LaBRI Research Report No. Compiled on April 18, 2001
The final version of this paper has been published in *Handbook of graph grammars and computing by graph transformation*, vol. III, World Scientific, H. Ehrig, H.-J. Kreowski, U. Montanari and G. Rozenberg eds (1999), 1–56.

GRAPH RELABELLING SYSTEMS AND DISTRIBUTED ALGORITHMS

Igor Litovsky,
ESSI, Sophia Antipolis, BP 132, 06561 Valbonne CEDEX, France

Yves Métivier and Éric Sopena
LaBRI, Université Bordeaux I - ENSERB, 351 Cours de la Libération, 33405 Talence, France

Abstract. Graph relabelling systems have been introduced as a suitable model for expressing and studying distributed algorithms on a network of communicating processors. We recall the basic ideas underlying that model and we present the main questions that have been considered and the main results that have been obtained in that framework.

Keywords. Models for distributed systems, Graph relabelling systems, Distributed algorithms, Local computations, Coverings, Election problem, Termination detection.

1 Introduction

Graph relabelling systems and, more generally, local computations in graphs are powerful models which provide general tools for encoding distributed algorithms, for proving their correctness and for understanding their power. We consider a network of processors with arbitrary topology. It is represented as a connected, undirected graph where vertices denote processors, and edges denote direct communication links. An algorithm is encoded by means of local relabelings. Labels attached to vertices and edges are modified *locally*, that is on a subgraph of fixed radius k of the given graph, according to certain rules depending on the subgraph only ($k-$*local computations*). The relabelling is performed until no more transformation is possible. The corresponding configuration is said to be in normal form. Two sequential relabelling steps are said to be *independent* if they are applied on disjoint subgraphs. In this case they may be applied in any order or even concurrently.

The present contribution reflects classical topics including basic properties of local computations. Among paradigms associated with local computations, we present the election problem, the recognition problem and the local detection of the termination problem.

For these three problems, we consider graphs which are uniformly labelled by some initial label (which may encode some knowledge on the graph as, for instance, the number of vertices and/or edges). For the recognition problem, the presence or the absence of certain final labels determines whether G is accepted or not. The aim of an election algorithm is to choose exactly one element among the set of vertices. The more general assumption used in the paper is as follows. We suppose that initially every vertex and every edge has the same label which encodes some knowledge on the graph. Then we consider local computation systems such that for each irreducible graph there is a given vertex label which appears exactly once in the graph. A distributed algorithm terminates whenever it reaches a terminal configuration, that is a configuration in which no step of the algorithm can be applied. We are interested in the question whether the global termination of a system of local computations can be detected also locally. This means that for every terminal configuration there is a vertex in the graph such that its neighbourhood of a given radius r determines that a normal form has been reached. In this case, we say that global termination is $r-$*locally detected*. We use coverings as a fundamental tool which enable to understand the borderline between positive and negative results about distributed computations. We are particularly interested in the question whether certain additional knowledge about the network, which is used in specific distributed algorithms, is really needed for solving the given problem or not. For lack of space, some results about the comparison with the power of logic formulas will not be given, see [25, 31].

Among models related to our model there are local computations systems, as defined by Rosensthiel *et al.* [1], Angluin [2] or Yamashita and Kameda [3, 4]. In [1] a synchronous model is considered, where all vertices are equipped with a deterministic finite automata (the same for all vertices). A basic computation step consists then in computing the next state of each processor according to its own state and to the states of its neighbours. In [2] an asynchronous model is considered: during a basic computation step, two adjacent vertices exchange their labels and then compute new labels. In [3, 4] an asynchronous model is also considered where during a basic computation step a processor either changes its state and sends a message or receives a message.

Introduction to distributed algorithms and main topics of the field are presented in [5, 6].

2 Graphs

All graphs we consider are finite, undirected, with no multiple edges nor self-loops. A graph G is thus a pair $(V(G), E(G))$, where $V(G)$ is a finite set of vertices and $E(G) \subseteq \{\{v, v'\} \mid v, v' \in V(G),\ v' \neq v\}$ is the set of edges. The number of vertices in a graph G is called the *size* of G.

Let $e = \{v, v'\}$ be an edge; we say that e is *incident* with v and v' and that v' is a *neighbour* of v. The set of neighbours of a vertex v, together with v itself, is denoted $N_G(v)$. Two edges are *adjacent* if they share a common vertex. The *degree* of a vertex v, denoted by $d(v)$, is the number of edges incident with v. Vertices of degree 1 are called *leafs*, other vertices are *internal vertices*. A *path* P from v_1 to v_i in G is a sequence $P = v_1, e_1, v_2, e_2, \ldots, e_{i-1}, v_i$ of alternating vertices and edges such that for every j, $1 \leq j < i$, e_j is an edge incident with vertices v_j and v_{j+1}; $i-1$ is the *length* of P. If $v_1 = v_i$ then P is a *cycle*. A path P is *simple* if no vertex occurs twice in P. Two vertices v and w are *connected* if there exists a path from v to w. A graph is connected if every two vertices are connected. Let v and v' be two connected vertices; the *distance* between v and v', denoted by $d(v, v')$, is the minimum length of a (simple) path from v to v'. The maximal distance $d(v, v')$, taken over all pairs of vertices $\{v, v'\}$ of a connected graph G, is the *diameter* of G and is denoted by $D(G)$. A tree is a connected graph containing no cycle. In a tree, every two vertices are thus connected by precisely one simple path.

