Definitions and Comparisons of Local Computations on Graphs

(extended abstract)

Igor Litovsky* Yves Métivier* Eric Sopena*

Laboratoire Bordelais de Recherche en Informatique
Unité associée C.N.R.S. 1304
351, cours de la Libération
F-33405 TALENCE

Abstract. We are interested in models to encode and to prove decentralized and distributed computations on graphs or networks. In this paper, we define and compare six models of graph rewriting systems. These systems do not change the underlying structure of the graph on which they work, but only the labelling of its components (edges or vertices). Each rewriting step is fully determinated by the knowledge of a fixed size subgraph, the local context of the rewritten occurrence. The studied families are based on the rewriting of partial or induced subgraphs and we use two kinds of mechanisms to locally control the applicability of rules : a priority relation on the set of rules or a set of forbidden contexts associated with each rule. We show that these two basic (i.e. without local control) families of graph rewriting systems are distinct, but whenever we consider the local controls of the rewriting, the four so-obtained families are equivalent.

1 Introduction

We are interested in models to encode and to prove decentralized and distributed computations on graphs or networks. The presented models are graph rewriting systems satisfying the following constraints which seem to be natural when describing distributed computations with a decentralized control:

($C1$) they do not change the underlying graph (i.e. the network) but only the labelling of its components (edges and/or vertices), the final labelling being the result of the computation,

($C2$) they are *local*, that is, each rewriting step changes only the labelling of a fixed size connected subgraph of the underlying graph,

*With the support of the PRC Mathématiques et Informatique and the European Basic Research Action ESPRIT No 3166 ASMICS.

($C3$) they are *locally generated*, that is, the application condition of the rewriting only depends on the *local context* of the rewritten subgraph.

For such systems, the distributed aspect comes from the fact that several rewriting steps can be performed simultaneously on "far enough" subgraphs.

In this paper, we define and compare six types of graph rewriting systems. Any such system, say $\mathcal{R}$, is defined by a finite set of rewriting rules (and thus uses a finite set of labels) and may be equipped with a mechanism which locally controls the rewriting rules application. A rewriting rule r consists in the relabelling of a fixed connected subgraph G_r, and is given as $r : (G_r, \lambda) \longrightarrow (G_r, \lambda')$.

We say that a labelled graph (G, l) is rewritten by $\mathcal{R}$ in (G, l') if there exists a finite sequence of *allowed applications* (in a sense precised below) of relabellings in $\mathcal{R}$ leading from (G, l) to (G, l'). Given a graph rewriting system $\mathcal{R}$, we are interested in the function $Irred_{\mathcal{R}}$ which, with each graph (G, λ), associates the set of *irreducible* graphs (i.e. where no allowed application of rule is possible) obtained from (G, λ). We say that two graph rewriting systems $\mathcal{R}$ and $\mathcal{R}'$ are *equivalent* when $Irred_{\mathcal{R}} = Irred_{\mathcal{R}'}$. A family $\mathcal{F}_1$ of graph rewriting systems is *less powerfull* than a family $\mathcal{F}_2$ if every graph rewriting system in $\mathcal{F}_1$ is equivalent to a graph rewriting system in $\mathcal{F}_2$. The families $\mathcal{F}_1$ and $\mathcal{F}_2$ are equivalent if each one is less powerfull than the other one.

We now present the six various types of graph rewriting systems we will consider in this paper. For each of them we have to specify the notion of *allowed application* of a rule r in a graph (G, l). The first criterium characterizing the applicability of a rule is given by the definition of the *occurrence* of the left-hand side (G_r, λ) in (G, l). Such an occurrence may be :

- a partial subgraph of (G, l) isomorphic to (G_r, λ),
- an induced subgraph of (G, l) isomorphic to (G_r, λ).

Hence, we respectively obtain the families of $pGRS$'s and $iGRS$'s. We prove that the family of $pGRS$'s is strictly less powerfull than the family of $iGRS$'s.

