

Path Algorithms on Regular Graphs

Didier Caucal, Trong Hieu Dinh

▶ **To cite this version:**

Didier Caucal, Trong Hieu Dinh. Path Algorithms on Regular Graphs. FCT'07, Aug 2007, Budapest, Hungary. pp.199-212, 10.1007/978-3-540-74240-1_18 . hal-00620155

HAL Id: hal-00620155
https://hal.science/hal-00620155

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.

Path algorithms on regular graphs

Didier Caucal and Dinh Trong Hieu

IGM–CNRS, 5 bd Descartes, 77454 Marne-la-Vallée, France

caucal@univ-mlv.fr dinh@univ-mlv.fr

Abstract. We consider standard algorithms of finite graph theory, like for instance shortest path algorithms. We present two general methods to polynomially extend these algorithms to infinite graphs generated by deterministic graph grammars.

1 Introduction

The regularity of infinite graphs was first considered by Muller and Schupp [6]. They studied the transition graphs of pushdown automata, called pushdown graphs, and showed that their connected components are the connected graphs of finite degree whose decomposition by distance from a(ny) vertex yields finitely many non-isomorphic connected components. More generally, a graph is regular if it admits a finite decomposition (not necessarily by distance) or, equivalently, if it can be generated by a deterministic graph grammar. Regular graphs have been defined by Courcelle and called hyperedge replacement equational graphs [3]. Any connected regular graph is finitely decomposable by distance, hence the connected components of pushdown graphs coincide with the connected regular graphs of finite degree. This identity was also generalized to non connected graphs: the regular restrictions of pushdown graphs are the regular graphs of finite degree [1]. A regular graph may be seen as an infinite automaton [9] : it recognizes the set of path labels between two given finite vertex sets. Even though a regular graph can have vertices of infinite degree, regular graphs recognize exactly the family of context-free languages. Many publications focus on finite graphs and their applications, but few deal with regular graphs. This paper is a first step towards developing an algorithmic theory of regular graphs.

We consider a set of edge labels forming an idempotent and continuous semiring. For any graph, we define the value of a path to be the product of its successive labels, that we extend by summation to any set of paths between vertex sets. The value (or the computation) of a grammar is then a vector whose components are the values of the graphs generated from each of the left hand sides of the grammar. In the case of deterministic graph grammars in a restricted form, we show the equivalence between the algebraic and operational semantics: the value of the grammar is the least upper bound of the sequence obtained by iteratively applying the interpretation of the grammar from the least element (Theorem 2.5). We then present two methods to compute the value of a grammar over a commutative (idempotent and continuous) semiring. The first method

applies to linear semirings having a greatest element reached by any increasing Kleene sequence. In this case, the grammar value is determined by applying the grammar interpretation a number of times (Theorem 3.1). The second algorithm works with any commutative semiring and is a simple graph generalization of the Hopkins-Kozen method [5] developped for context-free grammars (cf. Corollary 3.5). Finally we give a polynomial transformation of any grammar into an equivalent grammar in restricted form (Proposition 4.3). We end up with a polynomial complexity algorithm to solve the shortest path problem on regular graphs using graph grammars (Corollary 4.4).

2 Computations with graph grammars

We consider a graph as a denumerable set of labelled arcs. The arc labels are elements of an idempotent and continuous semiring. The value of a path is the product of its successive labels. The value of a graph from initial vertices to final vertices is the sum of the values of its paths from an initial vertex to a final vertex.
We want to compute values of graphs generated by a deterministic graph grammar. In this section, we restrict ourselves to grammars in which each left hand side is a labelled arc from vertex 1 to vertex 2, and no right hand side has an arc from 2 or an arc to 1. The value of any left hand side is the value from 1 to 2 of its generated graph. The value of a grammar is the vector of its left hand side values. By considering a grammar as a function from and into the semiring (to the power of the rule number), its iterative application from the least element gives by least upper bound the value of the grammar (Theorem 2.5).

Recall that an algebra $(K, +, \cdot, 0, 1)$ is a *semiring* if $(K, +, 0)$ is a commutative monoid, $(K, \cdot, 1)$ is a monoid, *multiplication* $\cdot$ distributes over *addition* $+$, and 0 annihilates K: $0 \cdot a = a \cdot 0 = 0$ for any $a \in K$.
We say that a semiring K is *complete* if for the following relation:

$$a \leq b \quad \text{if} \quad a + c = b \quad \text{for some} \quad c \in K,$$

any increasing sequence $a_0 \leq a_1 \leq \ldots \leq a_n \leq \ldots$ has a least upper bound $\bigvee_n a_n$. This implies that $\leq$ is a partial order, 0 is the least element, and the operations $+$ and $\cdot$ are monotonous:

$$a \leq b \quad \wedge \quad a' \leq b' \quad \Longrightarrow \quad a + a' \leq b + b' \quad \wedge \quad a \cdot a' \leq b \cdot b'.$$

This also permits to define the sum of any sequence $(a_n)_{n \geq 0}$ in K by

$$\sum_{n \geq 0} a_n \quad := \quad \bigvee_{n \geq 0} \Big(\sum_{i=0}^{n} a_i \Big).$$

We say that a semiring K is *idempotent* if $+$ is idempotent:

$$a + a = a \quad \text{for any} \quad a \in K \qquad \text{or equivalently} \quad 1 + 1 = 1.$$

For K complete and idempotent, we have $\sum_{n \geq 0} a_n = \sum_{n \geq 0} a_{\pi(n)}$ for any permutation π, and this sum is also denoted $\sum_{n \geq 0} \{ a_n \mid n \geq 0 \}$.
We say that a complete semiring K is *continuous* if $+$ and $\cdot$ are continuous *i.e.* commute with $\bigvee$: for any increasing sequences $(a_n)_{n \geq 0}$ and $(b_n)_{n \geq 0}$,

$$\big(\bigvee_{n \geq 0} a_n \big) * \big(\bigvee_{n \geq 0} b_n \big) = \bigvee_{n \geq 0} (a_n * b_n) \quad \text{where} \quad * \quad \text{stands for} \quad + \quad \text{or} \quad \cdot$$

So $\big(\sum_{n\geq 0}\, a_n\big) + \big(\sum_{n\geq 0}\, b_n\big) \;=\; \sum_{n\geq 0} (a_n + b_n)$ for any $a_n\,,\, b_n \,\in\, K\,.$
For K continuous and idempotent and for any sequences $(a_n)_{n\,\geq\,0}$ and $(b_n)_{n\,\geq\,0}$

$$\big(\sum_{n\geq 0}\, a_n\big)\cdot\big(\sum_{n\geq 0}\, b_n\big) \;=\; \sum\{\; a_i{\cdot}b_j \mid i,j\geq 0\;\}\,.$$

From now on K will denote a complete and idempotent semiring.

