HAL open science

Synchronization of Pushdown Automata

Didier Caucal

▶ To cite this version:

Didier Caucal. Synchronization of Pushdown Automata. DLT'06, Jun 2006, Santa Barbara, CA, United States. pp.120-132, 10.1007/11779148_12 . hal-00620158

HAL Id: hal-00620158

https://hal.science/hal-00620158v1

Submitted on 30 Sep 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

SYNCHRONIZATION OF PUSHDOWN AUTOMATA

Didier Caucal

caucal@irisa.fr

IRISA–CNRS, Campus de Beaulieu, 35042 Rennes, France

Abstract. We introduce the synchronization of a pushdown automaton by a sequential transducer associating an integer to each input word. The visibly pushdown automata are the automata synchronized by an one state transducer whose output labels are $-1, 0, 1$. For each transducer, we can decide whether a pushdown automaton is synchronized. The pushdown automata synchronized by a given transducer accept languages which form an effective boolean algebra containing the regular languages and included in the deterministic real-time context-free languages.

1 Introduction

It is well-known that the context-free languages are not closed under intersection and complementation, and that the deterministic context-free languages are not closed under intersection and union. Alur and Madhusudan have shown that the languages accepted by the visibly pushdown automata form a boolean algebra included in the deterministic real-time context-free languages [AM 04]. The notion of visibly pushdown automaton is based on the synchronization between the input symbols and the actions performed on the stack: this enforces that the variation of the stack height is entirely characterized by the input word.

It appears that the closure results for the languages accepted by the visibly pushdown automata are based on a geometrical property of their graphs with regard to the stack height. This geometrical property which holds for every pushdown graph (not only visibly) was discovered by Muller and Schupp [MS 85]. A simple adaptation of their result shows that the graph of every pushdown automaton is regularly generated by increasing stack height [Ca 95]. This regularity is described by a finite deterministic graph grammar which in n steps of parallel rewritings, generates the graph restricted to the configurations with stack height at most n.

In this article, we generalize the notion of synchronization to abstract from the

stack height. Towards this goal, we introduce a sequential transducer associating an integer to each input word. Provided that this transducer defines a weight for the vertices of the pushdown graph, we show that we can decide whether the graph can be generated regularly with regard to that weight. This is the notion of synchronization by a transducer. For any fixed transducer, the languages accepted by the pushdown automata synchronized by this transducer, are deterministic real-time context-free languages and form an effective boolean algebra containing the regular languages.

2 Graphs and finite automata

By allowing labels not only on arcs but also on vertices, we define graphs as a simple extension of automata: the vertex labelling is not restricted to indicate initial and final vertices but it also permits to add information on vertices (e.g. the vertices accessible from a given colour, more generally the vertices defined by a μ-formula, ...). First we give some notations.
Let $\mathbb{N}$ be the set of non-negative integers and $\mathbb{Z}$ be the set of integers. For any set E, we denote $|E|$ its cardinality. For every $n \geq 0$, E^n is the set of n tuples of elements of E, and $E^* = \bigcup_{n\geq 0} E^n$ is the free monoid generated by E for the *concatenation*: $(e_1, \ldots, e_n)\cdot(e'_1, \ldots, e'_n) = (e_1, \ldots, e_n, e'_1, \ldots, e'_n)$. 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 represented by the juxtaposition of its letters: $u = u(1)\ldots u(|u|)$. The word of length 0 is the *empty word* ε. We denote $|u|_P := |\{\ 1 \leq i \leq |u| \mid u(i) \in P\ \}|$ the *number of occurrences* of $P \subseteq E$ in $u \in E^*$. For any binary relation $R \subseteq E{\times}F$ from E into a set F, we write also $e\,R\,f$ for $(e, f) \in R$, and we denote $Dom(R) := \{\ e \mid \exists\ f,\ e\,R\,f\ \}$ the *domain* of R, and $Im(R) := \{\ f \mid \exists\ e,\ e\,R\,f\ \}$ the *image* (or the range) of R.

Now we present our notion of graph which generalizes the notion of automaton. Let L and C be disjoint countable sets of symbols for respectively labelling arcs and labelling vertices. Here a graph is simple, oriented, arc labelled in a finite subset of L and vertex labelled in a finite subset of C. Precisely, a *graph* G is a subset of $V{\times}L{\times}V \cup C{\times}V$ where V is an arbitrary set such that its *vertex* set

$$V_G \ := \ \{\ p \mid \exists\ a, q,\ (p, a, q) \in G\ \vee\ (q, a, p) \in G\ \}\ \cup\ \{\ p \mid \exists\ c,\ (c, p) \in G\ \}$$

is finite or countable, with its *vertex label* set or *colour* set

$$C_G \ := \ \{\ c \in C \mid \exists\ p,\ (c, p) \in G\ \} \quad \text{is finite,}$$