On the other hand, to increase the computational power of these basic graph rewriting systems, we use two kinds of local control on the applicability of rules :

- The first one has been introduced in [2] and consists in adding a partial order relation, called *priority*, on the set of rewriting rules. In such systems, the application of a rule r is *allowed* on an occurrence θ of (G_r, λ) if no rule with a greater priority has an occurrence overlapping θ. Note that the effect of these priorities is strictly local (constraint ($C3$) is respected).

- The second one, inspired from [3], consists in adding to each rewriting rule r a set of *forbidden contexts*, where a context is a graph having (G_r, λ) as a subgraph. For such systems, an application of r is *allowed* on an occurrence θ of (G_r, λ) if θ is not a subgraph of a forbidden context in (G, l).

These rewriting systems are respectively called $PxGRS$'s and $FCxGRS$'s (for $x \in \{p, i\}$). The so-defined families are strictly more powerfull than the previous ones. It is easy to see that the family of $FCxGRS$'s is more powerfull than the family

of $PxGRS$'s (for $x \in \{p, i\}$). The main part of this paper is devoted to proving the equivalence of the $FCpGRS$'s and the $PpGRS$'s. This result is not immediate : for example, it is easy to give a one-rule $FCpGRS$ recognizing the class of complete graphs, but no "simple" $PpGRS$ can do it. We also prove that with such a local control Y (Priority or Forbidden Contexts), the $YpGRS$'s and the $YiGRS$'s are equivalent. Hence, we have :

$$pGRS \subsetneq iGRS \subsetneq \{PpGRS = FCpGRS = PiGRS = FCiGRS\}$$

When proving that every $FCpGRS$ is equivalent with a $PpGRS$, the main difficulty comes from the fact that a $FCpGRS$ forbids the application of a rewriting rule by only considering the forbidden contexts associated with this rule, since a $PpGRS$ only forbids such an application when another rule (with a greater priority) is applicable on an *overlapping* occurrence. Assuming first that one works on graphs having a distinguished vertex, depth-first traversals can be sequentially processed by using a $PpGRS$ [2]. In this case, every $FCpGRS$ can be simulated by a $PpGRS$ in the following way : each depth-first traversal looks for applying a fc-rule ; when it founds one or more such rules, it "chooses" one of them and applies it ; when no fc-rule is applicable, the $PpGRS$ stops (see $\mathcal{R}_{locsim}$ in Section 3.3). But it is known that the problem of distinguishing one vertex (known as the *election problem*) is not solvable for any type of graph (see [1]). Hence, the main idea of this paper is to construct, using a $PpGRS$, a partition of the graph into subgraphs (called *countries*) of k-bounded diameter, each country having an elected vertex (the *capital*). This "k-election" mechanism, used together with the $PpGRS$ $\mathcal{R}_{locsim}$, enables us to simulate every $FCpGRS$ by a $PpGRS$ (Proposition 3.4).

By using techniques inspired from [1], power and limitations of such local computations on graphs are studied in [7].

Complete proofs of the results presented here can be found in [6].

2 Definitions and Notation

2.1 Graphs

A simple, loopless, undirected *graph* G is defined as a pair $(v(G), e(G))$ where $v(G)$ is a finite set of vertices and $e(G)$ a set of edges, an edge being a set of two distinct vertices in $v(G)$. A *labelled graph* is a pair (G, λ) where G is a graph and λ is a mapping from $v(G) \cup e(G)$ into a finite set of labels $\mathcal{L}$. Let (G, λ) and (G', λ') be two labelled graphs. The labelled graph (G', λ') is a *(partial) subgraph* of (G, λ) if:

$$\begin{cases} v(G') \subseteq v(G) \\ e(G') \subseteq e(G) \\ \lambda|_{G'} = \lambda' \end{cases}$$

where $\lambda|_{G'}$ denotes the mapping induced from λ by $v(G') \cup e(G')$.