Recall that $(K^*,.,\varepsilon)$ is the free monoid generated by the set K, *i.e.* $K^* = \bigcup_{n\geq 0} K^n$ is the set of tuples of elements of K where . is the tuple concatenation and the neutral element ε is the 0-tuple (). Any tuple $(a_1,\ldots,a_p)$ is written $a_1\ldots a_p$. The identity relation on K is extended to the morphism $[\![\ ]\!]$ from $(K^*,.,\varepsilon)$ into the monoid $(K,\cdot,1)$:

$$[\![\varepsilon]\!] = 1 \quad ; \quad \forall\, a\in K,\;\; [\![a]\!] = a \quad ; \quad \forall\, u,v\in K^*,\;\; [\![u.v]\!] \;=\; [\![u]\!]{\cdot}[\![v]\!]\,.$$

We extend by summation the *value* mapping $[\![\ ]\!]$ to any language:

$$[\![U]\!] \;:=\; \sum\{\; [\![u]\!] \mid u\in U\;\} \;\text{ for any }\; U\subseteq K^*.$$

Let L be an arbitrary set. Here a L-graph is just a set of arcs labelled in L. Precisely, a *L-graph* G is a subset of $V{\times}L{\times}V$ where V is an arbitrary set such that the *vertex* set of G defined by

$$V_G \;:=\; \{\; s \mid \exists\; a,t,\;\; (s,a,t)\in G \;\vee\; (t,a,s)\in G\;\}$$

is finite or countable, and its *label* set

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

Any (s,a,t) of G is a *labelled arc* of *source* s, of *goal* (or target) t, with label a, and is identified with the labelled transition $s \xrightarrow[G]{a} t$, or directly $s \xrightarrow{a} t$ if G is understood. The transformation of a graph G by a function h from V_G into a set V is the graph

$$h(G) \;:=\; \{\; h(s) \xrightarrow{a} h(t) \mid s \xrightarrow[G]{a} t \;\wedge\; s,t\in \mathit{Dom}(h)\;\}\,.$$

An *isomorphism* h from a graph G to a graph H is a bijection from V_G to V_H such that $h(G) = H$. The *language recognized* by G from a vertex set I to a vertex set F is the set of labels of the paths from I to F:

$$L(G,I,F) \;:=\; \{\; a_1\ldots a_n \mid n\geq 0 \;\wedge\; \exists\; s_0,\ldots,s_n,\;\; s_0\in I \;\wedge\; s_n\in F \;\wedge\; s_0 \xrightarrow{a_1} s_1 \ldots\; s_{n-1} \xrightarrow{a_n} s_n\;\}.$$

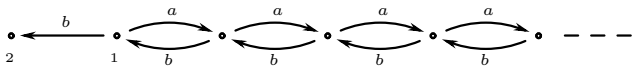
In particular $\varepsilon\in L(G,I,F)$ for $I\cap F\neq\emptyset$.
We also write $s \xrightarrow[G]{}{}^{*}\; t$ for $L(G,s,t)\neq\emptyset$, and $s \xRightarrow[G]{u} t$ for $u\in L(G,s,t)$.
The *graph value* on a K-graph G from $I\subseteq V_G$ to $F\subseteq V_G$ is defined as the value of its recognized language:

$$[\![G,I,F]\!] \;:=\; [\![L(G,I,F)]\!]\,.$$

For any graph G and by identifying any label $a\in L_G$ with $\{a\}$, we can take the semiring $(2^{L_G^*},\cup,.,\emptyset,\{\varepsilon\})$ of languages in which $[\![u]\!] = \{u\}$ for any $u\in L_G^*$.
For this semiring, the values of G are the recognized languages:

$$[\![G,I,F]\!] \;=\; L(G,I,F) \quad \text{for any } \; I,F\subseteq L_G^*\,.$$

Example 2.1 We consider the following graph G:



a) By identifying a with $\{a\}$ and b with $\{b\}$, the value $[\![G, 1, 2]\!]$ of G from vertex 1 to vertex 2 for the semiring $(2^{\{a,b\}^*}, \cup, ., \emptyset, \{\varepsilon\})$ is the Lukasiewicz language.

b) Now taking the semiring $(2^{\mathbb{N}\times\mathbb{N}}, \cup, +, \emptyset, (0,0))$ with $a = \{(1,0)\}$ and $b = \{(0,1)\}$, the value $[\![G, 1, 2]\!]$ is the Parikh image $\{ (n, n+1) \mid n \geq 0 \}$ of the Lukasiewicz language.

c) Taking $a, b \in \mathbb{R}$ and the semiring $(\mathbb{R} \cup \{-\omega, \omega\}, Min, +, \omega, 0)$, the value

$$[\![G, 1, 2]\!] \quad = \quad \begin{cases} b & \text{if } a + b \geq 0, \\ -\omega & \text{otherwise,} \end{cases}$$

is the smallest value labelling the paths from 1 to 2.

d) Finally having $a, b \in \mathbb{R}_+$ the set of non negative real numbers and for the semiring $(\mathbb{R}_+ \cup \{\omega\}, Max, \times, 0, 1)$, the value from 1 to 2 is

$$[\![G, 1, 2]\!] \quad = \quad \begin{cases} b \text{ if } a \times b \leq 1, \\ \omega \text{ otherwise.} \end{cases}$$

