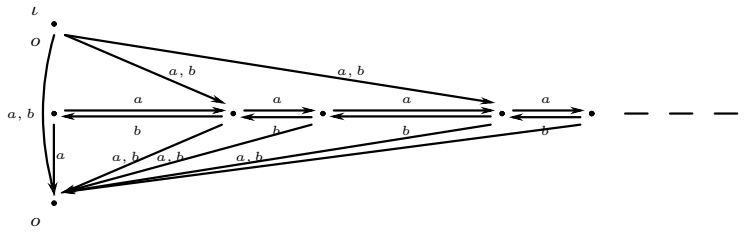
In particular $\varepsilon, a, b \in L(R)$. Then $\widehat{R}$ is transformed into the grammar $R^{\top}_{\perp}$:



that we put in a terminal-outside form:



So $R^{\top}_{\perp}$ generates $G^{\top}_{\perp}$:



□

The grammars R and $R^{\top}_{\perp}$ synchronize the same languages.

Proposition 4.3 *For any regular automaton G with $\top, \perp \notin V_G$, the pointed automaton $G^{\top}_{\perp}$ remains regular and $Sync(G^{\top}_{\perp}) = Sync(G)$.*

It follows that, in order to define families of languages by synchronization by a regular automaton G, we can restrict to pointed automata G. A stronger normalization is to transform any grammar R into a grammar S such that $Sync(S) = Sync(R)$ and S is an *arc-grammar* in the following sense: S is an 0-grammar whose any non-terminal $A \in N_S - \{Z\}$ is of arity 2, and for any non axiom rule $Ast \longrightarrow H$, there is no arc in H of goal s or of source t: for any $p \xrightarrow[H]{a} q$, we have $p \neq t$ and $q \neq s$.
We can transformed any 0-grammar R into a bi-synchronized arc-grammar $\prec R \succ$.
We assume that each rule of R is of the form $A1 \ldots \varrho(A) \longrightarrow H_A$ for any $A \in N_R$.

We take a new symbol 0 (not a vertex of R) and a new label $A_{i,j}$ of arity 2 for each $A \in N_R$ and each $i,j \in [1,\varrho(A)]$ in order to generate paths from i to j in $R^\omega(A1\ldots\varrho(A))$. We define the *splitting* $\prec G\succ$ of any F_R-hypergraph G without vertex 0 as being the graph:

$$\prec G\succ = [G] \cup \{\, X(i) \xrightarrow{A_{i,j}} X(j) \mid AX \in G \wedge A \in N_R \wedge i,j \in [\varrho(A)] \,\}$$

and for $p,q \in V_G$ and $P \subseteq V_G$ with $0 \notin V_G$, we define

$$G_{p,P,q} = \big(\{\, s \xrightarrow[\prec G\succ]{a} t \mid t \neq p \wedge s \neq q \wedge s,t \notin P \,\}\big)_{|I} \quad \text{for } p \neq q$$

$$G_{p,P,p} = \big(\{\, s \xrightarrow[\prec G\succ]{a} t \mid t \neq p \wedge s,t \notin P \,\} \cup \{\, s \xrightarrow{a} 0 \mid s \xrightarrow[\prec G\succ]{a} p \,\}\big)_{|J}$$

with $I = \{\, s \mid p \underset{\prec G\succ}{\Longrightarrow} s \underset{\prec G\succ}{\Longrightarrow} q \,\}$ and $J = \{\, s \mid p \underset{\prec G\succ}{\Longrightarrow} s \underset{\prec G\succ}{\Longrightarrow} 0 \,\}$.

This allows to define the *splitting* $\prec R\succ$ of R as being the following arc-grammar:

$$Z \longrightarrow \prec H_Z\succ$$

$$A_{i,j}12 \longrightarrow h_{i,j}\big((H_A)_{i,[\varrho(A)]-\{i,j\},j}\big) \text{ for each } A \in N_R \text{ and } i,j \in [1,\varrho(A)]$$

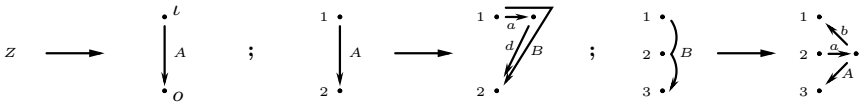
where $h_{i,j}$ is the vertex renaming defined by

$$h_{i,j}(i) = 1\,,\ h_{i,j}(j) = 2\,,\ h_{i,j}(x) = x \ \text{ otherwise, for } \ i \neq j$$

