Synchronization of grammars

Didier Caucal
IGM–CNRS
caucal@univ-mlv.fr

Stéphane Hassen
IREMIA
shassen@univ-reunion.fr

Abstract. Deterministic graph grammars are finite devices which generate the transition graphs of pushdown automata. We define the notion of synchronization by grammars, generalizing previous sub-classes such as visibly and height-deterministic pushdown automata. The languages recognized by grammars synchronized by a given grammar form an effective boolean algebra lying between regular languages and deterministic context-free languages. We also provide a sufficient condition to obtain the closure under concatenation and its iteration.

1 Introduction

In recent literature, several restrictions of pushdown automata have been studied in order to define classes of languages which generalize regular languages while retaining good closure properties (namely closure under boolean operations, concatenation and its iteration). All these approaches consist in defining a notion of synchronization between pushdown automata.
The first such approach is to partition the input alphabet between pushing, popping and internal symbols, and to impose that stack movement only depend on the type of symbol read, yielding the so-called visibly pushdown automata [AM 04]. This enforces that the stack height variation be entirely characterized by the input word.
A first generalization is to replace the partition of the input alphabet by a finite transducer assigning a weight to every input word. A pushdown automaton is synchronized by a transducer if two initial computations ending in the same configuration are labelled by words of the same weight, and we can only reach a finite number of configurations by initial computations of a given weight [Ca 06].
A last generalization, defining the height-deterministic pushdown automata, is to synchronize a pushdown automaton by another pushdown automaton. The notion of synchronization is simply that two initial computations with the same input word end in configurations with the same height [NS 07].
The classes of languages accepted by pushdown automata synchronized by a given partition of the input alphabet [AM 04], a finite transducer [Ca 06] or a pushdown automaton [NS 07] are boolean algebras but, for the last two approaches, are not in general closed under concatenation and its iteration.
Instead of using the stack height of pushdown automata and doing a special treatment of the ε-moves, a general approach is to define the synchronization at a graph level. The transition graphs of the pushdown automata are generated

by infinite parallel rewritings by (deterministic graph) grammars [MS 85,Ca 07]. The weight of a vertex in the generated graph is defined as the minimal number of steps of parallel rewritings necessary to produce it. The notion of synchronization is defined for grammars generating deterministic graphs which can have vertices of infinite in-degree. This allows a uniform treatment of real-time and non-real-time deterministic pushdown automata.
A grammar G is synchronized by a grammar H if for any initial path of (the graph generated by) G, there exists an initial path of H with the same label and these paths end in vertices of same weight. By extending usual constructions from finite automata to grammars, we show that the languages recognized by all grammars synchronized with a given grammar form a boolean algebra containing the regular languages, and we provide a simple sufficient condition for the closure under concatenation and its iteration.
It appears that the boolean algebras yielded by the previous notions of synchronization can all be captured using synchronization by grammars. We also show that the family of balanced languages [BB 02], which is not synchronized according to the previous notions, fit our formalism.

2 Deterministic graph grammars

In this section, we recall the notion of deterministic graph grammar together with the family of graphs they generate: the regular graphs. For these graphs, we introduce the level of a vertex and for deterministic regular graphs, we define the weight of the label of an initial path. We end with a classical result on their recognized languages: the labels of their accepting paths are the context-free languages, and are the deterministic (resp. and real-time) context-free languages by restricting to deterministic regular graphs (resp. and of finite degree).

Let $\mathbb{N}$ be the set of natural numbers. For a set E, we write $|E|$ its cardinality, 2^E its powerset and for any $n \geq 0$, $E^n = \{ (e_1, \ldots, e_n) \mid e_1, \ldots, e_n \in E \}$ is the set of n-tuples of elements of E. Thus $E^* = \bigcup_{n \geq 0} E^n$ is the free monoid generated by E for the *concatenation*: $(e_1, \ldots, e_m) \cdot (e'_1, \ldots, e'_n) = (e_1, \ldots, e_m, e'_1, \ldots, e'_n)$, and whose neutral element is the 0-tuple $()$. A finite set E of symbols is an *alphabet* of *letters*, and E^* is the set of *words* over E. Any word $u \in E^n$ is of *length* $|u| = n$ and is also represented by a mapping from $[n] := \{1, \ldots, n\}$ into E, or by the juxtaposition of its letters: $u = u(1) \ldots u(|u|)$. The neutral element is the word of length 0 called the *empty word* and denoted by ε.
Let F be a set of symbols called *labels*, ranked by a mapping $\varrho : F \longrightarrow \mathbb{N}$ associating to each label f its *arity* $\varrho(f) \geq 0$, and such that

$$F_n \ := \ \{ \ f \in F \mid \varrho(f) = n \ \} \text{ is countable for every } n \geq 0.$$