For continuous and idempotent semirings, we want to extend algorithms computing the values of finite graphs to the graphs generated by graph grammars.
In this section, we restrict ourselves to grammars in which each left hand side is a labelled arc from vertex 1 to vertex 2, and no right hand side has an arc of goal 1 or of source 2.
Precisely a *2-grammar* R is a finite set of rules of the form:

$$(1, A, 2) \longrightarrow H$$

where $(1, A, 2)$ is an arc labelled by A from vertex 1 to vertex 2, and H is a finite graph.
The labels of the left hand sides form the set N_R of *non-terminals* of R:

$$N_R := \{ A \mid (1, A, 2) \in Dom(R) \}$$

and the remaining labels in R form the set T_R of *terminals*:

$$T_R := \{ A \notin N_R \mid \exists\, H \in Im(R),\ \exists\, s, t,\ (s, A, t) \in H \}.$$

Furthermore we require that each right hand side has no arc of goal 1 or of source 2:

$$\forall\, H \in Im(R),\ \forall\, (s, a, t) \in H,\ s \neq 2 \ \wedge\ t \neq 1.$$

We say that R is an *acyclic grammar* if its right hand sides are acyclic graphs.
Starting from any graph, we want a graph grammar to generate a unique graph up to isomorphism. We thus restrict ourselves to *deterministic* 2-grammars in which two rules have distinct left hand sides:

$$((1, A, 2), H), ((1, A, 2), K) \in R \quad \Longrightarrow \quad H = K.$$

An example is given below.

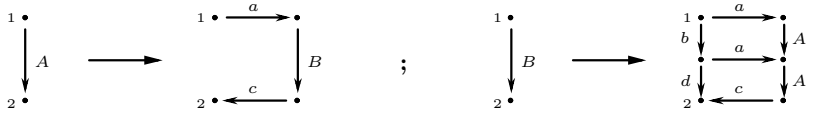


Figure 2.2 *A deterministic graph grammar.*

Starting from a graph, this grammar generates a unique infinite graph obtained by applying indefinitely parallel rewritings. Precisely and for any 2-grammar R, the *rewriting* $\underset{R}{\longrightarrow}$ is the binary relation between graphs defined by $M \underset{R}{\longrightarrow} N$ if we can choose a non-terminal arc $X = (s, A, t)$ in M and a right hand side H of A in R to replace X by H in M:

$$N \;=\; (M - \{X\}) \cup h(H)$$

for some function h mapping vertex 1 to s, vertex 2 to t, 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 non-terminal arc X is extended in an obvious way to the rewriting $\underset{R,\,E}{\longrightarrow}$ of any subset E of non-terminal arcs. A *complete parallel rewriting* $\underset{R}{\Longrightarrow}$ is the rewriting according to the set of all non-terminal arcs: $M \underset{R}{\Longrightarrow} N$ if $M \underset{R,\,E}{\longrightarrow} N$ where E is the set of all non-terminal arcs of M.

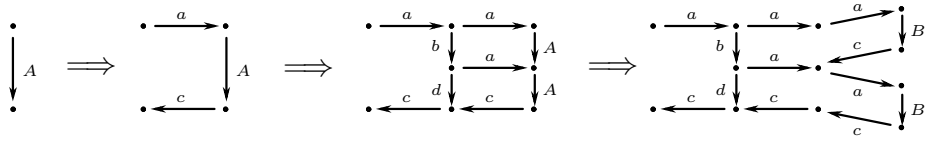


Figure 2.3 *Parallel rewritings according to the grammar of Figure 2.2.*

Due to the two non-terminal arcs in the right hand side of B for the grammar of Figure 2.2, we get after n parallel rewritings from $H_0 = \{\, 1 \xrightarrow{A} 2 \,\}$, a graph H_n having an exponential number $|H_n|$ of arcs.

The *derivation* $\underset{R}{\Longrightarrow}^*$ is the reflexive and transitive closure for the composition of the parallel rewriting $\underset{R}{\Longrightarrow}$ *i.e.* $G \underset{R}{\Longrightarrow}^* H$ if H is obtained from G by a consecutive sequence of parallel rewritings. We denote by $[M]$ the set of terminal arcs of M:

$$[M] \;:=\; M \cap V_M \times T_R \times V_M \,.$$

We now assume that any 2-grammar is deterministic.

A 2-*grammar over* K is a 2-grammar R such that $T_R \subseteq K$.

A graph G is *generated by a 2-grammar* R from a graph H if G is isomorphic to a graph in the following set $R^\omega(H)$ of isomorphic graphs:

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

For instance by iterating indefinitely the derivation of Figure 2.3, we get the infinite graph depicted below.

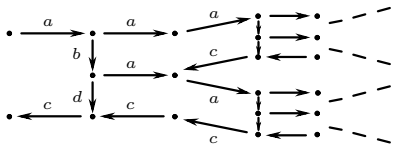


Figure 2.4 *Graph generated by the grammar of Figure 2.2.*

For any 2-grammar R and any non-terminal $A \in N_R$,
its right hand side in R is $R(\{(1,A,2)\})$, also denoted $R(A)$,
the generated graph by R from A is $R^\omega(\{(1,A,2)\})$, also denoted $R^\omega(A)$,
if $T_R \subseteq K$, the value of A by R is $[\![R^\omega(A), 1, 2]\!]$ also denoted $[\![R^\omega(A)]\!]$.