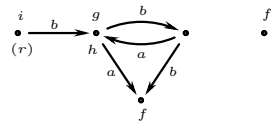
and its *arc label* set or *label* set

$$L_G \ := \ \{\ a \in L \mid \exists\ p, q,\ (p, a, q) \in G\ \} \quad \text{is finite.}$$

Any (p, a, q) of G is a *labelled arc* of *source* p, of *goal* q, with label a, and is identified with the labelled transition $p \xrightarrow[G]{a} q$ or directly $p \xrightarrow{a} q$ if G is understood. Any (c, p) of G is a vertex p labelled by c and is also written $c\,p$ if G is understood. We denote $V_{G,i} := \{\ p \mid i\,p\ \in\ G\ \}$ the set of vertices of G labelled by the colour $i \in C$.
A graph is *deterministic* if distinct arcs with the same source have distinct labels:

$$r \xrightarrow{a} p \;\wedge\; r \xrightarrow{a} q \quad \Longrightarrow \quad p = q\,.$$

Note that a graph G is finite if and only if it has a finite vertex set V_G. For instance $\{r \xrightarrow{b} p\,,\, p \xrightarrow{a} s\,,\, p \xrightarrow{b} q\,,\, q \xrightarrow{a} p\,,\, q \xrightarrow{b} s\,,\, i\,r\,,\, g\,p\,,\, h\,p\,,\, f\,s\,,\, f\,t\}$ is a finite graph of vertices p, q, r, s, t, of colours i, f, g, h, and of (arc) labels a, b. It is represented below.



Note that a vertex r is depicted by a dot named by (r) where parentheses are used to differentiate a vertex name with a vertex label (a colour).
For any $p \in V_G$, $d^+(p) := |\{\ (a,q) \mid p \xrightarrow{a} q\ \}|$ and $d^-(p) := |\{\ (q,a) \mid q \xrightarrow{a} p\ \}|$ are respectively the *out-degree* and the *in-degree* of p; $d(p) := d^+(p) + d^-(p)$ is the *degree* of p and $d_G := sup\{\ d(p) \mid p \in V_G\ \}$ is the *degree* of G.
A graph is of *finite degree* (or *locally finite*) if $d(p) < \omega$ for any vertex p; a graph G is of *bounded degree* (or *locally bounded*) if $d_G < \omega$.
A graph G without vertex label *i.e.* $C_G = \emptyset$ is called an *uncoloured graph*.
The *restriction* (or the *induced subgraph*) of a graph G to a vertex subset P is

$$G_{|P} \;:=\; \{\ p \xrightarrow[G]{a} q \mid p, q \in P\ \} \;\cup\; \{\ c\,p \in G \mid p \in P\ \}.$$

Any tuple $(p_0, a_1, p_1, \ldots, a_n, p_n)$ such that $n \geq 0$ and $p_0 \xrightarrow{a_1} p_1 \ldots p_{n-1} \xrightarrow{a_n} p_n$, denoted also by the word $p_0 a_1 p_1 \ldots a_n p_n$ (which have a sense if $V_G \cap L_G^* = \emptyset$), is a *path* from p_0 to p_n labelled by $u = a_1 \ldots a_n$, and we write $p_0 \xRightarrow[G]{u} p_n$ or directly $p_0 \xRightarrow{u} p_n$ if G is understood; for $n = 0$, the path $p_0 \xRightarrow[G]{\varepsilon} p_0$ is reduced to $p_0 \in V_G$. For any $U \subseteq L^*$, we write $p \xRightarrow{U} q$ if $p \xRightarrow{u} q$ for some $u \in U$. We also write $p \xRightarrow[G]{}{}^* q$ if $p \xRightarrow[G]{L^*} q$.
We say that a vertex r is a *root* of G if every vertex p is accessible from r: $r \Longrightarrow^* p$. The *accessible subgraph* $G_{/p} := G_{|\{\,q \mid p \Longrightarrow^* q\,\}}$ of a graph G from a vertex p is the restriction of G to the vertices accessible from p.
Given a graph G and vertex sets $P, Q \subseteq V_G$, we denote $L(G, P, Q)$ the language of path labels from vertices in P to vertices in Q:

$$L(G, P, Q) \;:=\; \{\ u \mid \exists\, p \in P\ \exists\, q \in Q\ \ p \xRightarrow[G]{u} q\ \}\,.$$

Given colours $i, f \in C$, we define $L(G, i, f) := L(G, V_{G,i}, V_{G,f})$ the path labels from the set $V_{G,i}$ of vertices labelled by i to the set $V_{G,f}$ of vertices labelled by f.
For instance taking the previous graph, its path labels from i to f is $b(ba)^*(a{+}bb)$.
So a finite graph G with two colours i and f is a *finite automaton* recognizing the language $L(G, i, f)$. The family

$$Rat(T^*) \;:=\; \{\ L(G, i, f) \mid |G| < \omega \;\wedge\; i, f \in C\ \}$$

of languages over T recognized by the finite automata coincides with the family of *rational languages* (or *regular languages*). So the finite graphs describe the structures of the rational languages and permit to derive properties on these languages. For the context-free languages which are the languages recognized by

the pushdown automata, their graphs are generated by the deterministic graph grammars which are also really powerful to get properties on these languages. Our purpose is to use the deterministic graph grammars in order to describe geometrically the notion of visibly pushdown automata and to get in this way a natural generalization.