We consider simple, oriented and labelled hypergraphs: a *hypergraph* G is a subset of $\bigcup_{n \geq 0} F_n V^n$, where V is an arbitrary set, such that

its *vertex* set $V_G \ := \ \{ \ v \in V \mid FV^* v V^* \cap G \neq \emptyset \ \}$ is finite or countable,
its *label* set $F_G \ := \ \{ \ f \in F \mid fV^* \cap G \neq \emptyset \ \}$ is finite.

Any $f v_1 \ldots v_{\varrho(f)} \in G$ is a *hyperarc* labelled by f and of successive vertices

$v_1, \ldots, v_{\varrho(f)}$; it is depicted for

$\varrho(f) \geq 2$ as an arrow labelled f and successively linking $v_1, \ldots, v_{\varrho(f)}$;
$\varrho(f) = 1$ as a label f on vertex v_1 and f is called a *colour* of v_1 ;
$\varrho(f) = 0$ as an isolated label f called a *constant*.

This is illustrated in the figures throughout the paper. Note that a vertex v is depicted by a dot named (v) where parentheses are used to differentiate a vertex name from a vertex label (a colour).

For a subset $E \subseteq F$ of labels, we write

$$V_{G,E} \ := \ \{\ v \in V \mid EV^*vV^* \cap G \neq \emptyset\ \} \ = \ V_G \cap EV_G^*$$

the set of vertices of G linked by a hyperarc labelled in E.

A *graph* G is a hypergraph whose labels are only of arity 1 or 2 : $F_G \subset F_1 \cup F_2$. Hence a graph G is a set of *arcs* av_1v_2 identified with the labelled transition $v_1 \xrightarrow[G]{a} v_2$ or directly $v_1 \xrightarrow{a} v_2$ if G is understood, plus a set of coloured vertices $f\,v$. A tuple $(v_0, a_1, v_1, \ldots, a_n, v_n)$ for $n \geq 0$ and $v_0 \xrightarrow[G]{a_1} v_1 \ \ldots \ v_{n-1} \xrightarrow[G]{a_n} v_n$ is a *path* from v_0 to v_n labelled by $u = a_1 \ldots a_n$; we write $v_0 \overset{u}{\underset{G}{\Longrightarrow}} v_n$ or directly $v_0 \overset{u}{\Longrightarrow} v_n$ if G is understood. For $P, Q \subseteq V_G$ and $u \in F_2^*$, we write

$$P \overset{u}{\underset{G}{\Longrightarrow}} Q \ \text{ if } \ p \overset{u}{\underset{G}{\Longrightarrow}} q \quad \text{for some } p \in P \text{ and } q \in Q$$
$$\text{and } \ L(G, P, Q) := \{\ u \mid P \overset{u}{\underset{G}{\Longrightarrow}} Q\ \}$$

is the language recognized by G from P to Q. In these notations, we can replace P (and/or Q) by a colour i to designate the subset $V_{G,i}$. In particular $i \overset{u}{\underset{G}{\Longrightarrow}} Q$ means that there is a path labelled by u from a vertex coloured by i to a vertex in Q, and $L(G, i, f)$ is the label set of the paths from a vertex coloured by i to a vertex coloured by f.

In this paper, we only use two colours $i, f \in F_1$ to mark respectively the initial vertices and the final vertices. For any graph G, we denote

$$L(G) \ := \ L(G, i, f) \quad \text{the } \textit{language recognized} \text{ by } G$$
$$L(G, i) \ := \ L(G, i, V_G) \text{ the } \textit{complete language recognized} \text{ by } G.$$

Recall that the *regular languages* over an alphabet $T \subset F_2$ form the set:

$$Rat(T^*) \ := \ \{\ L(G) \mid G \text{ finite } \wedge \ F_G \ \subseteq T \cup \{i, f\}\ \}.$$

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 joining pairwise distinct vertices $x_1 \neq \ldots \neq x_{\varrho(f)}$ and H is a finite hypergraph with $\{x_1, \ldots, x_{\varrho(f)}\} \ \subseteq \ V_H$; we denote by

$N_R := \{\, f \in F \mid \exists\ x_1, \ldots, x_{\varrho(f)}\ fx_1 \ldots x_{\varrho(f)} \in Dom(R)\,\}$ the *non-terminals* of R, the labels of the left hand sides,

$T_R := \{\ f \in F - N_R \mid \exists\ P \in Im(R),\ V_{P,f} \neq \emptyset\ \}$ the *terminals* of R, the labels of R which are not non-terminals,

$F_R := N_R \cup T_R$ the labels of R.