Given a 2-grammar R over K, we want to compute $[\![R^\omega(A)]\!]$ for any non-terminal A. First we put R in the following *reduced* form S:

$$N_S \;=\; \{\; A \in N_R \mid L(R^\omega(A),1,2) \neq \emptyset \;\} \quad \text{and} \quad 0 \notin T_S$$

and for every $A \in N_S$, we have

$$[\![S^\omega(A)]\!] \;=\; [\![R^\omega(A)]\!] \quad \text{and} \quad \forall\, s \in V_{S(A)},\; 1 \underset{S(A)}{\longrightarrow}{}^{*}\; s \underset{S(A)}{\longrightarrow}{}^{*}\; 2\,.$$

We begin by removing the arcs labelled by 0 in the right hand sides of R. Then we compute the set

$$E \;=\; \{\; A \in N_R \mid L(R^\omega(A),1,2) \neq \emptyset \;\}$$

of non-terminals whose generated graph has a path from 1 to 2. This set E is the least fixed point of the following equation:

$$E \;=\; \{\; A \in N_R \mid \exists\, u \in (T_R \cup E)^*,\; 1 \underset{R(A)}{\overset{u}{\Longrightarrow}} 2 \;\}\,.$$

This allows us to restrict to the rules of non-terminals in E and to remove the arcs labelled by a non-terminal not in E in the right hand sides of the grammar. Finally we get S by restricting each right hand side to the vertices accessible from 1 and co-accessible from 2. The overall time complexity is quadratic according to the description length of R (due to the computation of E).
Henceforth we assume that any 2-grammar R is in reduced form. In that case,

$$R \text{ is acyclic} \iff R^\omega(A) \text{ is acyclic for all } A \in N_R\,.$$

Given a 2-grammar R over K, we order the set $N_R = \{A_1, \ldots, A_p\}$ of its p non-terminals. We want to determine the *grammar value* $[\![R^\omega]\!]$ of R as being the following tuple in K^p:

$$[\![R^\omega]\!] \;:=\; ([\![R^\omega(A_1)]\!], \ldots, [\![R^\omega(A_p)]\!])\,.$$

A standard semantic way is to define the *interpretation* $[\![R]\!]$ of R as being the mapping:

$$\begin{aligned} [\![R]\!] : \quad & K^p \longrightarrow K^p \\ & (a_1, \ldots, a_p) \longmapsto ([\![R(A_1)[a_1, \ldots, a_p]]\!], \ldots, [\![R(A_p)[a_1, \ldots, a_p]]\!]) \end{aligned}$$

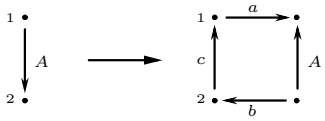
where $G[a_1, \ldots, a_p]$ is the graph obtained from graph G by replacing each label A_i by a_i for every $1 \leq i \leq p$. This interpretation $[\![R]\!]$ is a continuous mapping.
As K^p is a complete set for the product order whose least element is $0 = (0, \ldots, 0)$, we can apply the Knaster-Tarsky theorem: the least upper bound of the iterative application of $[\![R]\!]$ from 0, is the least fixed point of $[\![R]\!]$ denoted $\mu\, [\![R]\!]$. A first result is that this least upper bound is also the value $[\![R^\omega]\!]$ of R.

Theorem 2.5 *For any continuous and idempotent semiring K and for any 2-grammar R over K,*

$$[\![R^\omega]\!] \;=\; \bigvee\nolimits_{n \geq 0} [\![R]\!]^n(0) \;=\; \mu\, [\![R]\!]\,.$$

The first equality holds for K complete and idempotent when R is acyclic.

Note that the first equality of Theorem 2.5 can be false if we allow in a right hand side of the grammar an arc of goal 1 or of source 2. For instance, taking the semiring $(2^{\{a,b,c\}^*}, \cup, ., \emptyset, \{\varepsilon\})$ and the grammar R reduced to the rule:



we have $\bigvee_{n\geq 0} [\![R]\!]^n(\emptyset) = \emptyset \neq [\![R^\omega]\!]$.

3 Computation algorithms

We present two general algorithms to compute the value of any 2-grammar R for idempotent and continuous semirings which are commutative. The first algorithm compares the differences between the $|R|$-th ($|R|$ is the number of rules) approximant $[\![R]\!]^{|R|}(0)$ with the next one in order to detect increments strictly greater than 1 in the value of the generated graphs (cf. Theorem 3.1). The second algorithm is a simple graph generalization of the Hopkins-Kozen method [5] developped for context-free grammars (cf. Corollary 3.5). Although the first algorithm is more efficient than the second one, it works with semirings whose order is linear and admits a greatest element.

Given a semiring and an algorithm to compute $[\![G, I, F]\!]$ for any finite graph G (and vertex sets I, F) of time complexity $C_{G,I,F}$, we want to use this algorithm to compute $[\![R^\omega]\!]$ for any 2-grammar R. The complexity will be expressed with the following parameters:

$|R| = |N_R|$ the number p of rules of R (its cardinality),
$\ell_R := \sum \{ |R(A)| \mid A \in N_R \}$ the *length* of the description of R,
$C_R := \sum \{ C_{R(A)[0],1,2} \mid A \in N_R \}$ the time complexity to compute $[\![R]\!](0)$.

Theorem 2.5 requires continuous and idempotent semirings. Our algorithms require that these semirings are commutative: the multiplication $\cdot$ is commutative. A *cci-semiring* means a commutative continuous idempotent semiring.
The first algorithm also needs that the semirings be *linear*, meaning that the relation $\leq$ is a linear (total) order. Let us describe this first algorithm.
For any 2-grammar R, Theorem 2.5 gives a standard way to compute $[\![R^\omega]\!]$: if there exists n such that $[\![R]\!]^n(0) = [\![R]\!]^{n+1}(0)$ then $[\![R^\omega]\!] = [\![R]\!]^n(0)$. This is true for any *bounded semiring i.e.* with no infinite increasing sequence. But the sequence $\left([\![R]\!]^n(0)\right)_n$ is strict in general. We compare the vectors $[\![R]\!]^p(0)$ and $[\![R]\!]^{p+1}(0)$ (with $p = |R|$) and determine the ranks for which they differ:

$$E = \{ A_i \mid ([\![R]\!]^p(0))_i \neq ([\![R]\!]^{p+1}(0))_i \}.$$

By a classical pumping argument and for each $A \in E$, we can find an increment

> 1 in the value of its generated graph. Assuming that we have a greatest element $\top$ reached by any strict increasing sequence $(a^n)_{n\geq 0}$, we get $[\![R^\omega(A)]\!] = \top$. This is also true for any non-terminal having A in its right hand side. We complete E into the set $\overline{E}$ which is the least fixed point of the following equation:

$$\overline{E} \;=\; E \cup \{\, A \in N_R \mid L_{R(A)} \cap \overline{E} \neq \emptyset \,\}\,.$$