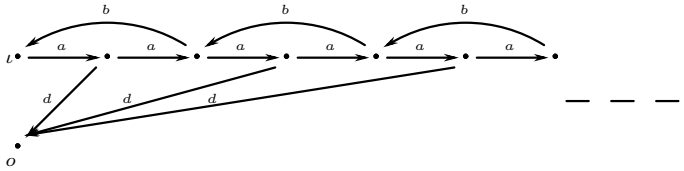
$$h_{i,i}(i) = 1\,,\ h_{i,i}(0) = 2\,,\ h_{i,i}(x) = x \ \text{ otherwise.}$$

Thus R and $\prec R\succ$ are bi-synchronized, and $\prec R\succ$ is unambiguous when R is unambiguous. Note that we can put $\prec R\succ$ into a reduced form by removing any non-terminal $A_{i,j}$ such that $\prec R\succ^\omega(A_{i,j}12)$ is without path from 1 to 2.

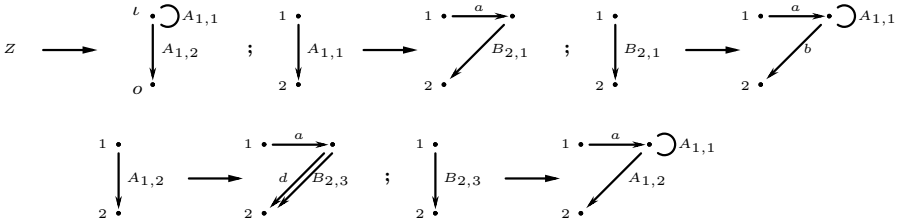
Example 4.4 The following 0-grammar R:



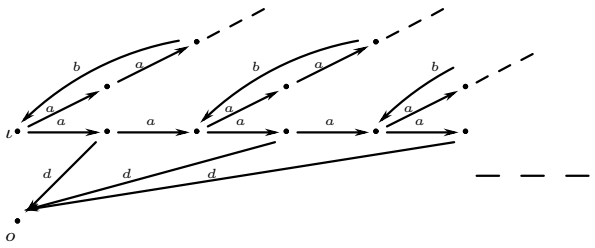
generates the following automaton G:



The splitting $\prec R\succ$ of R is the following grammar:



generating the following automaton:



As $R \bowtie \prec R \succ$, we have $Sync(R) = Sync(\prec R \succ)$.
□

To study closure properties of $Sync(R)$ for any grammar R, we can work with its normal form $\prec R^{\top}_{\perp} \succ$ which is an arc-grammar generating a pointed automaton. This normalization is really useful to study the closure property of $Sync(R)$ under complement relative to $L(R)$, under concatenation and its iteration.
We have seen that $Sync(R)$ is not closed in general under intersection, hence it is not closed under complement according to $L(R)$ since for any $L, M \subseteq L(R)$, $L \cap M = L(R) - [(L(R) - L) \cup (L(R) - M)]$. For R unambiguous, $Sync(R)$ is closed under intersection, and this remains true under complement according to $L(R)$ [Ca 08]. We give here a simpler construction.
As $\prec R^{\top}_{\perp} \succ$ remains unambiguous, we can assume that R is an arc-grammar. Let $S \lhd R$. We want to show that $L(R) - L(S) \in Sync(R)$. So S is an 0-grammar and S is *level-unambiguous* as defined in [Ca 08] : for any accepting paths λ, μ with the same label u and for every prefix v of u, the prefixes of λ and μ labelled by v lead to vertices of the same level *i.e.* for (any) $G \in S^{\omega}$,

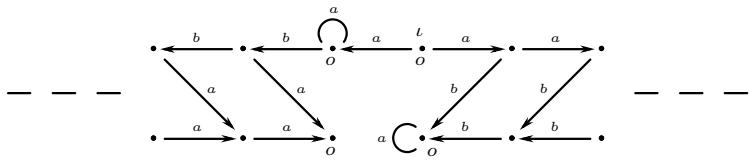
$$s_0 \xrightarrow[G]{a_1} s_1 \dots \xrightarrow[G]{a_n} s_n \ \wedge \ t_0 \xrightarrow[G]{a_1} t_1 \dots \xrightarrow[G]{a_n} t_n \ \wedge \ \iota\, s_0\,,\, \iota\, t_0\,,\, o\, s_n\,,\, o\, t_n \in G$$
$$\Longrightarrow \ \ell^S_G(s_i) = \ell^S_G(t_i) \quad \forall\, i \in [0, n]\,.$$