We use grammars to generate graphs. Hence in the following, we may assume that any terminal is of arity 1 or 2 : $T_R \subset F_1 \cup F_2$.

As for 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 unique constant non-terminal $Z \in N_R \cap F_0$; the *axiom* of R is the right hand side H of the rule of Z : $Z \longrightarrow H$.
To simplify, we add the condition that i only colours vertices of the axiom.
Starting from the axiom, we want that R generates a unique graph up to isomorphism. So we finally assume that any grammar R is *deterministic* meaning that there is only one rule per non-terminal:

$$(X, H)\,, (Y, K) \in R \ \wedge \ X(1) = Y(1) \quad \Longrightarrow \quad (X, H) = (Y, K).$$

For any rule $X \longrightarrow H$, we say that

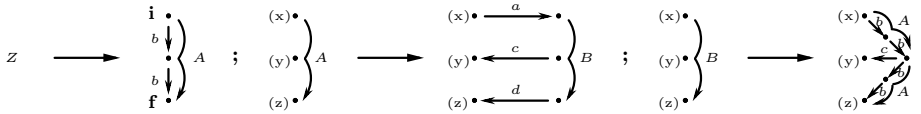
$V_X \cap V_H$ are the *inputs* of H

and $\bigcup\{\ V_Y \mid Y \in H \ \wedge \ Y(1) \in N_R\ \}$ are the *outputs* of H.

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 a non input vertex:

$$H \ \cap \ (T_R V_X V_X \cup T_R V_X) \ = \ \emptyset \quad \text{for any rule } \ (X, H) \in R.$$

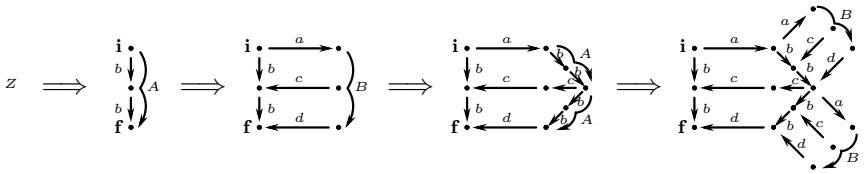
We will use upper-case letters $A, B, C, \ldots$ for non-terminals and lower-case letters $a, b, c \ldots$ for terminals. Here is an example of a (deterministic graph) grammar R :



For this grammar R, we have $N_R = \{Z, A, B\}$, $T_R = \{a, b, c, d\}$ and x, y, z are the inputs of the last two rules (the axiom rule having no input).
Given a grammar R, the *rewriting* $\underset{R}{\longrightarrow}$ 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 the 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.
Taking the previous grammar, we depict in the next figure the first four steps of the parallel derivation from its constant non-terminal Z :



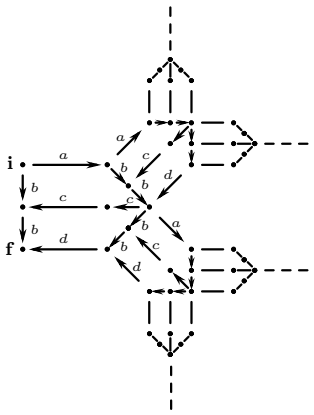
Given a deterministic grammar R and a hypergraph H, we denote

$$[H] \;:=\; H \cap T_R V_H^* \;=\; H \cap (T_R V_H V_H \cup T_R V_H)$$

the set of terminal arcs and of terminal coloured vertices of H.
A graph G is *generated* by R (from its axiom) if G belongs to the following set R^ω of isomorphic graphs:

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

For instance by iterating indefinitely the previous derivation, we get the infinite graph depicted below.