3 Graph grammars and pushdown automata

A pushdown automaton is a particular case of a word rewriting system where the rules are only applied by prefix. By restriction to rational vertex sets, the pushdown automata and the word rewriting systems define the same (prefix) graphs which are the graphs of bounded degree and regular in the sense that they can be generated by a deterministic grammar [Ca 90]. We extend this result to the pushdown automata which are in a weak form used by the visibly pushdown automata (cf. Theorem 3.1). We recall that the graphs of the pushdown automata are regular by increasing length (cf. Proposition 3.2).

We fix an alphabet T of *terminals*. Recall that a *labelled word rewriting system* R is a finite subset of $N^* \times T \times N^*$ for some alphabet N of *non-terminals* *i.e.* is a finite uncoloured graph of (arc) labels in T and whose vertices are words over N. The graph:

$$G{\cdot}P \ := \ \{ \ uw \xrightarrow{a} vw \mid u \xrightarrow[G]{a} v \ \wedge \ w \in P \ \}$$

is the *right concatenation* of any graph $G \subseteq N^* \times T \times N^*$ by any language $P \subseteq N^*$. Rewritings of a system are generally defined as applications of rewriting rules in every context. On the contrary, we are here concerned with prefix rewriting [Bü 64]. The *prefix transition graph* of R is the uncoloured graph $R{\cdot}N^*$ which is of bounded degree and has a finite number of non isomorphic connected components.

A subclass of labelled word rewriting systems is the standard model of real-time pushdown automata. A *pushdown automaton* R (without ε-rule) is a finite set of rules of the form:

$$pA \xrightarrow{a} qU \quad \text{with} \quad p, q \in Q, \ A \in P, \ U \in P^*, \ a \in T$$

where P and Q are disjoint alphabets of respectively *pushdown letters* and *states*. The *transition graph* of R is $R{\cdot}P^* \ = \ \{ \ pAV \xrightarrow{a} qUV \mid pA \xrightarrow[R]{a} qU \ \wedge \ V \in P^* \ \}$ the restriction of the prefix transition graph $R{\cdot}(P \cup Q)^*$ of R to the rational set QP^* of *configurations*.

A strong way to normalize the rules of pushdown automata is given by a *weak pushdown automaton* R which is a finite set of rules of the following form:

$$p \xrightarrow{a} q \quad \text{or} \quad p \xrightarrow{a} qA \quad \text{or} \quad pA \xrightarrow{a} q \quad \text{with} \quad p, q \in Q, \ A \in P, \ a \in T.$$

Its transition graph $R{\cdot}P^*$ is isomorphic to $S{\cdot}P^*\bot$ where $\bot$ is a new pushdown letter (the bottom of the stack) and S is the following pushdown automaton:

$$\begin{aligned} S = \{ \ & pA \xrightarrow{a} qA \mid p \xrightarrow[R]{a} q \ \wedge \ A \in P \cup \{\bot\} \ \} \\ & \cup \{ \ pB \xrightarrow{a} qAB \mid p \xrightarrow[R]{a} qA \ \wedge \ B \in P \cup \{\bot\} \ \} \ \cup \ \{ \ pA \xrightarrow{a} q \mid pA \xrightarrow[R]{a} q \ \}. \end{aligned}$$

The labelled word rewriting systems and the weak pushdown automata define the same prefix transition graphs, hence also for the pushdown automata which are intermediate devices.

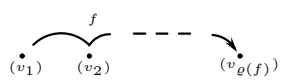
Theorem 3.1 *The transition graphs of weak pushdown automata,*
the transition graphs of pushdown automata,
the prefix transition graphs of labelled word rewriting systems,
have up to isomorphism the same
accessible subgraphs: the rooted regular graphs of bounded degree,
connected components: the connected regular graphs of bounded degree,
rational restrictions: the regular graphs of bounded degree.

This theorem has been first established in [Ca 90] and completed in [Ca 95] but without considering the weak pushdown automata.
It remains to recall what is a regular graph and more exactly to reintroduce the notion of a deterministic graph grammar to generate a graph. Such a generation needs to use non-terminal arcs linking several vertices and called hyperarcs.