Thus $\prec S \succ$ is a level-unambiguous arc-grammar. We take a new colour $c \in F_1 - \{\iota, o\}$ and for any grammar S', we denote S'_c (resp. $S'_{\overline{c}}$) the grammar obtained from S' by replacing the final colour o by c (resp. c by o). So $R + \prec S \succ_c$ is an arc-grammar and $(R + \prec S \succ_c)_{\overline{c}}$ is level-unambiguous. It remains to apply the grammar determinization defined in [Ca 08] and given below, to get the grammar $R/S = Det(R + \prec S \succ_c)$ such that $(R/S)_{\overline{c}}$ is unambiguous and bi-synchronized to $(R + \prec S \succ_c)_{\overline{c}}$. Finally we keep in R/S the final vertices which are not coloured by c to obtain a grammar synchronized by R and recognizing $L(R) - L(S)$.

Theorem 4.5 *For any unambiguous regular automaton G, the set $Sync(G)$ is an effective boolean algebra according to $L(G)$, containing all the regular languages included in $L(G)$.*

So we can decide the inclusion $L(S) \subseteq L(S')$ for two grammars S and S' synchronized by a common unambiguous grammar. Furthermore for grammars R_1 and R_2 such that $R_1 + R_2$ is level-unambiguous, $Sync(R_1 + R_2) = \{\ L_1 \cup L_2 \mid L_1 \in Sync(R_1) \ \wedge \ L_2 \in Sync(R_2)\ \}$ is a boolean algebra included in $L(R_1) \cup L(R_2)$,

containing $Sync(R_1)$ and $Sync(R_2)$.
The automata of Examples 3.9 to 3.16 are unambiguous hence their families of synchronized languages are boolean algebra. This regular automaton G:



is 2-ambiguous: there are two accepting paths for the words $a^n b^n a^n$ with $n > 0$ and a unique accepting path for the other accepted words. But $Sync(G)$ is not closed under intersection since $\{\ a^m b^m a^n \mid m, n \geq 0\ \}$ and $\{\ a^m b^n a^n \mid m, n \geq 0\ \}$ are languages synchronized by G.

Let us give the Det operation applied on any arc-grammar.
As for the level synchronization product, the standard powerset construction to determinize a graph is only done level preserving.
The *level-determinization* of any grammar R is

$$\mathrm{Det}(R^\omega) \ := \ \{\ K \mid \exists\ G \in R^\omega, K \ \text{ isomorphic to } \mathrm{Det}(G)\ \}$$

whose the level-determinization $\mathrm{Det}(G)$ of any $G \in R^\omega$ is defined by

$$\begin{aligned}\mathrm{Det}(G) := \{\ P \xrightarrow{a} Q \mid P, Q \in \Pi \ \wedge\ Q \subseteq Succ_a(P) \ \wedge \\ \forall\ q \in Succ_a(P) - Q,\ Q \cup \{q\} \notin \Pi\ \} \\ \cup\ \{\ \iota\, P \mid P \in \Pi \ \wedge\ \forall\ p \in P\ \iota\, p \in G \ \wedge \\ \forall\ q\ (\iota\, q \in G \ \wedge\ q \notin P \implies P \cup \{q\} \notin \Pi)\ \} \\ \cup\ \{\ cP \mid P \in \Pi \ \wedge\ c \in F_1 - \{\iota\} \ \wedge\ \exists\ p \in P \ \ cp \in G\ \}\end{aligned}$$

restricted to the vertices accessible from ι and such that Π is the set of subsets of vertices with same level:

$$\Pi \ := \ \{\ P \mid \emptyset \neq P \subseteq V_G \ \wedge\ \forall\ p, q \in P,\ \ell(p) = \ell(q)\ \}$$

and $Succ_a(P)$ is the set of successors of vertices in $P \in \Pi$ by $a \in F_G \cap F_2$:

$$Succ_a(P) \ := \ \{\ q \mid \exists\ p \in P\ (p \xrightarrow[G]{a} q)\ \}.$$

Contrary to the level synchronization product, Det does not preserve the regularity.
However $\mathrm{Det}(R^\omega)$ can be generated by a grammar when R is an arc-grammar.
Let R be any arc grammar with R^ω accessible from ι.
We denote H_A the right hand side of the rule of $A \in N_R$.
To any $A \in N_R - \{Z\}$, we associate a new symbol $\overline{A}$ of arity 2 and we define the grammar $\overline{R}$ obtained from R by adding the rules $\overline{A}12 \longrightarrow H_A$ for all $A \in N_R - \{Z\}$, and then by replacing in the right hand sides any non-terminal arc $s \xrightarrow{B} 2$ by $s \xrightarrow{\overline{B}} 2$:

$$\begin{aligned}\overline{R} := & \{\, (Z, H_Z) \,\} \\ & \cup \{\, \big(A12\,,\, (H_A - N_R V_{H_A} 2) \cup \{\, \overline{B}s2 \mid B \in N_R \ \wedge\ Bs2 \in H_A \,\}\big) \\ & \qquad\qquad \mid A \in N_R - \{Z\} \,\} \\ & \cup \{\, \big(\overline{A}12\,,\, (H_A - N_R V_{H_A} 2) \cup \{\, \overline{B}s2 \mid B \in N_R \ \wedge\ Bs2 \in H_A \,\}\big) \\ & \qquad\qquad \mid A \in N_R - \{Z\} \,\}\,.\end{aligned}$$

We take a linear order $<$ on $2^{N_{\overline{R}}-\{Z\}}$ of smallest element $\emptyset$ (Z does not appear in the right hand side of R). To each $\emptyset \neq P \subseteq N_{\overline{R}} - \{Z\}$, we associate

a new symbol P' of arity $2^{|P|}$

a hyperarc $<P> \ = \ P' p_1 \ldots p_m$ with $\{p_1, \ldots, p_m\} = 2^P$ and $p_1 < \ldots < p_m$

and we take a graph H_P such that

$$\{\, Z \xrightarrow{A} A \mid A \in P \,\} \ \cup\ \{\iota\, Z\} \ \underset{\overline{R}}{\Longrightarrow}\ H_P$$

and for $P = \emptyset$, we define $<\emptyset> \ = \ Z$ and $H_\emptyset = H_Z$.

To each $P \subseteq N_{\overline{R}} - \{Z\}$, we apply on H_P the level-determinization to get the graph

$$H'_P \ := \ \mathrm{Det}(H_P)[\emptyset/\{Z\}] - \{\iota\, \emptyset\}$$

whose the vertex level ℓ is defined by

$\ell(A) = 0 \ \ \forall\ A \in P - N_R$; $\ell(A) = 1 \ \ \forall\ A \in P \cap N_R$; $\ell(s) = 2 \ \ \forall\ s \in V_{H_P} - (P \cup \{Z\})$.

Note that the level $\ell(Z)$ of Z is not significant because there is no arc of goal Z in H_P. To each $P \subseteq N_{\overline{R}} - \{Z\}$, we associate the following rule:

$$<P> \ \longrightarrow \ [H'_P] \cup \{\, <Q>[U_E/E]_{E \subseteq Q} \mid U \subseteq V_{H'_P} \ \wedge\ Q \neq \emptyset \,\}$$

with

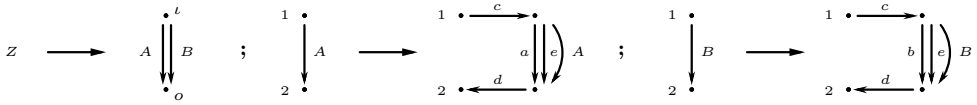
$$\begin{aligned} Q &:= \{\, A \in N_{\overline{R}} \mid \exists\ s \in U,\ s \xrightarrow[H'_P]{A} \,\} \\ U_\emptyset &:= U \\ U_E &:= \{\, t \mid \exists\ s \in U\ \exists\ A \in E,\ s \xrightarrow[H'_P]{A} t \,\} \quad \text{for any } \emptyset \neq E \subseteq Q\,. \end{aligned}$$

Note that for R unambiguous, we can restrict $<P>$ to

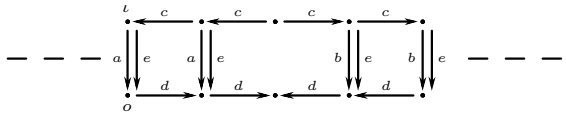
$$<P> \ = \ P' p_1 \ldots p_m \ \text{ with } \ \{p_1, \ldots, p_m\} = P.$$

By taking all the rules accessible from Z, we get a grammar $\mathrm{Det}(R)$.