Grammars R and S are *equivalent* if they generate the same graph(s): $R^\omega = S^\omega$. A *regular graph* is a graph generated by a (deterministic graph) grammar.
Given a (regular) graph G generated by a grammar R:

$$G \;=\; \textstyle\bigcup_{n\geq 0}[H_n] \quad \text{with} \quad Z \underset{R}{\longrightarrow} H_0 \underset{R}{\Longrightarrow} \dots H_n \underset{R}{\Longrightarrow} H_{n+1} \dots$$

we define the *level* $\ell(s)$ of a vertex $s \in V_G$, denoted also $\ell_G^R(s)$ to precise G and R, as being

$$\ell(s) \;:=\; \min\{\, n \mid s \in V_{H_n} \,\}$$

the minimal number of rewritings applied from the axiom to get s.
For any grammar R and for $G \in R^\omega$, we denote

$$L(R) \;:=\; L(G) \quad \text{the } \textit{language recognized} \text{ by } R$$
$$L(R,i) \;:=\; L(G,i) \text{ the } \textit{complete language recognized} \text{ by } R.$$

These languages are well-defined since generated graphs by a grammar are isomorphic. For instance, the previous grammar R recognizes the language $L(R)$ generated from A by the following context-free grammar:

$$A \;=\; bb \;+\; aAAd \;+\; aAccb$$

As the generated graph(s) by R is co-accessible from f (from any vertex, there is a path to a final vertex), $L(R,i)$ is the set of prefixes of the words in $L(R)$.
A graph G is (label) *complete* if for any arc label $a \in F_G \cap F_2$, any vertex $s \in V_G$ is source of an a-arc: $\exists\, t,\ s \xrightarrow[G]{a} t$. For a grammar R generating a complete graph R^ω, we have $L(R,i) \;=\; (T_R - \{i,f\})^*$.
A graph G is *deterministic* if i colours a unique vertex, and two arcs with the same source have distinct labels: $r \xrightarrow[G]{a} s \;\wedge\; r \xrightarrow[G]{a} t \;\Longrightarrow\; s = t$.
For a deterministic graph G generated by R, we can define the *weight* $\|\, u \,\|_R$

of any word $u \in L(R,i)$ by the level of the ending vertex of the initial path labelled by u:

$$\| u \|_R := \ell(s) \quad \text{for} \quad i \stackrel{u}{\underset{G}{\Longrightarrow}} s.$$

For the previous grammar R, we have $\| \varepsilon \|_R = \| b \|_R = \| bb \|_R = 0$ and $\| a \|_R = \| abbc \|_R = \| abbbb \|_R = 1$; $\| aa \|_R = 3$.

The regular graphs of finite degree (any vertex is linked by a finite number of edges) are the transition graphs of pushdown automata restricted to a regular set of configurations. So the regular graphs of finite degree recognize the context-free languages. By adding ε-transitions, any regular graph (allowing vertices of infinite degree) recognizes a context-free language. Furthermore by identifying vertices linked by ε-edges on the transition graphs of deterministic pushdown automata, we get the deterministic regular graphs [Ca 07].

Proposition 2.1 *The grammars (resp. generating deterministic graphs, deterministic graphs of finite degree) recognize the (resp. deterministic, deterministic and real-time) context-free languages.*

3 Synchronization of grammars

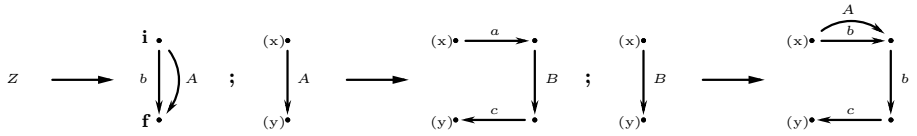
We introduce the synchronization as a binary relation between grammars generating deterministic graphs. To each grammar R, we associate the family $Sync(R)$ of the languages recognized by its synchronized grammars. By applying standard constructions on finite automata to synchronized grammars, we show that $Sync(R)$ is an extension of the regular languages which remains a boolean algebra (cf. Theorem 3.5).

In this section, we restrict to any grammar R generating a deterministic graph. Note that the determinism of R^ω is a first order sentence which is decidable.

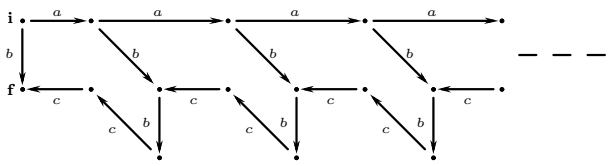
A grammar R *synchronizes* a grammar S and we write $R \rhd S$ or $S \lhd R$ if

$$\forall\, u \in L(S,i) \;\; (\, u \in L(R,i) \;\wedge\; \| u \|_R = \| u \|_S \,)$$

meaning that the label u of any initial path of the graph(s) generated by S is the label of an initial path of the graph generated by R, and these paths end in vertices of the same level. For instance the grammar of the previous section synchronizes the following one:



whose generated graph(s) is represented below by vertices of increasing level (the vertices of a same level are the vertices in a same vertical line).