Let F be a set of symbols called *functions*, graded by a mapping $\varrho : F \longrightarrow \mathbb{N}$ associating to each function f its *arity* $\varrho(f)$, and such that

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

with $F_1 = C$ and $F_2 = L$.
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 $V_G := \{\ p \in V \mid FV^*pV^* \cap G \neq \emptyset\ \}$ is finite or countable,
its *label* set $F_G := \{\ f \in F \mid fV^* \cap G \neq \emptyset\ \}$ is finite.
Any $fv_1 \ldots v_{\varrho(f)} \in G$ 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 $v_1, \ldots, v_{\varrho(f)}$:



The transformation of a hypergraph G by a function h from V_G into any set V is the graph $h(G) := \{\ fh(v_1)\ldots h(v_{\varrho(f)}) \mid fv_1\ldots v_{\varrho(f)} \in G\ \}$. Note that the graphs are the hypergraphs whose any label is of arity 1 or 2 : any arc $p \xrightarrow{a} q$ corresponds to the hyperarc apq.
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. The labels of the left hand sides form the set N_R of *non-terminals* of R :

$$N_R := \{\ X(1) \mid X \in Dom(R)\ \},$$

and the labels of R which are not non-terminals form the set T_R of *terminals* :

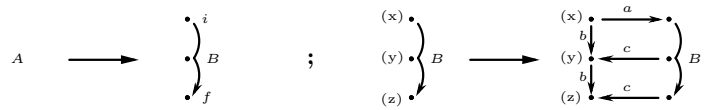
$$T_R := \{\ X(1) \notin N_R \mid \exists\ P \in Im(R),\ X \in P\ \}.$$

Any graph grammar R is used to generate graphs of arc labels in T hence we assume that $T_R \subset T \cup C$. We will use capital letters for the non-terminals

and small letters for the terminals. Starting from any non-terminal hyperarc, we want to generate by a graph grammar a unique graph up to isomorphism. So we restrict any graph grammar to be *deterministic*: there is only one rule per non-terminal. For instance taking $A \in F_0$, $B \in F_3$, $a, b, c \in T$ and $i, f \in C$, the following two rules:

$$A \longrightarrow \{ip\,,\, fr\,,\, Bpqr\} \;\;;\;\; Bxyz \longrightarrow \{axp\,,\, bxy\,,\, cqy\,,\, byz\,,\, crz\,,\, Bpqr\}$$

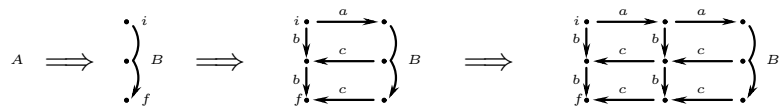
constitute a deterministic graph grammar which is represented below:



For any (deterministic graph) grammar R, the *rewriting* $\underset{R}{\longrightarrow}$ is the binary relation between hypergraphs defined by $M \underset{R}{\longrightarrow} N$ 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 to replace X by H in M:

$$N \;=\; (M - X) \;+\; h(H)$$

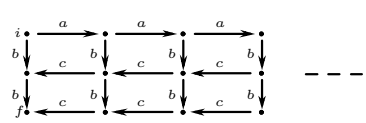
for some function h mapping x_i to s_i, and the other vertices of H injectively to vertices outside of M; this rewriting is denoted by $M \underset{R,\,X}{\longrightarrow} N$. The rewriting $\underset{R,\,X}{\longrightarrow}$ of a hyperarc X is extended in an obvious way to the rewriting $\underset{R,\,E}{\longrightarrow}$ of any set E of non-terminal hyperarcs. A *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.
For instance, the first three steps of the parallel derivation from the hypergraph $\{A\}$ according to the above grammar are depicted in the figure below.



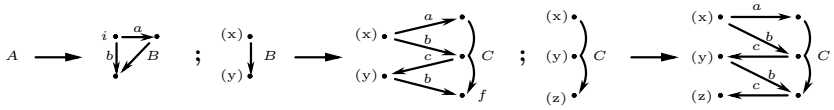
Let $[H] := H \cap (CV_H \cup TV_HV_H)$ be the set of terminal arcs and of coloured vertices of any hypergraph H.
A *regular graph*, also called a *hyperedge replacement equational graph* [Co 90], is a graph G *generated* by a hypergraph grammar R from a non-terminal hyperarc X. More formally, G is isomorphic to a graph in the following set $R^\omega(X)$ of isomorphic graphs:

$$R^\omega(X) := \{\, \textstyle\bigcup_{n\geq 0}[H_n] \mid H_0 = X \;\wedge\; \forall\, n \geq 0,\; H_n \underset{R}{\Longrightarrow} H_{n+1} \,\}\,.$$

For instance by continuing infinitely the previous derivation, we get the infinite graph:



In particular any regular graph of finite degree is of bounded degree.
The regular graphs trace the context-free languages: for any regular graph G (not necessarily bounded) and for any colours $i, f \in C$, $L(G, i, f)$ is a context-free language and by Theorem 3.1, the converse is true. Graph grammars are suitable to deduce the pumping lemma, or to prove the Parikh lemma. Here we will use graph grammars to describe geometrically the notion of visibly pushdown automaton and to extend it.
A regular graph can be generated by several grammars. For instance instead of generating the previous graph by 'vertical slides', we can generate it by 'diagonal slides' using the following grammar:



We specify the regularity of a graph G according to a mapping g from V_G into $\mathbb{N}$. Precisely for every $n \geq 0$, we define the graph $G_{g,n}$ of the first n *levels* of G according to g by

$$G_{g,n} \ := \ \{ \ p \xrightarrow[G]{a} q \mid g(p) < n \ \vee \ g(q) < n \ \} \ \cup \ \{ \ c\,p \in G \mid g(p) < n \ \} \, .$$

We say that a graph G is *regular by* g if there exists a grammar R and a non-terminal hyperarc I such that for any parallel derivation $I \underset{R}{\Longrightarrow}^n H$, the set of terminal arcs of H is $[H] = G_{g,n}$ and its vertex set of its non-terminal hyperarcs is $V_{H-[H]}$ which is included in

$$\{ \ p \in V_G \mid g(p) = n \ \} \ \cup \ \{ \ p \in V_G \mid g(p) > n \ \wedge \ \exists \, q \, (p \text{ — } q \ \wedge \ g(q) < n) \ \}$$

with the notation $p \text{ — } q$ for $\exists \, a, \ p \xrightarrow{a} q \ \vee \ q \xrightarrow{a} p$.
So any graph regular by some mapping is of bounded degree.
We consider the regularity of the transition graph $R{\cdot}P^*$ of any pushdown automaton R according to the *stack height* $|U|$ of any configuration pU where $p \in Q$ is a state and $U \in P^*$ is a pushdown word. When R is weak then $R{\cdot}P^*$ is regular by stack height with the grammar reduced to this unique rule:

$$Zq_1 \ldots q_n \ \longrightarrow \ R \cup \{ \ Z(q_1 A) \ldots (q_n A) \mid A \in P \ \} \quad \text{for} \quad \{q_1, \ldots, q_n\} = Q.$$

By synchronisation product of this rule with any finite automaton, we deduce that any rational restriction of the transition graph of any weak pushdown automaton is regular by stack height (or by length). This result is extended to any pushdown automaton.

Proposition 3.2 *The rational restrictions of the prefix transition graphs of labelled word rewriting systems are regular by length.*

This proposition has been established for any morphism [Ca 95].
We can now present a geometrical description of the visibly pushdown automata, and its extension by synchronization with a sequential transducer with integer output.

4 Visibly pushdown automata

We present the visibly pushdown automata defined in [AM 04] with the main result (cf. Theorem 4.1), and we consider the regularity of their transition graphs by stack height.

The visibly pushdown automata are given according to a splitting of the alphabet T of terminals into three disjoint alphabets T_{-1}, T_0, T_1 to indicate respectively the letters allowed to pop the topmost stack symbol, to unchange the stack, and to push a symbol on the stack. A *visibly pushdown automaton* R is a finite set of rules of the following form:

$$pA \xrightarrow{a} q \quad \text{or} \quad p \xrightarrow{b} q \quad \text{or} \quad p \xrightarrow{c} qA \quad \text{or} \quad p\bot \xrightarrow{a} q\bot$$

with $p, q \in Q,\ A \in P,\ a \in T_{-1}, b \in T_0, c \in T_1$, where $P, Q, \{\bot\}$ are disjoint alphabets of respectively *pushdown letters*, of *states* and of the bottom of the stack. The transition graph of R is $R{\cdot}P^*\bot$ and the language $L(R{\cdot}P^*\bot\,,\, I\bot\,,\, FP^*\bot)$ recognized from a set $I \subseteq Q$ of *initial states* to a set $F \subseteq Q$ of *final states* is a *visibly pushdown language*.
For instance taking $a \in T_1$, $c \in T_0$, $b \in T_{-1}$, the language $\{\ a^n c b^n \mid n \geq 0\ \}$ is a visibly pushdown language and the Lukasiewicz language *i.e.* the language $L(G, A)$ generated by the context-free grammar $G = \{A \longrightarrow aAA\,,\, A \longrightarrow b\}$, is also a visibly pushdown language. But the language $\{\ a^n b a^n \mid n \geq 0\ \}$ and the language $L(G, A)$ generated by the grammar $G = \{A \longrightarrow aAAA\,,\, A \longrightarrow b\}$ are not visibly pushdown languages for any partition of T in T_{-1}, T_0 and T_1. So the visibly pushdown languages are not preserved in general by morphism and inverse morphism.
Any rational language over T is a visibly pushdown language according to any partition $T = T_{-1} \cup T_0 \cup T_1$: for any finite T-graph H and any $I, F \subseteq V_H$, the rational language $L(H, I, F) = L(R{\cdot}P^*\bot\,,\, I\bot\,,\, FP^*\bot)$ for $P = \{A\}$ reduced to a unique pushdown letter A and for the following visibly pushdown automaton:

$$R = \{\ p \xrightarrow{a} qA \mid p \xrightarrow[H]{a} q\ \wedge\ a \in T_1\ \} \cup \{\ p \xrightarrow{a} q \mid p \xrightarrow[H]{a} q\ \wedge\ a \in T_0\ \}$$
$$\cup\ \{\ pA \xrightarrow{a} q \mid p \xrightarrow[H]{a} q\ \wedge\ a \in T_{-1}\ \} \cup \{\ p\bot \xrightarrow{a} q\bot \mid p \xrightarrow[H]{a} q\ \wedge\ a \in T_{-1}\ \}.$$

The family of visibly pushdown languages is an extension of the regular languages with same basic closure properties.