The pair $((v(G) \setminus v(G'), e(G) \setminus e(G')), \lambda)$ is called the *context* of (G', λ') in (G, λ). It is denoted by $(G, \lambda) \setminus (G', \lambda')$. A mapping φ from v(G) into v(G') is an homo-

morphism from (G, λ) into (G', λ') if for any x, y in $v(G)$, we have :

$$\begin{cases} \{x, y\} \in e(G) \Longrightarrow \{\varphi(x), \varphi(y)\} \in e(G') \\ \lambda(x) = \lambda'(\varphi(x)) \\ \lambda(\{x, y\}) = \lambda'(\{\varphi(x), \varphi(y)\}) \end{cases}$$

Let φ be an homomorphism from (G, λ) into (G', λ'), we will denote by $\varphi(G)$ the graph $(\varphi(v(G)), \mathcal{E})$ where $\{\varphi(x), \varphi(y)\} \in \mathcal{E}$ iff $\{x, y\} \in e(G)$. So $(\varphi(G), \lambda')$ is a subgraph of (G', λ'). Whenever φ is injective, φ is said to be an *occurrence* of (G, λ) in (G', λ'). If moreover φ is bijective, (G, λ) and (G', λ') are said to be *isomorphic.*

Let (G', λ') be a subgraph of (G, λ). We say that (G', λ') is an *induced* subgraph of (G, λ) iff for all x, y in $v(G')$, $\{x, y\} \in e(G) \Longleftrightarrow \{x, y\} \in e(G')$. An occurrence θ of (G'', λ'') in (G, λ) is said to be an *induced occurrence* if $(\theta(G''), \lambda'')$ is an induced subgraph of (G, λ).

Let r be an integer and x be a vertex in $v(G)$; the *ball subgraph* $B(x, r)$ is the induced subgraph of (G, λ) whose vertices are all the vertices in $v(G)$ whose distance to vertex x is at most r.

From now on, as we will only deal with connected labelled graphs, we will simply say *graph* for connected labelled graphs.

2.2 Rewriting on Partial or Induced Subgraphs

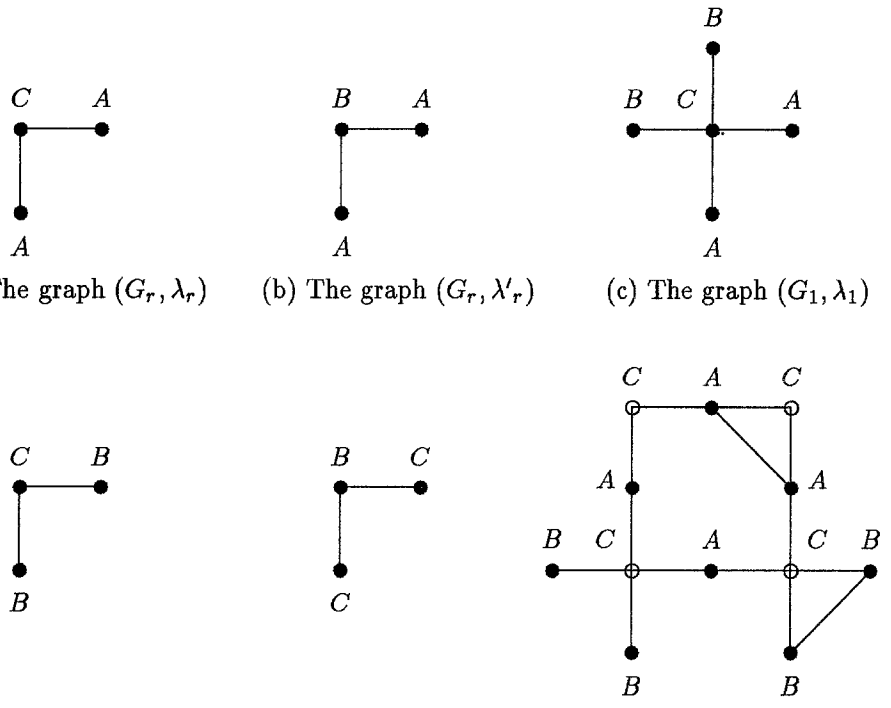
A *partial-graph rewriting rule* is a pair $r = \big((G_r, \lambda_r), (G_r, \lambda'_r)\big)$, denoted $(G_r, \lambda_r, \lambda'_r)$ for short. (G_r, λ_r) (resp. (G_r, λ'_r)) is called the *left-hand side* (resp. *right-hand side*) of the rule r. The rewriting relation $\xrightarrow[r]{p}$ is defined by : $(G, \lambda) \xrightarrow[r]{p} (G, \lambda')$ if there exists an occurrence θ of (G_r, λ_r) in (G, λ) such that θ is an occurrence of (G_r, λ'_r) in (G, λ') and the contexts of $\theta(G_r)$ in (G, λ) and in (G, λ') are identical. We say that θ is the *rewritten* occurrence.