Note that $\rhd$ is a reflexive and transitive relation which is not antisymmetric; we denote $\bowtie$ the *bi-synchronization* relation:

$$R \bowtie S \quad \text{if} \quad R \rhd S \text{ and } S \rhd R.$$

So $R \bowtie S \iff R \rhd S \wedge L(R,i) = L(S,i)$. Let us give a basic property of $\rhd$.

Lemma 3.1 *$R \rhd S$ and R^ω of finite degree $\implies$ S^ω of finite degree.*

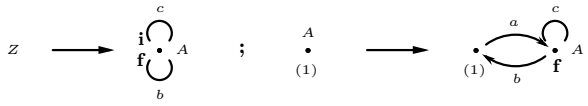
The grammar synchronization does not depend on colour f *i.e.* on final vertices. For instance the language recognized by the previous graph is $\{ a^n b(bcc)^n \mid n \geq 0 \}$ which has no common word with the language recognized by the graph of the previous section. For each grammar R, we associate the following family:

$$Sync(R) := \{ L(R) \cap L(S) \mid R \rhd S \} \text{ of } \textit{synchronized languages} \text{ by } R.$$

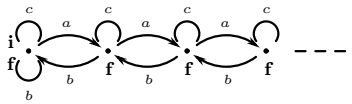
In particular $L(R) \in Sync(R)$

$$Sync(R) = \{ L(S) \mid R \rhd S \} \text{ if any vertex of } R^\omega \text{ is final.}$$

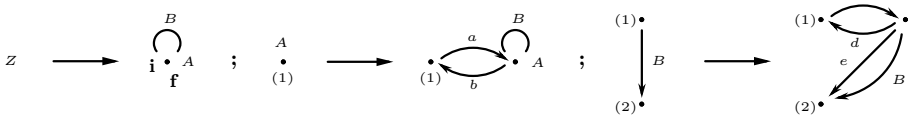
Taking the following grammar R:



generating the following (label) complete graph:



its set $Sync(R)$ is the family of *visibly pushdown languages* for a a pushing letter, b a popping letter and c an internal letter [AM 04]. This family is a boolean algebra containing all the regular languages and closed under concatenation $\cdot$ and under its Kleene iteration $*$. Except for the last two operations, we extend these closure properties to $Sync(R)$ for any grammar R, and we will give a general condition on R such that $Sync(R)$ is also closed under $\cdot$ and $*$. Due to Proposition 2.1 and Lemma 3.1, we have more families of synchronized languages by allowing grammars generating graphs of infinite in-degree. For instance the following grammar R:



has a family $Sync(R) \neq Sync(S)$ for any grammar S such that S^ω is of finite degree.
The closure properties of $Sync(R)$ are just obtained by translating usual constructions on finite automata to synchronized grammars.
Recall that the *synchronization product* $G{\times}H$ of two graphs G and H is the graph

$$G{\times}H := \{\ (s,p) \xrightarrow{a} (t,q) \mid s \xrightarrow[G]{a} t \ \wedge\ p \xrightarrow[H]{a} q\ \}$$
$$\cup\ \{\ i(s,p) \mid is \in G \ \wedge\ ip \in H\ \} \ \cup\ \{\ f(s,p) \mid fs \in G \ \wedge\ fp \in H\ \}.$$

So $L(G{\times}H,i) = L(G,i) \cap L(H,i)$ and $L(G{\times}H) = L(G) \cap L(H)$.
By synchronization product of a grammar R by a finite deterministic automaton, we get that any regular language included in $L(R)$ is a synchronized language of R.