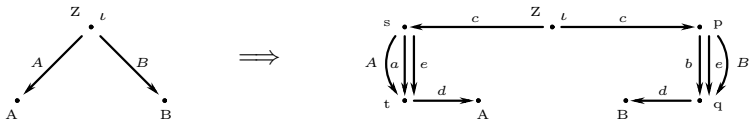
Let us illustrate the construction of $\mathrm{Det}(R)$ to the following arc grammar R:



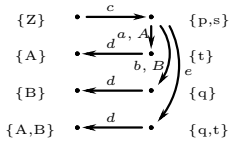
generating the following graph G:



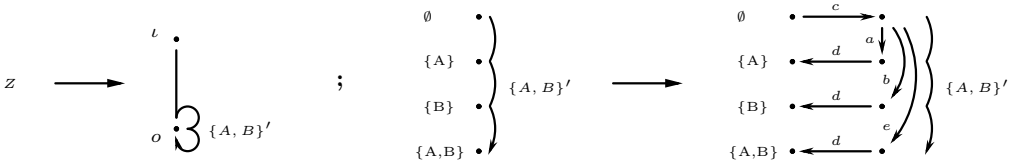
We have the following parallel rewriting:



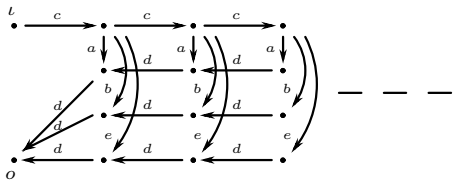
Taking $\ell(A) = \ell(B) = 1$ and $\ell(s) = \ell(t) = \ell(p) = \ell(q) = 2$, the right hand side $H_{A,B}$ gives by level-determinization the following graph $\mathrm{Det}(H_{A,B})$:



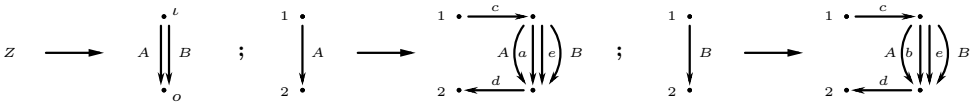
and the following grammar $\mathrm{Det}(R)$:



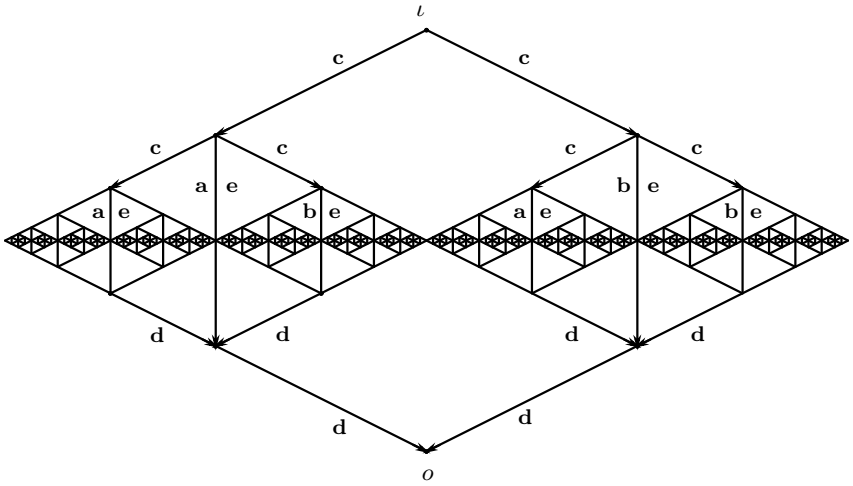
generating $\mathrm{Det}(G)$:



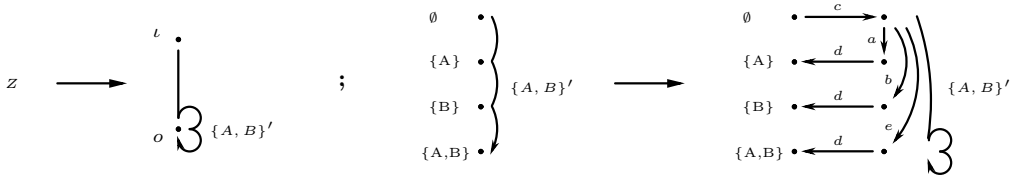
A similar example is given by the following arc grammar R:



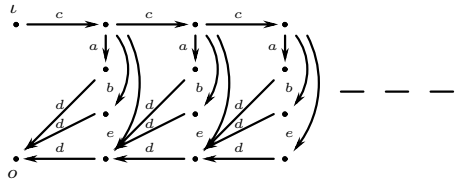
generating the following graph G:



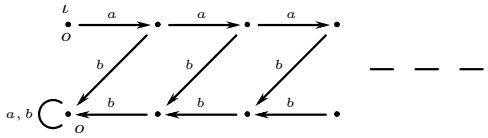
We obtain the following grammar $\mathrm{Det}(R)$:



generating $\mathrm{Det}(G)$:



For any regular automaton G, the closure of $Sync(G)$ under concatenation $\cdot$ (resp. under its transitive closure $^+$) does not require the unambiguity of G. As $L(G) \in Sync(G)$, a necessary condition is to have $L(G).L(G) \in Sync(G)$ (resp. $L(G)^+ \in Sync(G)$). Note that this necessary condition implies that $L(G)$ is closed under $\cdot$ (resp. $^+$). In particular $Sync(G)$ is not closed under $\cdot$ and $^+$ for the automata of Examples 3.12 to 3.16. But this necessary condition is not sufficient since the following regular automaton G:



recognizes $L(G) = \varepsilon + M(a+b)^*$ for $M = \{\, a^n b^n \mid n > 0 \,\}$, hence $L(G).L(G) = L(G) = L(G)^+$ but $M \in Sync(G)$ and $M.M, M^+ \notin Sync(G)$.
Let us give a simple and general condition on a grammar R such that $Sync(R)$ is closed under $\cdot$ and $^+$. We say that a grammar is *iterative* if any initial vertex is in the axiom and for (any) $G \in R^\omega$ and any accepting path $s_0 \xrightarrow[G]{a_1} s_1 \ldots \xrightarrow[G]{a_n} s_n$ with $\iota\, s_0\,,\, o\, s_n \in G$ and for any final vertex t *i.e.* $o\, t \in G$, there exists a path $t \xrightarrow[G]{a_1} t_1 \ldots \xrightarrow[G]{a_n} t_n$ with $o\, t_n \in G$ such that $\ell(t_i) = \ell(t) + \ell(s_i)$ for all $i \in [1, n]$.
For instance the automaton of Example 3.10 can be generated by an iterative grammar. And any 0-grammar generating a regular automaton having a unique initial vertex which is the unique final vertex, is iterative. Standard constructions on finite automata for the concatenation and its iteration can be extended to iterative grammars.

Proposition 4.6 *For any iterative grammar R, the family $Sync(R)$ is closed under concatenation and its transitive closure.*

However the automaton G of Example 3.11 cannot be generated by an iterated grammar but $Sync(G)$ is closed under $\cdot$ and $^+$ [AM 04]. We can also obtain families of synchronized languages which are closed under $\cdot$ and $^+$ by saturating grammars. The *saturation* G^+ of an automaton G is the automaton

$$G^+ \;=\; G \;\cup\; \{\; s \xrightarrow{a} r \mid \iota\, r \in G \;\wedge\; \exists\, t\; (s \xrightarrow[G]{a} t \;\wedge\; o\, t \in G) \,\}$$

recognizing $L(G^+) = (L(G))^+$.
Note that if G is regular with infinite sets of initial and final vertices, G^+ can be non regular (but is always prefix-recognizable). If G is generated by an 0-grammar R, its saturation G^+ can be generated by a grammar R^+ that we define.
Let (Z, H) be the axiom rule of R and $r_1, \ldots, r_p$ be the initial vertices of H ; we can assume that $r_1, \ldots, r_p$ are not vertices of $R - \{(Z, H)\}$. To each $A \in N_R - \{Z\}$ and $I \subseteq [1, \varrho(A)]$, we associate a new symbol A_I of arity $\varrho(A) + p$ and we define R^+ with the following rules:

$$Z \longrightarrow [H]^+ \cup \{\; A_{\{\, i \mid o\, X(i) \in H \,\}} X r_1 \ldots r_p \mid AX \in H \;\wedge\; A \in N_R \,\}$$
$$A_I X r_1 \ldots r_p \longrightarrow K_I \quad \text{for each } (AX, K) \in R \text{ and } A \neq Z \text{ and } I \subseteq [1, \varrho(A)]$$