We deduce that for any rank $1 \leq i \leq p$,

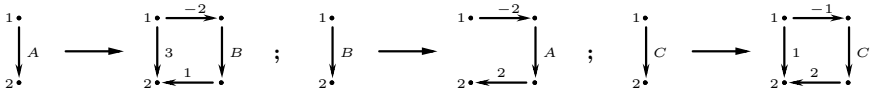
$$\big([\![R^\omega]\!]\big)_i \;=\; \begin{cases} \top & \text{if } A_i \in \overline{E}, \\ \big([\![R]\!]^p(0)\big)_i & \text{otherwise.} \end{cases}$$

The time complexity is the complexity to compute $[\![R]\!]^p(0)$.

Theorem 3.1 *For any linear cci-semiring K having a greatest element $\top$ such that $\bigvee_n a^n = \top$ for every $a > 1$ and $a{\cdot}\top = \top$ for every $a \neq 0$, we can compute $[\![R^\omega]\!]$ in time $O(|R|\, C_R)$ for any 2-grammar R over K.*

We can apply Theorem 3.1 to the semiring $(\mathbb{R} \cup \{-\omega, \omega\}, Min, +, \omega, 0)$ of Example 2.1 (c). This allows us to solve the shortest path problem on any graph generated by a 2-grammar. We take Floyd's algorithm (with negative cycle test) on finite graphs. For any 2-grammar R, we have C_R in $O(\ell_R^3)$ thus the complexity to compute $[\![R^\omega]\!]$ is $O(|R|\, \ell_R^3)$.

When R is acyclic, we can take Bellman's algorithm on finite graphs; so C_R is $O(\ell_R)$ and the complexity for $[\![R^\omega]\!]$ is $O(|R|\, \ell_R)$, hence quadratic time. For instance taking the following 2-grammar R:



we have the following approximants:

n	0	1	2	3	4
$\big([\![R]\!]^n(0)\big)_A$	ω	3	3	2	2
$\big([\![R]\!]^n(0)\big)_B$	ω	ω	3	3	2
$\big([\![R]\!]^n(0)\big)_C$	ω	1	1	1	1

giving by Theorem 3.1 the value $[\![R^\omega]\!] = (-\omega, -\omega, 1)$ *i.e.*

$$[\![R^\omega(A)]\!] \;=\; [\![R^\omega(B)]\!] \;=\; -\omega \quad \text{and} \quad [\![R^\omega(C)]\!] \;=\; 1\,.$$

Note that we can apply Theorem 3.1 to the semiring $(\mathbb{R}_+ \cup \{\omega\}, Max, \times, 0, 1)$ of Example 2.1 (d).

For the second algorithm, we need to show that (graph) 2-grammars are language-equivalent to cf-grammars (on words), 'cf' is short for context-free.
Recall that a *context-free grammar* P is a finite binary relation on words in which each left hand side is a letter called a non-terminal, and the remaining letters of P are terminals. By denoting N_P and T_P the respectives sets of non-terminals and terminals of P, the rewriting $\underset{P}{\longrightarrow}$ according to P is the binary

relation on $(N_P \cup T_P)^*$ defined by

$$UAV \underset{P}{\longrightarrow} UWV \quad \text{if } (A,W) \in P \text{ and } U,V \in (N_P \cup T_P)^*.$$

The derivation $\underset{P}{\longrightarrow}^*$ is the reflexive and transitive closure of $\underset{P}{\longrightarrow}$ with respect to composition. The language $L(P,U)$ generated by P from any $U \in (N_P \cup T_P)^*$ is the set of terminal words deriving from U:

$$L(P,U) := \{\, u \in T_P^* \mid U \underset{P}{\longrightarrow}^* u \,\}.$$

2-grammars and cf-grammars are language-equivalent with linear time translations.

Proposition 3.2 a) *We can transform in linear time any 2-grammar R into a cf-grammar P such that $L(R^\omega(A),1,2) = L(P,A)$ for any $A \in N_R$.*

b) *We can transform in linear time any cf-grammar P into an acyclic 2-grammar R such that $L(P,A) = L(R^\omega(A),1,2)$ for any $A \in N_P$.*

The first transformation is analogous to the translation of any finite automaton into an equivalent right linear grammar. For each non-terminal $A \in N_R$, let h_A be a vertex renaming of $R(A)$ such that $h_A(1) = A$ and $h_A(2) = \varepsilon$, and the image $Im(h_A) - \{\varepsilon\}$ is a set of symbols with $Im(h_A) \cap Im(h_B) = \{\varepsilon\}$ for any $B \in N_R - \{A\}$. We define:

$$P := \{\, (h_A(s), a h_A(t)) \mid \exists\, A \in N_R,\ s \underset{R(A)}{\overset{a}{\longrightarrow}} t \,\}.$$

Note that each right hand side of P is a word of length at most 2, and the number of non-terminals of P depends on the description length ℓ_R of R:

$$|N_P| = \left(\textstyle\sum_{A \in N_R} |V_{R(A)}|\right) - |N_R| .$$

For the second transformation, we have $N_R = N_P$ and for each $A \in N_P$, its right hand side in R is the set of distinct paths from 1 to 2 labelled by the right hand sides of A in P. Note that by using the two transformations of Proposition 3.2, we can transform in linear time any 2-grammar into a language equivalent acyclic 2-grammar. Then we can apply Theorem 3.1 with the Bellman's algorithm for the shortest path problem.