Lemma 3.2 *For any grammar R and any regular language L,*
$$L(R) \cap L \in Sync(R).$$

Proof.
We want to define a grammar synchronized by R and recognizing $L(R) \cap L$.
Let K be a finite deterministic graph (automaton) recognizing L: $L(K) = L$.
We order the vertices (states) of K: $V_K = \{q_1, \ldots, q_n\}$ with $n = |V_K|$.
To each non-terminal $A \in N_R$ of R, we associate a new symbol A' of arity $\varrho(A){\times}n$ except that $Z' = Z$.
To each non-terminal hyperarc $As_1 \ldots s_m$ (with $A \in N_R$ and $m = \varrho(A)$), we associate the following hyperarc:
$$(As_1 \ldots s_m)' := A'(s_1,q_1)\ldots(s_1,q_n)\ldots(s_m,q_1)\ldots(s_m,q_n).$$
The synchronization product of R by K is the following grammar:
$$R{\times}K := \{\ \big(X', [H]{\times}K \cup \{\ Y' \mid Y \in H \ \wedge\ Y(1) \in N_R\ \}\big) \mid (X,H) \in R\ \}.$$
This grammar is synchronized by R.
As $G{\times}K \in (R{\times}K)^\omega$ for any $G \in R^\omega$, the grammar $R{\times}K$ recognizes
$$L(R{\times}K) = L(R) \cap L(K) = L(R) \cap L.$$
□

The synchronization product of a grammar by a deterministic finite automaton can be extended for two grammars synchronized by a given one.
Let S and S' be grammars synchronized by R: $R \rhd S$ and $R \rhd S'$.
To each non-terminal couple $(A,B) \in N_S{\times}N_{S'}$, we associate a new symbol $(A,B)'$ of arity $\varrho(A){\times}\varrho(B)$ except that $(Z,Z)' = Z$.
To each non-terminal hyperarc $As_1 \ldots s_m$ of S ($A \in N_S$ and $m = \varrho(A)$) and each non-terminal hyperarc $Bt_1 \ldots t_n$ of S' ($B \in N_{S'}$ and $n = \varrho(B)$), we associate the hyperarc:
$$(As_1 \ldots s_m\,,\, Bt_1 \ldots t_n)' := (A,B)'(s_1,t_1)\ldots(s_1,t_n)\ldots(s_m,t_1)\ldots(s_m,t_n).$$
The *synchronization product* $S{\times}S'$ of S by S' is the grammar of the rules
$$(X,Y)' \longrightarrow [H]{\times}[K] \cup \{\ (fU,gV)' \mid fU \in H \ \wedge\ f \in N_S \ \wedge\ gV \in K \ \wedge\ g \in N_{S'}\ \}$$
for each $(X,H) \in S$ and $(Y,K) \in S'$.
The synchronization product of grammars synchronized by a grammar R is used

to deduce the closure by intersection of $Sync(R)$.

Lemma 3.3 *For any grammars R, S, S', we have*
$S \lhd R \wedge S' \lhd R \implies S{\times}S' \bowtie S'{\times}S \lhd S$ *and* $(S{\times}S')^\omega = S^\omega{\times}S'^\omega$
$S \lhd R \implies R{\times}S \bowtie S$
$Sync(R)$ is closed under intersection.

Proof.
The grammar $S{\times}S'$ is synchronized by S and S' hence by R and
$$(S{\times}S')^\omega = S^\omega{\times}S'^\omega := \{\, G{\times}G' \mid G \in S^\omega \wedge G' \in S'^\omega \,\}.$$
Thus $L(S{\times}S') = L(S) \cap L(S')$ and $L(S{\times}S', i) = L(S, i) \cap L(S', i)$.
The closure under intersection of $Sync(R)$ results from the equality:
$$\big(L(R) \cap L(S)\big) \cap \big(L(R) \cap L(S')\big) = L(R) \cap L(S{\times}S').$$
Finally $R{\times}S \lhd S$ and $L(R{\times}S, i) = L(S, i)$ thus $R{\times}S \bowtie S$.
□

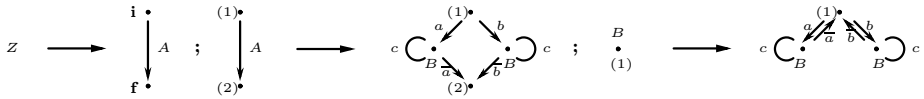
The closure under intersection of $Sync(R)$ implies that
$$Sync(R) = \{\, L(S) \mid R \rhd S \wedge L(S) \subseteq L(R) \,\}.$$
Using Lemma 3.3, we show that we have the same families of synchronized languages by restricting to grammars R generating (deterministic) graphs R^ω which are accessible from i and co-accessible from f; in that case, $L(R, i)$ is the set of prefixes of $L(R)$.
For the closure under union, we define a generalized synchronization product $G{\otimes}H$ of graphs G and H such that $L(G{\otimes}H, i) = L(G, i) \cup L(H, i)$ and $L(G{\otimes}H) = L(G) \cup L(H)$.
As for the synchronization product $S{\times}S'$ of grammars S and S' synchronized by a grammar R, we define the product $S{\otimes}S'$ generating $(S{\otimes}S')^\omega = S^\omega \otimes S'^\omega$.
It follows the closure of $Sync(R)$ under union and complement with respect to $L(R)$. This also permits to decide whether R synchronizes S.