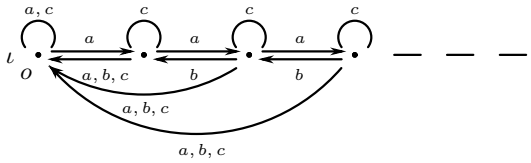
whose K_I is the automaton obtained from K as follows:

$$K_I = [K] \;\cup\; \{\; s \xrightarrow{a} r_j \mid j \in [p] \;\wedge\; \exists\, i \in I\; (s \xrightarrow[K]{a} X(i)) \,\}$$
$$\cup\, \{\; B_{\{\, j \mid \exists\; i \in I,\; Y(j) = X(i) \,\}} Y r_1 \ldots r_p \mid BY \in K \;\wedge\; B \in N_R \,\}\,.$$

So R is synchronized by R^+ and $G^+ \in (R^+)^\omega$ for $G \in R^\omega$.
To characterize $Sync(R^+)$ from $Sync(R)$, we define the *regular closure* $Reg(E)$ of any language family E as being the smallest family of languages containing E and closed under $\cup$, $\cdot$, $+$.

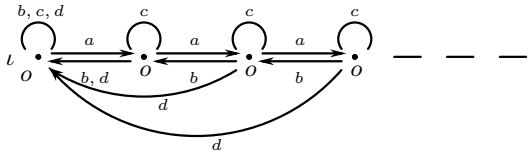
Proposition 4.7 *For any 0-grammar R, $Sync(R^+) = Reg(Sync(R))$.*

By Propositions 4.3, 4.6 and 4.7, the following regular automaton G:

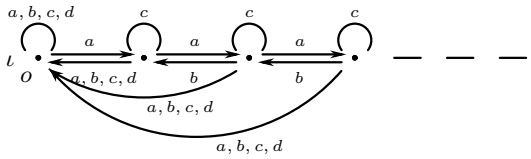


has the same synchronized languages than the automaton of Example 3.10 : $Sync(G)$ is the family of input-driven languages (for a pushing, b popping and c internal). By adding an b-loop on the initial (and final) vertex of G, we obtain an automaton H such that $Sync(H)$ is the family of visibly pushdown languages hence by Proposition 4.7, is closed under $\cdot$ and $^+$.

Example 4.8 A natural extension of the visibly pushdown languages is to add reset letters. For a pushing, b popping and c internal, we add a reset letter d to define the following regular automaton G :

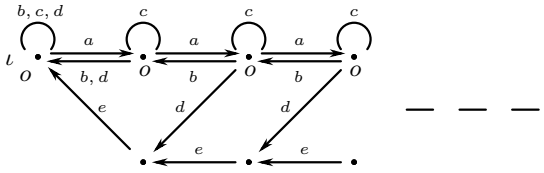


Any language of $Sync(G)$ is a visibly pushdown language taking d as an internal letter, but not the converse: $\{\ a^n d b^n \mid n \geq 0\ \} \notin Sync(G)$. By Theorem 4.5, $Sync(G)$ is a boolean algebra. Furthermore the following automaton H :



satisfies $Sync(H) = Sync(G)$ and $H^+ = H$ hence by Proposition 4.7, $Sync(G)$ is also closed under $\cdot$ and $^+$.
□

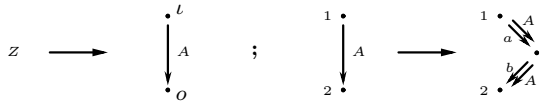
Note that the automata of the previous example have infinite degree. Furthermore for any automaton G of finite degree having an infinite set of initial or final vertices, the pointed automaton $G^\top_\bot$ is of infinite degree. However any regular automaton of infinite degree (in fact any prefix-recognizable automaton) can be obtained by ϵ-closure from a regular automaton of finite degree using ε-transitions. For instance let us take a new letter $e \notin T$ (instead of the empty word) and let us denote π_e the morphism erasing e in the words over $T \cup \{e\}$: $\pi_e(a) = a$ for any $a \in T$ and $\pi_e(e) = \varepsilon$, that we extend by union to any language $L \subseteq (T \cup \{e\})^*$: $\pi_e(L) = \{\ \pi_e(u) \mid u \in L\ \}$, and by powerset to any family P of languages: $\pi_e(P) = \{\ \pi_e(L) \mid L \in P\ \}$. The following regular automaton K :