Theorem 4.1 [AM 04] *For any partition of the input letters, the class of visibly pushdown languages is a subfamily of deterministic real-time context-free languages, and is an effective boolean algebra closed by concatenation and its transitive closure.*

In particular the universality problem and the inclusion problem are decidable for the visibly pushdown languages. For any visibly pushdown automaton R and by discarding the rules $p\bot \xrightarrow{a} q\bot$ (for $a \in T_{-1}$), note that

$$pU \overset{u}{\underset{R\cdot P^*}{\Longrightarrow}} qV \quad \Longrightarrow \quad |V| - |U| \ = \ |u|_{T_1} - |u|_{T_{-1}}$$

which implies the following first key point:

$$|u|_{T_1} - |u|_{T_{-1}} \;=\; |v|_{T_1} - |v|_{T_{-1}} \;=\; |U| \quad \text{for any } u, v \in L(R{\cdot}P^*, I, pU).$$

A second key point is given by Proposition 3.2 : any rational restriction of $R{\cdot}P^*$ can be generated by a graph grammar S by stack height. We will see that these two key points are sufficient to establish Theorem 4.1 (without the closure by concatenation and its transitive closure). These two key points indicate that the weak form of a visibly pushdown automaton is inessential. We need that the transition graph G restricted to the configurations accessible from a given set I, satisfies the property that for every vertex s, any label u of a path from I to s has the same value $|u|_{T_1} - |u|_{T_{-1}}$ (first key point) called the weight of s, and that G is regular by weight (second key point). Note that the weight of a vertex can be negative which allows to discard the rules of the form $p\bot \xrightarrow{a} q\bot$. Finally we will generalize the visibility defined by the partition $T = T_{-1} \cup T_0 \cup T_1$ to any sequential transducer from T^* into $\mathbb{Z}$.

5 Synchronized pushdown automata

The synchronization of pushdown automata over T is done according to a sequential transducer A from T^* into $\mathbb{Z}$. The synchronization by A is defined for the regular graphs of bounded degree which are by Theorem 3.1 the rational restrictions of the transition graphs of pushdown automata. It is decidable whether a regular graph is synchronized by A (cf. Proposition 5.4), and the traces of the graphs synchronized by A form an effective boolean algebra of deterministic real-time context-free languages containing the rational languages (cf. Theorem 5.8).

We fix a colour $i \in C$ to indicate initial vertices.
A sequential *transducer* (or generalized sequential machine) from the free monoid T^* into the additive monoid $\mathbb{Z}$ is a finite graph A of label set $L_A \subset T{\times}\mathbb{Z}$ and of colour set $C_A = \{i\}$ such that A is input deterministic:

$$p \xrightarrow[A]{(a,x)} q \;\wedge\; p \xrightarrow[A]{(a,y)} r \quad\Longrightarrow\quad x = y \;\wedge\; q = r$$

$$i\,p\,,\, i\,q \in A \quad\Longrightarrow\quad p = q \qquad \text{(a unique state is coloured by } i\text{)}.$$

A (sequential) transducer A realizes the transduction

$$L(A, i, V_A) \;=\; \{\; (u, m) \mid \exists\, s, t,\; i\,s \in A \;\wedge\; s \overset{(u,m)}{\underset{A}{\Longrightarrow}} t \;\}$$

of the label set of the paths from the vertex coloured by i to any vertex, for the operation

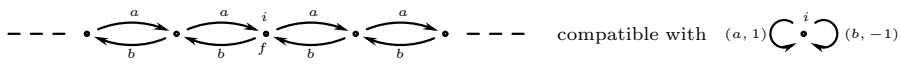
$$(u, m).(v, n) \;:=\; (uv, m + n) \quad \text{for every } \; u, v \in T^* \; \text{ and } \; m, n \in \mathbb{Z}.$$