A *partial-Graph Rewriting System (pGRS)* is a finite set $\mathcal{R}$ of rewriting rules. The rewriting relation $\xrightarrow[\mathcal{R}]{p}$ is defined by : $(G, \lambda) \xrightarrow[\mathcal{R}]{p} (G, \lambda')$ if and only if there exists a rule $r \in \mathcal{R}$ such that $(G, \lambda) \xrightarrow[r]{p} (G, \lambda')$.

A *partial-Graph Rewriting System with Priorities (PpGRS)* is a finite set $\mathcal{R}$ of rewriting rules equipped with a partial ordering relation $>$ called *priority* which works as follows : let θ be an occurrence of a rule $r \in \mathcal{R}$. The rule r is *applicable* on θ if there is no occurrence θ' of a rule $r' > r$ such that $v(\theta(G_r)) \cap v(\theta'(G_{r'})) \neq \emptyset$. If two or more rules are simultaneously applicable on (G, λ), one of them (randomly chosen) is applied. We note $(G, \lambda) \xrightarrow[\mathcal{R}]{p} (G, \lambda')$ if there exists a rule $r \in R$ such that $(G, \lambda) \xrightarrow[r]{p} (G, \lambda')$ and r was applicable in (G, λ) on the rewritten occurrence.

A *partial-graph rewriting rule with forbidden contexts (fc-rule for short)* is a pair $(r, \mathcal{H})$ where r is a rewriting rule $(G_r, \lambda_r, \lambda'_r)$ and $\mathcal{H}$ is a finite family of pairs $\{((G_i, \lambda_i), \theta_i)\}_{i \in I_r}$ where (G_i, λ_i) is a labelled graph and θ_i is an occurrence of (G_r, λ) in (G_i, λ_i). The contexts $(G_i \setminus \theta_i(G_r))$ are the *forbidden contexts* of the fc-rule and are used as follows : let θ be an occurrence of G_r ; the fc-rule $(r, \mathcal{H})$ is *applicable* on θ if for no i, there exists an occurrence φ of (G_i, λ_i) in (G, λ) such that $\varphi\theta_i = \theta$.

A *partial-Graph Rewriting System with Forbidden Context (FCpGRS)* is a finite set $\mathcal{R}$ of fc-rules. We note $(G, \lambda) \xrightarrow[\mathcal{R}]{p} (G, \lambda')$ if there exists a rule $r \in \mathcal{R}$ such that $(G, \lambda) \xrightarrow[r]{p} (G, \lambda')$ and r was applicable in (G, λ) on the rewritten occurrence.



(a) The graph (G_r, λ_r) (b) The graph (G_r, λ'_r) (c) The graph (G_1, λ_1)

(d) The graph (G_s, λ_s) (e) The graph (G_s, λ'_s) (f) The graph (G, λ)

Figure 1: Applicability of rewriting rules.

The same notions can be defined by using induced subgraphs instead of partial ones, leading respectively to *i-, Pi- and FCiGRS.*

Example 2.1 Consider the graph (G, λ) of Figure 1(f) and the graph rewriting rules $r = (G_r, \lambda_r, \lambda'_r)$, $s = (G_s, \lambda_s, \lambda'_s)$, where (G_r, λ_r), (G_r, λ'_r), (G_s, λ_s) and (G_s, λ'_s) are given by Figure 1(a,b,d,e) respectively.