Corollary 3.3 *For the semiring $(\mathbb{R} \cup \{-\omega,\omega\}, Min, +, \omega, 0)$, any 2-grammar R and any $A \in N_R$, the shortest path problem $[\![R^\omega(A12), 1, 2]\!]$ can be solved in $O(\ell_R^{\,2})$, and in $O(|R|\,\ell_R)$ when $R^\omega(A12)$ is acyclic.*

In particular for any context-free grammar P, the shortest path problem can be solved in $O(|P|\,\ell_P)$.

Starting from any pushdown automaton R, it is easy to transform R into a pushdown automaton R' recognizing the same language (by final states and/or empty stack) such that each right hand side is a state followed by at most two stack letters; the number of rules $|R'|$ hence its length of description $\ell_{R'}$ are in $O(|R|.\ell_R^2)$. Then we apply to R' the usual transformation to get an equivalent cf-grammar P: $|P|$ and ℓ_P are in $O(|R'|^3)$. Thus for any pushdown automaton R, the shortest path problem can be solved in $O((|R|.\ell_R^2)^6) = O(|R|^6.\ell_R^{12})$.

In next section, Corollary 3.3 will be extended to Corollary 4.4 starting from any generalized graph grammar.
Note also that by the two transformations of Proposition 3.2 and by the first equality of Theorem 2.5, we can compute the value of any 2-grammar for non continuous (but complete and idempotent) semirings.

A cf-grammar over K is a context-free grammar P such that $T_P \subseteq K$.
By ordering the non-terminal set: $N_P = \{A_1, \ldots, A_p\}$, the *cf-grammar value* is

$$[\![L(P)]\!] := ([\![L(P, A_1)]\!], \ldots, [\![L(P, A_p)]\!]) .$$

For the semiring $(2^{\mathbb{N}^q}, \cup, +, \emptyset, (0, \ldots, 0))$ in Example 2.1 (b) of commutative languages over q terminals of the form $\{(0, \ldots, 0, 1, 0, \ldots, 0)\}$, a first solution to determine $[\![L(P)]\!]$ was given by Parikh [7]. This method was refined in [8] and generalized in [5], and works for any *cci*-semiring. This last method is presented briefly below.
The *derivative* of any language $E \subseteq (N_P \cup T_P)^*$ by $A \in N_P$ is the following language:

$$\frac{\partial E}{\partial A} := \{\, UV \mid UAV \in E \,\}$$

and as $+$ is idempotent, we can restrict to remove only the first occurrence of A *i.e.* $|U|_A = 0$. The *Jacobian matrix* of P is

$$P' := \left(\frac{\partial P(A_i)}{\partial A_j}\right)_{1 \le i,j \le p}$$

where $P(A) = \{\, U \mid (A, U) \in P \,\}$ is the image of $A \in N_P$ by P.
The *interpretation* $[\![P']\!]$ of P' is the mapping:

$$[\![P']\!] : \quad K^p \longrightarrow K^{p \times p}$$
$$(a_1, \ldots, a_p) \longmapsto \left([\![\tfrac{\partial P(A_i)}{\partial A_j} [a_1, \ldots, a_p]]\!]\right)_{1 \le i,j \le p}$$

where $E[a_1, \ldots, a_p]$ is the language over K obtained from $E \subseteq (N_P \cup T_P)^*$ by replacing in the words of E each label A_i by a_i for every $1 \le i \le p$.
The *Hopkins-Kozen transformation* is the mapping:

$$\mathrm{HK} : \quad K^p \longrightarrow K^p$$
$$(a_1, \ldots, a_p) \longmapsto \left([\![P']\!](a_1, \ldots, a_p)\right)^* \times (a_1, \ldots, a_p)$$

where $M \times \vec{v} := \left(M \cdot (\vec{v})^t\right)^t$ with ^{t} for vector transposition, $\cdot$ for matrix multiplication and $*$ for its Kleene closure.
By applying iteratively this transformation from $([\![P(A_1)[0]]\!], \ldots, [\![P(A_p)[0]]\!])$, we get an increasing sequence which reaches its least upper bound $[\![L(P)]\!]$ after a finite number of iterations [5], and even after p iterations [4].

Theorem 3.4 **[5] [4]** *For any cci-semiring K and any cf-grammar P over K,*

$$[\![L(P)]\!] = HK^{|P|}([\![P(A_1) \cap T^*]\!], \ldots, [\![P(A_p) \cap T^*]\!]) .$$

Theorem 3.4 remains true even if the languages $P(A_1), \ldots, P(A_p)$ are infinite [4].
By applying the transformation of Proposition 3.2 (a) to any 2-grammar R over K, we get in linear time a cf-grammar $\widehat{R}$ over K such that $[\![R^\omega]\!] = [\![L(\widehat{R})]\!]$ that we can compute by Theorem 3.4 for any cci-semiring K. However the non-

terminal set $N_{\widehat{R}}$ depends on ℓ_R and not on $|R|$. So it is more efficient to extend Theorem 3.4 directly to 2-grammars.
Let R be any graph grammar over K with non-terminal set $N_R = \{A_1, \ldots, A_p\}$. Let us extend the derivative to any $(K \cup N_R)$-graph G with $1, 2, \in V_G$.
Let $A \in N_R$. We take a new symbol A_0 and we define the synchronization product

$$H := \big(G \cup \{\ s \xrightarrow{A_0} t \mid s \xrightarrow[G]{A} t\ \}\big) \times S_A$$
$$\text{with } S_A := \{\ 1 \xrightarrow{a} 1 \mid a \in K \cup N_P - \{A\}\ \} \cup \{1 \xrightarrow{A_0} 2\}$$
$$\cup\ \{\ 2 \xrightarrow{a} 2 \mid a \in K \cup N_P\ \}.$$

Taking the vertex renaming h of H defined by $h(1,1) = 1$, $h(2,2) = 2$, and $h(x) = x$ for any $x \in V_H - \{(1,1), (2,2)\}$, and by restricting $h(H)$ to the graph $\overline{H}$ with vertices accessible from 1 and co-accessible from 2, the *derivative* of graph G by A is the graph:

$$\tfrac{\partial G}{\partial A} := \big(\overline{H} - V_{\overline{H}} \times A_0 \times V_{\overline{H}}\big) \cup \{\ s \xrightarrow{1} t \mid s \xrightarrow[\overline{H}]{A_0} t\ \}.$$

The *Jacobian matrix* of R is $R' := \Big(\frac{\partial R(A_i)}{\partial A_j}\Big)_{1 \le i,j \le p}$ and its interpretation is

$$[\![\, R' \,]\!](a_1, \ldots, a_p) := \big([\![\, \tfrac{\partial R(A_i)}{\partial A_j}\, [a_1, \ldots, a_p] \,]\!]\big)_{1 \le i,j \le p} \quad \text{for any } a_1, \ldots, a_p \in K.$$

So $[\![\, R' \,]\!] = [\![\, P' \,]\!]$ for the infinite cf-grammar P: $P(A) = L(R(A), 1, 2)$ for any $A \in N_R$.