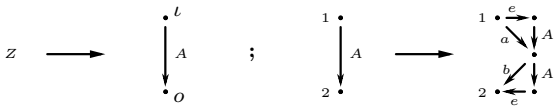
is of finite degree and satisfies $\pi_e(Sync(K)) = Sync(G)$ for the automaton G of Example 4.8. Let us give a simple transformation of any grammar R to a grammar R_e such that R_e^ω is of finite degree and $\pi_e(Sync(R_e)) = Sync(R)$.
As $Sync(R) = Sync(\prec R_\perp^\top \succ)$, we restrict this transformation to arc-grammars. Let R be an arc-grammar. We define R_e to be an arc-grammar obtained from R by replacing each non axiom rule $Ast \longrightarrow H$ by the rule:

$$Ast \longrightarrow \big([H] \ \cup \ \{s \xrightarrow{e} s_e\,,\, t_e \xrightarrow{e} t\} \ \cup \ h(H - [H])\big)_{|P}$$

with s_e, t_e be new vertices and h the vertex mapping defined for any $r \in V_H$ by $h(r) = r$ if $r \notin \{s,t\}$, $h(s) = s_e$ and $h(t) = t_e$, and P is the set of vertices accessible from s and co-accessible from t. For instance the arc-grammar R



is transformed into the following arc-grammar R_e :



For any rule of R_e , the inputs are separated from the outputs (by e-transitions), hence R_e^ω is of finite degree. Furthermore this transformation preserves the synchronized languages.

Proposition 4.9 *For any arc-grammar R, $Sync(R) = \pi_e(Sync(R_e))$.*

So for any R, $Sync(R) = \pi_e(Sync(\prec R_\perp^\top \succ_e))$ and $(\prec R_\perp^\top \succ_e)^\omega$ is of finite degree.

All the constructions given in this paper are natural generalizations of usual transformations on finite automata to graph grammars. In this way, basic closure properties could be lifted to sub-families of context-free languages.

Conclusion

The synchronization of regular automata is defined through devices generating these automata, namely functional graph grammars. It can also be defined using pushdown automata with ε-transitions [NS 07] because Theorem 3.8 asserts that the family of languages synchronized by a regular automaton is independent of the way the automaton is generated; it is a graph-related notion. This

paper shows that the mechanism of functional graph grammars provides natural constructions on regular automata generalizing usual constructions on finite automata. This paper is also an invitation to extend the notion of synchronization to more general sub-families of automata.

Acknowledgements

Many thanks to Arnaud Carayol and Antoine Meyer for helping me prepare the final version of this paper.

References

[AM 04] R. Alur and P. Madhusudan *Visibly pushdown languages*, 36^{th} STOC, ACM Proceedings, L. Babai (Ed.), 202–211 (2004).

[Be 79] J. Berstel *Transductions and context-free languages*, Ed. Teubner, pp. 1–278, 1979.

[BB 02] J. Berstel and L. Boasson *Balanced grammars and their languages*, Formal and Natural Computing, LNCS 2300, W. Brauer, H. Ehrig, J. Karhumäki, A. Salomaa (Eds.), 3–25 (2002).

[Ca 06] D. Caucal *Synchronization of pushdown automata*, 10^{th} DLT, LNCS 4036, O. Ibarra, Z. Dang (Eds.), 120-132 (2006).

[Ca 07] D. Caucal *Deterministic graph grammars*, Texts in Logic and Games 2, Amsterdam University Press, J. Flum, E. Grädel, T. Wilke (Eds.), 169–250 (2007).

[Ca 08] D. Caucal *Boolean algebras of unambiguous context-free languages*, 28^{th} FSTTCS, Dagstuhl Research Online Publication Server, R. Hariharan, M. Mukund, V. Vinay (Eds.) (2008).

[CH 08] D. Caucal and S. Hassen *Synchronization of grammars*, 3^{rd} CSR, LNCS 5010, E. Hirsch, A. Razborov, A. Semenov, A. Slissenko (Eds.), 110–121 (2008).

[Ha 78] M. Harrison *Introduction to formal language theory*, Addison-Wesley (1978).

[Me 80] K. Mehlhorn *Pebbling mountain ranges and its application to DCFL recognition*, 7^{th} ICALP, LNCS 85, J. de Bakker, J. van Leeuwen (Eds.), 422–432 (1980).

[MS 85] D. Muller and P. Schupp *The theory of ends, pushdown automata, and second-order logic*, Theoretical Computer Science 37, 51–75 (1985).

[NS 07] D. Nowotka and J. Srba *Height-deterministic pushdown automata*, 32^{nd} MFCS, LNCS 4708, L. Kucera, A. Kucera (Eds.), 125–134 (2007).