- As rule of a *pGRS*, r can be applied on the four corners of graph (G, λ) (vertices marked as o).
- As rule of a *iGRS*, r can be applied on each corner of graph (G, λ) except on the upper-right one, since there is a forbidden edge linking two vertices of the occurrence.
- As rule of a *FCpGRS*, with graph (G_1, λ_1) of Figure 1(c) as forbidden context, r can only be applied on the two upper corners of graph (G, λ).
- As rule of a *FCiGRS*, r can be applied on the upper-left corner of (G, λ) and on its bottom-right corner, since the forbidden context of r does not appear as an induced subgraph.
- As rule of a *PpGRS*, with $s > r$, rule r can only be applied on the two upper corners of (G, λ).
- As rule of a *PiGRS*, with $s > r$, rule r can be applied on the upper left corner, and the bottom-right corner of (G, λ) (since s cannot be applied on occurrences overlapping these corners).

2.3 Rewriting System Behaviour

In this paper, we will only consider *noetherian* graph rewriting systems, which means that from any graph, there exists no infinite rewriting chain.

Given a graph rewriting system $\mathcal{R}$, we consider the reflexive transitive closure $\xrightarrow[\mathcal{R}]{*}$ of $\xrightarrow[\mathcal{R}]{p}$. A graph (G,λ') is said to be *irreducible* with respect to $\mathcal{R}$ if there is no applicable rule of $\mathcal{R}$ on (G,λ'). For every graph (G,λ), we note $Irred_{\mathcal{R}}((G,\lambda))$ the set $\{(G,\lambda')/\ (G,\lambda) \xrightarrow[\mathcal{R}]{*} (G,\lambda')$and no rule of $\mathcal{R}$ is applicable on $(G,\lambda')\}$. Let $\mathcal{R}$ and $\mathcal{R}'$ be two rewriting systems. The systems $\mathcal{R}$ and $\mathcal{R}'$ are said to be *equivalent* if for any graph (G,λ), we have $Irred_{\mathcal{R}}((G,\lambda)) = Irred_{\mathcal{R}'}((G,\lambda))$. We will say that a family $\mathcal{F}_1$ of graph rewriting systems is *less powerfull* than a family $\mathcal{F}_2$, if every graph rewriting system in $\mathcal{F}_1$ is equivalent to a graph rewriting system in $\mathcal{F}_2$. The families $\mathcal{F}_1$ and $\mathcal{F}_2$ are *equivalent* if each one is less powerfull than the other one.

3 Preliminary Results

3.1 The k-election Problem

Let k be a given integer. The *k-election problem* on a graph can be intuitively introduced as follows. Each vertex of the graph stands for a town, each edge for a road segment joining to distinct towns. Initially, each town has a *neutral* status. We want to organize the graph by delimiting *countries*, each country having one *capital*. In each country, the minimal distance between any town and the capital must be at most k. Moreover, the distance between two any capitals in the graph must be not less than $k+1$. Each capital (resp. each town) has also to know the towns (resp. the capital) of its country.

The *PpGRS* which solves this problem will fill two additionnal requirements : the towns in all the countries will be classified according to their minimal distance to the capital and any town will belong to one of the countries whose capital is the neariest. This will be done by constructing a spanning forest of the initial graph, each tree (standing for a country) being rooted at a capital.

To solve this problem, we consider the *PpGRS* $\mathcal{R}_{k-elec} = <C, P>$, with the set of labels $C = \{N, C, T, T_1, ..., T_k, \times\}$ where N stands for *Neutral*, C for *Capital*, T for a town belonging to a country (but not yet classified), T_i $(1 \leq i \leq k)$ for a classified town, and $\times$ for marked edges (i.e. edges belonging to the spanning forest).

The set of rules P is given in Figure 2. Labels L_i stand for any vertex label. Except when it is explicitly specified (rules $R_4(i)$), any edge can be marked or unmarked, and is preserved in the right-hand side of any rule.