Corollary 3.5 *For any cci-semiring K and any graph grammar R over K,*

$$[\![\, R^\omega \,]\!] = HK^{|R|}\big([\![\, R \,]\!](0)\big).$$

For $+, \cdot, *$ in K in $O(1)$, the complexity is in $O(|R|^4 + |R|^2 C_R)$.

Contrary to Theorem 3.1, this corollary can be used for any *cci*-semiring. The computation of the Jacobian matrix R' is $O\big(\sum_{A \in N_R} |R||R(A)|\big) = O(|R|\ell_R)$. We compute $[\![\, R' \,]\!](a_1, \ldots, a_p)$ in $O\big(\sum_{A \in N_R} |R| C_{R(A)[a_1,\ldots,a_p],1,2}\big) = O(|R| C_R)$. Having a square matrix $M \in K^{p \times p}$ and a vector $\vec{a} \in K^p$, the value of $\vec{b} = M^*.\vec{a}$ corresponds to solving the linear system $\vec{b} = \vec{a} + M.\vec{b}$ of p equations with p variables $\{b_1, \ldots, b_p\}$. By the Gauss elimination method, we use $O(p^3)$ operations $+$ and $\cdot$ operations, and $O(p)$ $*$ operations.
For the semiring $(\mathbb{R} \cup \{-\omega, \omega\}, Min, +, \omega, 0)$ of Example 2.1 (c) and using Floyd's algorithm over finite graphs, we get a complexity in $O(|R|^2\, \ell_R^3)$ hence in $O(\ell_R^5)$ for computing $[\![\, R^\omega \,]\!]$ for any 2-grammar R over K. This is a greater complexity than for Theorem 3.1.

4 Computations on regular graphs

The family of regular graphs contains the pushdown graphs. Precisely the regular graphs of finite degree are the regular restrictions of pushdown graphs. They are the graphs generated by deterministic graph grammars allowing non-terminal hyperarcs of arity greater than 2. By splitting these hyperacs into arcs,

we provide a polynomial transformation of any graph grammar to a language-equivalent 2-grammar (Proposition 4.3). Then we can apply the previous results to obtain polynomial algorithms for solving the shortest path problem on regular graphs (Corollary 4.4).

We denote by a word $as_1 \ldots s_{\varrho(a)}$ a *hyperarc* of label a with an *arity* $\varrho(a) \geq 1$, linking in order the vertices $s_1, \ldots, s_{\varrho(a)}$. A hyperarc ast of arity $\varrho(a) = 2$ is an arc (s, a, t). A *hypergraph* G is a set of hyperarcs, and its *length* is $\ell_G := \sum \{\ |u| \mid u \in G\ \}$ for G finite.
A *graph grammar* R is a finite set of rules of the form:

$$A\,1 \ldots \varrho(A) \ \longrightarrow\ H$$

where $A\,1 \ldots \varrho(A)$ is a hyperarc labelled by A linking the vertices $1, \ldots, \varrho(A)$, and H is a finite hypergraph. Again the labels of the left hand sides form the set N_R of non-terminals, and the remaining labels in the right hand sides form the set T_R of terminals. As we only want to generate graphs, we assume that any terminal is of arity 2.
We extend in a natural way to any graph grammar R the notions of rewriting $\underset{R}{\longrightarrow}$ and of parallel rewriting $\underset{R}{\Longrightarrow}$.
Henceforth a graph grammar R is *deterministic*: two rules have distinct left hand sides, and like for 2-grammars, we define the graph $R^\omega(H)$ generated from any hypergraph H. We give below a graph grammar reduced to a unique rule, and its generated graph.

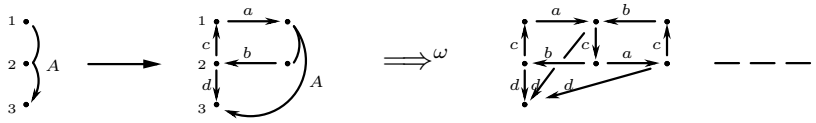


Figure 4.1 *Graph grammar and generated graph.*

A graph grammar R over K means that $T_R \subseteq K$.
For any graph grammar R and for any non-terminal $A \in N_R$,
its right hand side in R is $R(A\,1 \ldots \varrho(A))$, also denoted $R(A)$,
the graph generated by R from A is $R^\omega(A\,1 \ldots \varrho(A))$, also denoted $R^\omega(A)$,
$|R| = |N_R|$ is the number of rules of R (its cardinality),
$\ell_R := \sum \{\ |R(A)| + \varrho(A) \mid A \in N_R\ \}$ is the *length* of R,
$\varrho_R := \sum \{\ \varrho(A) \mid A \in N_R\ \}$ is the *arity* of R.
A *regular graph* is a graph generated by a graph grammar from a non-terminal.