For instance taking a unique state r, the transducer $\{i\,r\,,\, r \xrightarrow{(a,1)} r\,,\, r \xrightarrow{(b,-1)} r\}$ represented in the next figure, realizes $\{\; (u\,,\, |u|_a - |u|_b) \mid u \in \{a, b\}^* \;\}$.
For any (sequential) transducer A, we denote $L(A){:=}Dom(L(A, i, V_A))$ its first projection, and we say that A is *complete* if $L(A) = T^*$. For any word $u \in L(A)$, there is a unique integer $\| u \|_A$ called the *weight* of u in A such that $(u, \| u \|_A) \in L(A, i, V_A)$.
A transducer A is *visible* if it has a unique state, is complete and the value of any arc can be only $-1, 0, 1$: $|V_A| = 1$, $L_A \subseteq T{\times}\{-1, 0, 1\}$ and $Dom(L_A) = T$;

in that case for any $i \in \{-1, 0, 1\}$, $T_i = \{\ a \mid (a, i) \in L_A\ \}$.
We say that a graph G is *compatible with* a transducer A if for any vertex s of G, there is a path from (a vertex coloured by) i to s and the labels of the paths from i to s are in $L(A)$ and have the same weight:

$$\emptyset \neq L(G, i, s) \subseteq L(A) \quad \wedge \quad u, v \in L(G, i, s) \quad \Longrightarrow \quad \| u \|_A = \| v \|_A\,;$$

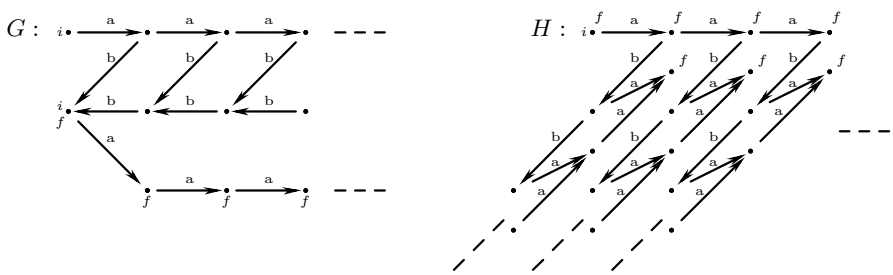
in that case we denote $\| s \|_A := \| u \|_A$ for any $u \in L(G, i, s)$. In the next figure,



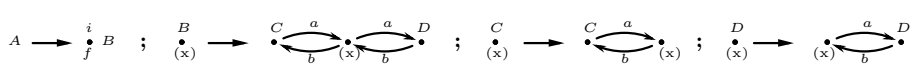
we have a graph G compatible with a visible transducer for $T = \{a, b\}$; note that $L(G, i, f) = \{\ u \in T^* \mid |u|_a = |u|_b\ \}$ is not a visibly pushdown language.
For G compatible with A and $H \subseteq G$, H is compatible with A. Let us give another fact.

Lemma 5.1 *For any regular graph G and any transducer A,*
$G^A := \{\ s \xrightarrow{(a,x)} t \mid s \xrightarrow[G]{a} t \wedge \exists\, p, q,\ p \xrightarrow[A]{(a,x)} q \wedge i \overset{Dom(L(A,i,p))}{\underset{G}{\Longrightarrow}} s\ \} \cup \big(G \cap CV_G\big)$
is a regular graph, and we can decide whether G is compatible with A.

Here are represented by increasing weight two regular graphs of finite degree which are compatible with the previous visible transducer.



Their languages $L(G, i, f) = \{\ a^n b^n \mid n \geq 0\ \}a^*$ and $L(H, i, f) = a^*\{\ b^n a^n \mid n \geq 0\ \}$ give by intersection the language $\{\ a^n b^n a^n \mid n \geq 0\ \}$ which is not context-free, hence is not the language between colours of a regular graph. We now discard the graph H because we cannot generate it by increasing weight: we would need non-terminal hyperarcs having an infinite number of vertices.
We say that a graph is *synchronized* by a transducer A if it is compatible with A and regular by the absolute value of the weight $\|\ \ \|_A$.
The graph above Lemma 5.1 is generated by increasing weight with the following grammar:



Let us give a graph synchronized by A with the same path labels.

Lemma 5.2 *For any transducer A, the following graph:*
$\overrightarrow{A} := \{ (p,n) \xrightarrow{a} (q,n+x) \mid p \xrightarrow[A]{(a,x)} q \ \wedge \ n \in \mathbb{Z} \} \cup \{ i\,(p,0) \mid i\,p \in A \}$
is synchronized by A and $L(G,i,V_G) = L(A)$.

Let us give a simple characterization of the regular graphs which are synchronized. We say that any graph G compatible with a transducer A is *finitely compatible* with A if for every integer $n \in \mathbb{Z}$, the vertex set $\{ s \in V_G \mid \| s \|_A = n \}$ is finite. By definition, any synchronized graph by A is regular and finitely compatible with A; the converse is true.

Proposition 5.3 *For any transducer A,*
G is synchronized by A $\iff$ G is regular and finitely compatible with A.

This permits to extend the decidability of Lemma 5.1 to the synchronization problem.