The priorities are given as :

$$R_1 < R_4(k) < R_4(k-1) < \cdots < R_4(1) < \left\{R_3(i)\right\}_{k<i\leq 2k} < \left\{R_2(i)\right\}_{1\leq i\leq k}$$

Rule R_1 says that any neutral town can spontaneously become a capital, except if there already exists a capital in its k-neighbourhood (rules $R_2(i)$ prevent rule R_1 to be applied).

Rules $R_2(i)$ are intended to mark any town in the k-neighbourhood of a capital as a town belonging to a country (label T). The classification of these towns will be

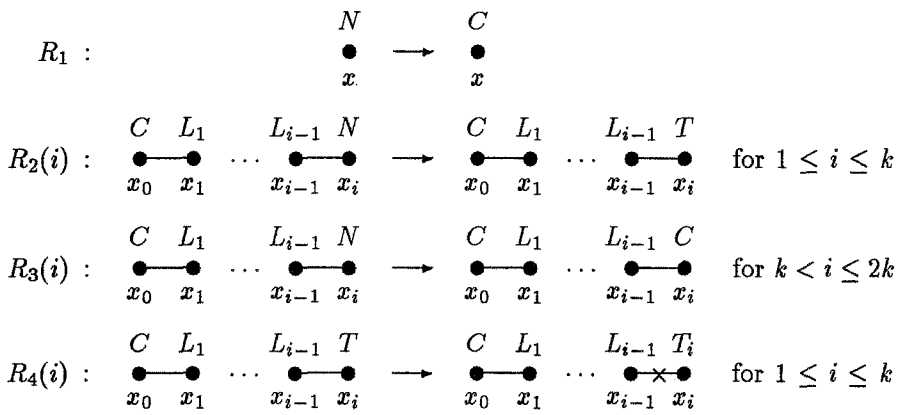


Figure 2: The PGRS $\mathcal{R}_{k-elec}$.

done later, by using rules $R_4(i)$.

Rules $R_3(i)$ have been essentially introduced for technical purposes. Thanks to them, a capital will only begin to classify its towns when the capitals of its neighbouring countries are elected. This will ensure a good classification of the towns in a country. Figure 3 shows what kind of problem would arise if we don't use such rules : let $k = 2$ and suppose vertex x_0 becomes capital. But for rules $R_3(i)$, it can mark vertices x_1 and x_2 as classified towns (see figure 3(a)). Then the system terminates by electing, say x_3, as a capital and x_4 remains a neutral town. Figure 3(b) shows a good terminal configuration, obtained by using rules $R_3(i)$.

Rules $R_4(i)$ implement the classification of the towns. Their priority ordonnancement ensures that each town will be labelled according to its distance to one of the nearest capital (a T_i-label means that this town is at distance i from one of the nearest capital).

The following result states that the *PpGRS* $\mathcal{R}_{k-elec}$ solves the k-election problem.

(a) A bad intermediate configuration (b) A good terminal configuration

Figure 3: The classification problem.

Theorem 3.1 *The PpGRS $\mathcal{R}_{k-elec}$ is noetherian. Moreover, let (G, λ) be a connected graph whith all vertices N-labelled, then, for any graph (G, λ') in $Irred((G, \lambda))$, the subgraph induced by the marked edges is a spanning forest of (G, λ') with all vertices labelled C, T_1, ..., T_{k-1} or T_k and such that :*

- *Every tree has only one C-labelled vertex,*
- *The distance between two C-labelled vertices is at least $k+1$,*
- *Every T_i-labelled vertex x satisfies $d(x, \lambda'^{-1}(C)) = i$.*

3.2 The k-enumeration Problem

Let G be a graph and k, r be two integers. We can construct a *PpGRS* which enables us to enumerate all the ordered k-tuples of vertices in a subgraph of G with radius r (more precisely in a ball $B(x, r)$ for a given vertex x in G). This *PpGRS* will act on a spanning tree $T(x)$ of the graph G, rooted at vertex x.