Lemma 3.4 *The synchronization relation $\rhd$ is recursive.*

We summarize previous results.

Theorem 3.5 *For any grammar R, the family $Sync(R)$ of synchronized languages is an effective boolean algebra with respect to $L(R)$, of deterministic context-free languages, containing all the regular languages included in $L(R)$.*

Let us give another family of synchronized languages than the visibly pushdown languages. Taking an internal letter c, two pushing letters a, b and their corresponding popping letters $\overline{a}, \overline{b}$, the following grammar R:



defines by synchronization the family $Sync(R)$ of balanced languages [BB 02]. This family $Sync(R)$ is not closed under $\cdot$ and $*$. By extending standard constructions on finite automata for the closure under $\cdot$ and $*$, we will get families of synchronized languages closed under $\cdot$ and $*$. However the constructions need to extend the synchronization to grammars generating non deterministic graphs.

4 Synchronization of weighted grammars

We generalize the notion of synchronization to grammars, called weighted grammars, generating non-deterministic graphs. A grammar is weighted if in the generated graph two initial paths with the same label end in vertices of same weight. We show that weighted grammars can be in a certain sense determinized (cf. Proposition 4.2) and hence do not allow to capture new boolean algebras. However they allow to use on grammars the standard constructions for concatenation and its iteration (which introduce non-determinism). As a consequence, we provide a simple sufficient condition for the boolean algebras, defined by synchronization of grammars, to be closed under concatenation and iteration (cf. Theorem 4.3).

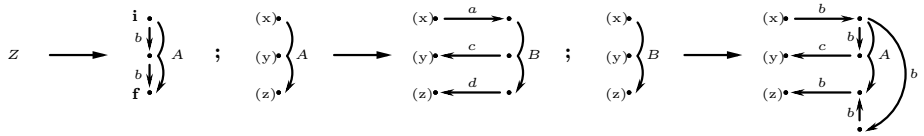
In this section, a deterministic graph grammar R can generate a non deterministic graph $G \in R^{\omega}$. We say that R is a *weighted grammar* if two initial paths with the same label end in vertices of the same level:

$$i \underset{G}{\overset{u}{\Longrightarrow}} s \ \wedge \ i \underset{G}{\overset{u}{\Longrightarrow}} t \quad \Longrightarrow \quad \ell(s) = \ell(t)$$

which allows to define the weight of any $u \in L(R, i)$ as above:

$$\| u \|_R := \ell(s) \quad \text{for} \quad i \underset{G}{\overset{u}{\Longrightarrow}} s.$$

Here is an example of a weighted grammar generating a non deterministic graph:



Any grammar generating a deterministic graph is weighted. Any weighted grammar generates a finite out-degree graph which can be of infinite in-degree. The decidability that a grammar generates a deterministic graph can be extended to the weighted property.

Lemma 4.1 *We can decide whether a grammar is weighted.*

The synchronization is generalized to weighted grammars R and S:

$$R \rhd S \quad \text{if} \quad \forall\, u \in L(S,i),\ u \in L(R,i) \ \wedge\ \|\, u \,\|_R \ =\ \|\, u \,\|_S$$

$$\text{and} \quad Sync_w(R) \ :=\ \{\ L(R) \cap L(S) \mid R \rhd S \ \wedge\ S \text{ weighted } \}\,.$$

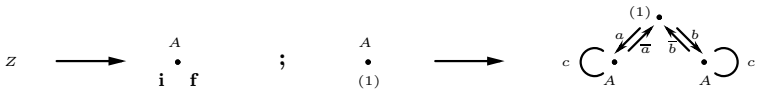
For instance the above grammar is synchronized by the grammar of the previous section. A key property is that any weighted grammar can be determinized.