We transform any graph grammar R into a language-equivalent 2-grammar by splitting any non-terminal hyperarc into all the possible arcs.
We assume that 0 is not a vertex of R and we take a new set of symbols

$$\{\ A_{i,j} \mid A \in N_R\ \wedge\ 1 \leq i, j \leq \varrho(A)\ \}.$$

We define the *splitting* $\prec G \succ$ of any $(T_R \cup N_R)$-hypergraph G by the graph:

$$\prec G\succ := \{\ s \xrightarrow{a} t \mid ast \in G \ \wedge\ a \in T_R\ \}$$
$$\cup\ \{\ s \xrightarrow{A_{i,j}} t \mid A \in N_R \ \wedge\ 1 \le i,j \le \varrho(A) \ \wedge$$
$$\exists\, s_1,\ldots,s_{\varrho(A)},\ As_1\ldots s_{\varrho(A)} \in H \ \wedge\ s = s_i \ \wedge\ t = s_j\ \}\,.$$

This allows us to define the splitting of R by the 2-grammar:

$$\prec R\succ := \{\ (1, A_{i,j}, 2) \longrightarrow h_{i,j}(R(A)_{i,j}) \mid A \in N_R \ \wedge\ 1 \le i,j \le \varrho(A)\ \}$$

where for $i \neq j$,

$$R(A)_{i,j} := \{\ s \xrightarrow[\prec R(A)\succ]{a} t \mid s \neq j \ \wedge\ t \neq i \ \wedge\ s,t \notin \{1,\ldots,\varrho(A)\} - \{i,j\}\ \}$$

with $h_{i,j}$ the vertex renaming of $R(A)_{i,j}$ defined by
$h_{i,j}(i) = 1\,,\ h_{i,j}(j) = 2\,,\ h_{i,j}(x) = x$ otherwise,

and
$$R(A)_{i,i} := \{\ s \xrightarrow[\prec R(A)\succ]{a} t \mid t \neq i \ \wedge\ s,t \notin \{1,\ldots,\varrho(A)\} - \{i\}\ \}$$
$$\cup\ \{\ s \xrightarrow{a} 0 \mid s \xrightarrow[\prec R(A)\succ]{a} i\ \}$$

with $h_{i,i}$ the vertex renaming of $R(A)_{i,i}$ defined by
$h_{i,i}(i) = 1\,,\ h_{i,i}(0) = 2\,,\ h_{i,i}(2) = 0\,,\ h_{i,i}(x) = x$ otherwise.

We then put $\prec R\succ$ into the reduced form of Section 2.



Figure 4.2 *Splitting in reduced form of the graph grammar of Figure 4.1.*

Note that $|\prec R\succ| \;=\; \sum_{A\in N_R} \varrho(A)^2 \;\le\; \varrho_R^{\,2}$
and $\ell_{\prec R\succ} \;\le\; \sum_{A\in N_R} \varrho(A)^2\,|R(A)|^2 \;\le\; \varrho_R^{\,2}\,\ell_R^{\,2}\,.$

So the splitting transformation is polynomial and is also language equivalent.

Proposition 4.3 *For any graph grammar R, any $(T_R \cup N_R)$-hypergraph H and any $s,t \in V_H$, we have* $L(R^\omega(H),s,t) \;=\; L(\prec R\succ^\omega(\prec H\succ),s,t)\,.$

Proposition 4.3 and Proposition 3.2 imply the well-known fact that the languages recognized (from and to a vertex) by regular graphs are exactly the context-free languages. But the judiciousness of Proposition 4.3 is that we can apply Theorem 3.1 and Corollary 3.5 to compute values of regular graphs. Let us extend Corollary 3.3.

Corollary 4.4 *For the semiring $(\mathbb{R} \cup \{-\omega,\omega\}, Min, +, \omega, 0)$, any graph grammar R, any left hand side X and any vertices $i,j \in V_X$, the shortest path problem $[\![\, R^\omega(X), i, j\,]\!]$ can be solved in $O(\varrho_R^{\,4}\,\ell_R^{\,4})$, and in $O(\varrho_R^{\,4}\,\ell_R^{\,2})$ when $R^\omega(X)$ is acyclic.*

Corollary 4.4 is just a particular path problem which can be solved with polynomial complexity by using Proposition 4.3 with Theorem 3.1 or Corollary 3.5. To summarize the shortest path problem from devices generating context-free languages, we got the complexity

$O(n^2)$ for any context-free grammar,
$O(n^2)$ for any 2-grammar generating an acyclic graph,
$O(n^4)$ for any 2-grammar,
$O(n^6)$ for any graph grammar generating an acyclic graph,
$O(n^8)$ for any graph grammar,
$O(n^{18})$ for any pushdown automaton.

These algorithms result from a pumping argument: Theorem 3.1.
Of course, the complexity of the shortest path problem for pushdown automata would be improved. In this paper, we focus on path algorithms starting from graph grammars over semirings. We thank an 'anonymous' referee for this conclusion.
Note that the semiring approach cannot be used for computing the throughput value of cf-languages; a polynomial solution to this particular problem has been given in [2].

Many thanks to Antoine Meyer for his help in the writing of this paper.

References

[1] D. Caucal *On the regular structure of prefix rewriting*, Theoretical Computer Science 106, 61–86 (1992).

[2] D. Caucal, J. Czyzowicz, W. Fraczak and W. Rytter *Efficient computation of throughput values of context-free languages*, 12^{th} CIAA, to appear in LNCS (2007).

[3] B. Courcelle *Infinite graphs of bounded width*, Mathematical Systems Theory 21-4, 187–221 (1989).

[4] J. Esparza, S. Kiefer and M. Luttenberger *On fixed point equations over commutative semirings*, 24^{th} STACS, LNCS 4393, W. Thomas, P. Weil (Eds.), 296–307 (2007).

[5] M. Hopkins and D. Kozen *Parikh's theorem in commutative Kleene algebra*, 14^{th} LICS, IEEE, G. Longo (Ed.), 394–401 (1999).

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

[7] R. Parikh *On context-free languages*, JACM 13-4, 570–581 (1966).

[8] D. Pilling *Commutative regular equations and Parikh's theorem*, J. London Math Soc 6-4, 663–666 (1973).

[9] W. Thomas *A short introduction to infinite automata*, 5^{th} DLT 01, LNCS 2295, W. Kuich, G. Rozenberg, A. Salomaa (Eds.), 130–144 (2001).