Let us intuitively describe the behaviour of such a *PpGRS* : vertex x will be the controler of the computation. It first looks for a vertex, say v_1, which can be chosen as the first component of a new k-tuple. When such a vertex is found, it looks for a second one and so one. When for i given vertices $v_1, \ldots, v_i$ the system has enumerated all the k-tuples having these vertices as first (ordered) components, vertex v_i is marked as having been the i^{th} component of all such k-tuples and a new i^{th} component is looked for.

All these steps will be handled by alternating depth-first traversals : vertex x initiates a traversal which looks for a given vertex ; when the control returns to it, it initiates a new traversal for the next search. This computation terminates when all the vertices in the graph are marked as having been the first component of all possible k-tuples. Then, we obtain :

Proposition 3.2 *Let k and r be two integers. There exists a PpGRS which, given a graph (G, λ) and any vertex $x \in \boldsymbol{v}(G)$, enumerates all k-tuples of the ball $B(x, r)$ in (G, λ).*

3.3 Local Simulation of a *FCpGRS* by a *PpGRS*

Let $\mathcal{R}$ be a *FCpGRS*. We can construct a *PpGRS* $\mathcal{R}_{locsim}$ which, on a given country with capital c and rooted spanning tree $T(c)$ (see 3.1) realizes a random application on $T(c)$ of one applicable fc-rule of $\mathcal{R}$ when such a rule exists. The main idea is the following : given a fc-rule $((G_r, \lambda_r, \lambda'_r), \mathcal{H})$, we can traverse any vertex x in $T(c)$ and, by enumerating the ad-hoc k-tuples in the ad-hoc sized balls $B(x, d)$, say whether the rule r is applicable or not on an occurrence $\theta(G_r)$ containing vertex x, and then apply it or not (the system will ensure that whenever there are applicable rules, one of them is applied). Then we get :

Proposition 3.3 *Let $\mathcal{R}$ be a FCpGRS. There exists a PpGRS $\mathcal{R}_{locsim}$ which, from any given rooted tree $T(c)$ included in a given graph (G, λ), can test whether a fc-rule $r \in \mathcal{R}$ is applicable in (G, λ) on an occurence θ of G_r such that $\theta(G_r) \cap T(c) \neq \emptyset$, or not. Furthermore, a random application of such an applicable fc-rule (when it exists) is done.*

3.4 Simulating the Activity of a *FCpGRS* by Using a *PpGRS*

To achieve the simulation of a given *FCpGRS* $\mathcal{R}$ by a *PpGRS*, we need a *PpGRS* $\mathcal{R}_{actsim}$ which supervises the *activity* of capitals. A capital x is said to be *active* if a previous *PpGRS* $\mathcal{R}_{locsim}$ is looking for applying a fc-rule in the country of the capital x. When such a *PpGRS* is acting, it works on balls centered at vertices of its country. But such balls may also contain vertices of near countries. Hence, we have to ensure that two near capitals will never be simustaneously active.

Let k be the greatest diameter of a left-hand side graph in the definition of $\mathcal{R}$. Given a graph (G, λ), we consider the graph $Cap(G)$ whose vertices are the capitals obtained via a k-election in (G, λ), and whose edges are linking two capitals when these two capitals are near (i.e. whose distance is at most $3k$). We can construct a *PpGRS* $\mathcal{R}_{cap}$ which simulates, in $Cap(G)$, the *activity* on (G, λ) of every execution of the given *FCpGRS* $\mathcal{R}$. More precisely, let us assume that we have a graph (G, λ) where a k-election has been made. An execution of $\mathcal{R}$ on G is defined by the sequence of rewritten occurences $\theta_1, \ldots, \theta_n$ in G. Each occurence θ_i intersects one or more countries with capitals $C_{i,1}, \ldots, C_{i,j_i}$ respectively. The goal of the *PpGRS* $\mathcal{R}_{cap}$ is to ensure that first, one of the capitals $C_{1,1}, \ldots, C_{1,j_1}$ is active (step_1), next one of the capitals $C_{2,1}, \ldots, C_{2,j_2}$ is active (step_2),..., next one of the capitals $C_{n,1}, \ldots, C_{n,j_n}$ (step_n) is active, and finally every capital must be unactive. Thus, at each step_i $(1 \leq i \leq n)$, a *PpGRS* $\mathcal{R}_{locsim}$ may simulate the rewriting of $\mathcal{R}$ on θ_i $(1 \leq i \leq n)$. To obtain a *PpGRS* no longer working on $Cap(G)$, but on the whole graph (G, λ), it is sufficient to consider that the edges are in fact paths of length at most $3k$.