Proposition 4.2 *Any weighted grammar can be transformed into an equivalent bi-synchronized grammar generating a deterministic graph.*

This implies that for any weighted grammar R, we can construct a grammar S generating a deterministic graph such that $Sync_w(R) = Sync(S)$.
Weighted grammars have been introduced for the closure under $\cdot$ and $*$ of $Sync(R)$ for grammars R generating deterministic graphs.
We say that R is a *cyclic grammar* if R^ω is deterministic and its initial vertex is the unique final vertex. In that case $L(R)$ is closed under concatenation.

Theorem 4.3 *For any cyclic grammar R, the family $Sync(R)$ is closed under $\cdot$ and under $*$.*

For instance taking the following cyclic grammar R:



the family $Sync(R)$ is the closure under concatenation and under iteration of the concatenation of the balanced languages [BB 02].

5 Synchronization of pushdown automata

The synchronization of height-deterministic pushdown automata [NS 07] and the synchronization with a transducer [Ca 06] define language families synchronized by grammars.

We begin with the last approach of [NS 07]. A (real-time) *pushdown automaton* in a weak form S over an alphabet T of *terminals* is a finite set of rules of the form:

$$Ap \xrightarrow{a} q \quad ; \quad Ap \xrightarrow{a} Aq \quad ; \quad Ap \xrightarrow{a} ABq$$

with $A, B \in P,\ p, q \in Q,\ a \in T$, where P, Q are disjoint alphabets of respectively *stack letters* and *states*. We associate to S a subset $F \subseteq Q$ of *final states*, and an *initial configuration* $c = \perp r$ for $r \in Q$ and $\perp \in P$ which cannot be popped ($\perp p \xrightarrow{a} q$ is not a rule of S). The *transition graph* $\mathrm{Tr}(S)$ of S is the set of its transitions

$$\{\ wu \xrightarrow{a} wv \mid u \xrightarrow[S]{a} v \ \wedge\ w \in P^*\ \} \ \cup\ \{i\,c\} \ \cup\ \{\ fu \mid u \in P^*F\ \}$$

restricted to the vertices accessible from c. We denote $L(S) := L(Tr(S))$ the *language recognized* by S, and we say that S is *complete* if $L(Tr(S), i) = T^*$.
A complete pushdown automaton S is *height-deterministic* [NS 07] if

$$c \underset{Tr(S)}{\overset{u}{\Longrightarrow}} xp \ \wedge \ c \underset{Tr(S)}{\overset{u}{\Longrightarrow}} yq \quad \Longrightarrow \quad |x| = |y|$$

meaning that two initial paths with the same label end in vertices of same length. Finally two height-deterministic pushdown automata S and S' are synchronized if

$$c \underset{Tr(S)}{\overset{u}{\Longrightarrow}} xp \ \wedge \ c' \underset{Tr(S')}{\overset{u}{\Longrightarrow}} yq \quad \Longrightarrow \quad |x| = |y|$$

and the family of languages synchronized by S is

$$Sync(S) \ := \ \{ \ L(S') \mid S' \text{ synchronized by } S \ \}.$$

As S is complete and $Sync(S)$ does not depend on the final states of S, $T^* \in Sync(S)$.
Any family $Sync(S)$ can be obtained by synchronization with a grammar.

Proposition 5.1 *We can transform any height-deterministic pushdown automaton S into a grammar R with R^ω deterministic of finite degree and $Sync(R) = Sync(S)$.*

Any family $Sync(S)$ contains T^*, hence cannot be the set of balanced languages. However and redefining $Sync(S) := \{ \ L(S) \cap L(S') \mid S' \lhd S \ \}$, Proposition 5.1 remains true and its converse must be studied.
The synchronization of pushdown automata, and more generally grammars, by a transducer [Ca 06] can also be captured using synchronization by a *linear grammar*: each right hand side has at most one non-terminal hyperarc. Taking the visibly pushdown languages, the converse is false.
In conclusion, the synchronization by grammars strictly generalizes the known synchronization notions of pushdown automata.

Many thanks to Arnaud Carayol and Antoine Meyer for their remarks and comments on this paper.

References

[AM 04] R. Alur and P. Madhusudan *Visibly pushdown languages*, 36^{th} STOC, ACM Proceedings, L. Babai (Ed.), 202–211 (2004).
[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 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 06] D. Caucal *Synchronization of pushdown automata*, 10^{th} DLT, LNCS 4036, O. Ibarra, Z. Dang (Eds.), 120-132 (2006).
[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).