Proposition 5.4 *For any transducer A, we can decide whether a regular graph G is synchronized by A, and in the affirmative, we can construct a graph grammar generating G by increasing weight $\| \ \|_A$.*

In particular we can decide whether a regular graph is visibly synchronized (we have only a finite number of visible transducers). We fix another colour $f \in C$ to indicate final vertices. The visibly pushdown languages are extended to any transducer A: a *synchronized language* by A is $L(G,i,f)$ for some graph G synchronized by A. Let us give basic examples of synchronized languages.

Example 5.5 The languages synchronized by a transducer A such that $L_A \subseteq T \times \{0\}$ are all the rational languages included in $L(A)$.

Example 5.6 Taking $m \geq 0$, the language $L_m := L(G,X)$ generated by the context-free grammar $G = \{X \longrightarrow aX^m\,,\ X \longrightarrow b\}$ is synchronized by the transducer $A = \{\, i\,p\,,\ p \xrightarrow{(a,m-1)} p\,,\ p \xrightarrow{(b,-1)} p \,\}$. This transducer has a unique state, and it is visible for $m = 0$ ($L_0 = \{a,b\}$), for $m = 1$ ($L_1 = a^*b$) and for $m = 2$ (L_2 is the Lukasiewicz language). For $m > 2$, L_m is not a visibly pushdown language. The language $\{ u \in \{a,b\}^* \mid |u|_b = (m-1)|u|_a \}$ is also synchronized by A.
More generally for $m,n \geq 0$, $L_{m,n} := \{ u \in \{a,b\}^* \mid m\,|u|_a = n\,|u|_b \}$ is a language synchronized by $\{\, i\,p\,,\ p \xrightarrow{(a,m)} p\,,\ p \xrightarrow{(b,-n)} p \,\}$.
For $m,n > 0$, $L_{m,n}$ is not a visibly pushdown language in the sense of [AM 04].

Example 5.7 The linear language $\{ uc\widetilde{u} \mid u \in \{a,b\}^* \}$ for $\widetilde{u}$ the mirror of u, is synchronized by $\{\, i\,p\,,\ p \xrightarrow{(a,1)} p\,,\ p \xrightarrow{(b,1)} p\,,\ ,p \xrightarrow{(c,0)} q\,,\ q \xrightarrow{(a,-1)} q\,,\ q \xrightarrow{(b,-1)} q \,\}$. Such a language cannot be synchronized by an one state transducer: we need several integers for the labels a and b (here 1 and -1).

We establish effective closure properties of the synchronized languages by A as for the rational languages: we apply the classical constructions on finite automata to the graph grammars generating by $\| \ \|_A$. By synchronization product of $\overrightarrow{A}$ by any finite automaton, we deduce that any rational language included in $L(A)$ is synchronized by A. By disjoint union of two graph grammars generating by $\| \ \|_A$, we deduce that the union of two synchronized languages by A remains synchronized. By synchronization product of two graph grammars generating by $\| \ \|_A$, we obtain that the intersection of two synchronized languages by A remains synchronized. Finally by a determinization of any graph grammar generating by $\| \ \|_A$, we show that any synchronized language is deterministic context-free, and its complement with respect to $L(A)$ remains synchronized.

Theorem 5.8 *For any transducer A, the class of synchronized languages contains all the rational languages in $L(A)$, is a subfamily of deterministic real-time context-free languages, and is an effective boolean algebra with respect to $L(A)$.*

This generalization of the visibly pushdown automata has permitted to work with unrestricted pushdown automata (the synchronization is independent of the length of the words in the rules), and by allowing any integer and several states (instead of $-1, 0, 1$ and a unique state). This paper also indicates that the deterministic graph grammars can be a powerful tool to investigate properties of context-free languages.

Thanks to Christof Löding for a survey on the visibly pushdown automata which has been at the origin of this paper. The first half part of this paper has been done during a stay in Udine; many thanks to Angelo Montanari for his invitation. The second half part of this paper has been done during a stay in Aachen; many thanks to Wolfgang Thomas for his support. Thanks to Arnaud Carayol for his help in the drafting of this paper.

References

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

[Bü 64] R. Büchi *Regular canonical systems*, Archiv für Mathematische Logik und Grundlagenforschung 6, 91–111 (1964) [also in *The collected works of J. Richard Büchi*, Springer-Verlag, New York, S. Mac Lane, D. Siefkes (Eds.), 317–337 (1990)].

[Ca 90] D. Caucal *On the regular structure of prefix rewriting*, 15th CAAP, LNCS 431, A. Arnold (Ed.), 87–102 (1990) [a full version is in Theoretical Computer Science 106, 61–86 (1992)].

[Ca 95] D. Caucal *Bisimulation of context-free grammars and of pushdown automata*, CSLI volume 53, Stanford, 85–106 (1995).

[Co 90] B. Courcelle *Graph rewriting: an algebraic and logic approach*, Handbook of Theoretical Computer Science Vol. B, J. Leeuwen (Ed.), Elsevier, 193–242 (1990).

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