Proposition 3.4 *Let $\mathcal{R}$ be a FCpGRS. There exists a PpGRS which, for any given graph (G, λ), can simulate on $Cap(G)$ the activity of any execution of the given FCpGRS $\mathcal{R}$.*

4 Comparisons Between the Classes of Rewriting Systems

We are now going to show that *PpGRS's* and *FCpGRS's* are in fact equivalent. For any *FCpGRS* $\mathcal{R}$, we can construct a *PpGRS* $\mathcal{R}$' which can simulate the behaviour of $\mathcal{R}$. The intuitive idea is the following : let k (resp. k') be the greatest diameter (resp. cardinality) of a left-hand side graph in the definition of $\mathcal{R}$; we first construct a covering of the graph by means of countries and capitals (two capitals being at distance at least $k+1$ from each other). Then, any capital can test whether a fc-rule can be applied on an occurrence overlapping its country or not, by enumerating the k'-tuples in its neighbourhood, and apply one of these rules when possible. The whole activity of the capitals is managed by the *PpGRS* seen in previous section.

Conversely, for any *PpGRS* $\mathcal{R}$, one can easily construct a *FCpGRS* $\mathcal{R}'$ which simulates the behaviour of $\mathcal{R}$. For any rule r in $\mathcal{R}$, one can characterize the contexts which have to prevent the application of r : it suffices to take into account all the rules r' in $\mathcal{R}$ which are more prioritary than r and which can overlap an occurrence of G_r.

Hence, we obtain the main result :

Theorem 4.1 *The PpGRS's and the FCpGRS's are equivalent.*

Finally, we can prove (see [6]) the following relationships between the different classes of rewriting systems.

Theorem 4.2 *The class of pGRS's is less powerfull than the class of iGRS's. The class of iGRS's is less powerfull than the class of FCpGRS's. The classes of PpGRS's, FCpGRS's, PiGRS's and FCiGRS's are equivalent.*

References

[1] D. Angluin, *Local and global properties in networks of processors*, Proceedings of the 12^{th} Symposium on Theory of Computing (1980), 82-93.

[2] M. Billaud, P. Lafon, Y. Métivier and E. Sopena, *Graph Rewriting Systems with Priorities*, Lecture Notes in Computer Science **411** (1989), 94-106.

[3] B. Courcelle, *Recognizable sets of unrooted trees* in *Definability and Recognizability of sets of trees*, Elsevier, to appear (1991).

[4] I. Litovsky and Y. Métivier, *Computing with Graph Rewriting Systems with Priorities*, Fourth International Worshop on Graph Grammars and their Applications to Computer Science, Bremen. Lecture Notes in Computer Science **532** (1991), 549-563.

[5] I. Litovsky and Y. Métivier, *Computing trees with graph rewriting systems with priorities*, in *Definability and Recognizability of sets of trees*, Elsevier, to appear (1991).

[6] I. Litovsky, Y. Métivier, E. Sopena, *Definitions and comparisons of local computations on graphs*, Internal Report No **91-43**, LaBRI, Université Bordeaux I (1991).

[7] I. Litovsky, Y. Métivier and W. Zielonka *The power and limitations of local computations on graphs and networks*, Internal Report No **91-31**, LaBRI, Université Bordeaux I, submitted to publication (1991).