Let G and G' be two graphs; G' is a *subgraph* of G if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. Let V' be a subset of $V(G)$; the subgraph of G *induced* by V', denoted by $G[V']$, has vertex set V' and contains all edges of G whose both endpoints belong to V'. Let v be a vertex and k a positive integer; the *ball* of radius k with center v, denoted by $B_G(v, k)$, is the subgraph of G induced by the set of vertices $V' = \{v' \in V \mid d(v, v') \leq k\}$. A *homomorphism* of a graph G to a graph H is a mapping $\gamma : V(G) \longrightarrow V(H)$ such that if $\{u, v\}$ is an edge of G then $\{\gamma(u), \gamma(v)\}$ is an edge of H. Since we deal with graphs having no self-loop, we necessarily have $\gamma(u) \neq \gamma(v)$ if $\{u, v\}$ is an edge of G. Note also that $\gamma(N_G(u)) \subseteq N_H(\gamma(u))$ for every vertex u. We say that γ is an *isomorphism* if γ is bijective and γ^{-1} is also a homomorphism. Two graphs G and H are *isomorphic*, denoted by $G \simeq H$, if there exists an isomorphism from G to H. A *class* of graphs is any collection of graphs closed under the isomorphism relation.

Notation. For every integers i and j, $i \leq j$, we shall denote by $[i, j]$ the set $\{i, i+1, \ldots, j\}$.

Remark 1 *In the following, we will only consider connected graphs and will simply call them "graphs".*

3 First Examples

The aim of this section is to illustrate, in an intuitive way, the notion of graph relabelling systems by showing how some algorithms on networks of processors may be encoded within our framework [7].

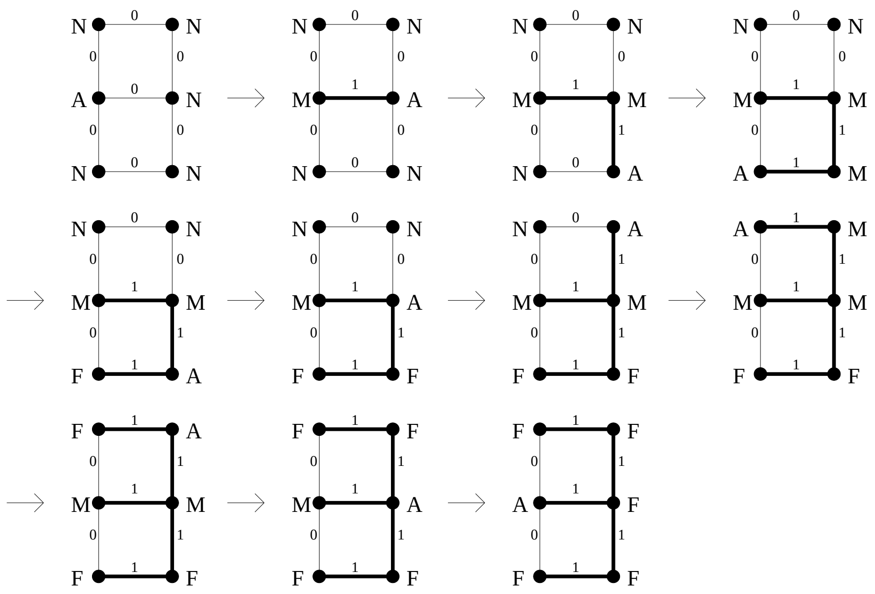


Figure 1: Sequential computation of a spanning tree

As usual, such a network is represented by a graph whose vertices stand for processors and edges for (bidirectional) links between processors. At every time, each vertex and each edge is in some particular state and this state will be encoded by a vertex or edge label. According to its own state and to the states of its neighbours, each vertex may decide to realize an elementary *computation step*. After this step, the states of this vertex, of its neighbours and of the corresponding edges may have changed according to some specific *computation rules*.

3.1 Sequential Computation of a Spanning Tree

We consider here the problem of building a spanning tree in a graph using the depth-first search algorithm.

Suppose that all the vertices are initially in some neutral state (with label N) except exactly one vertex which is in an active state (with label A) and that all edges have label 0. The algorithm will run in such a way that at every time exactly one vertex will be A-labelled.

At each step of the computation, the A-labelled vertex, say u, will act as follows:

1. If u has a N-labelled neighbour v, then u will activate this neighbour: u becomes marked (with label M), v becomes active (with label A) and the edge $\{u, v\}$ becomes 1-labelled.

2. If u has no N-labelled neighbour and has a (unique) M-labelled neighbour w then u will reactivate this neighbour: u enters a final state (with label F) and w becomes A-labelled.

The computation stops as soon as none of the above computation rules may be applied (in that case, all the neighbours of the A-labelled vertex are F-labelled). The spanning tree is then given by the set of all 1-labelled edges.

Figure 1 describes a sample computation using this algorithm (edges with label 1 are drawn as thick edges).

3.2 Distributed Computation of a Spanning Tree Without Local Detection of the Global Termination

In the previous example, the spanning tree is computed in a strictly sequential way since at any time at most one vertex is active (with label A). We will give here a second version of this algorithm that will run in a more distributed way.

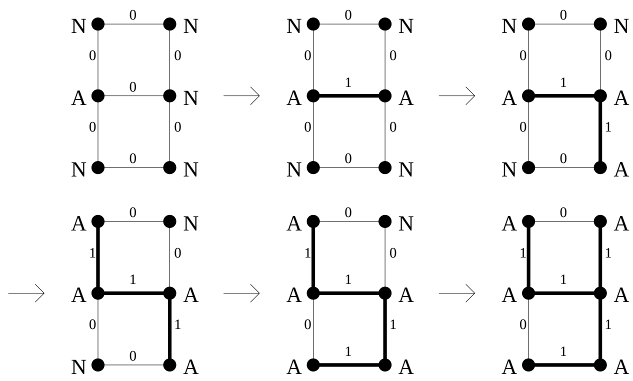


Figure 2: Distributed computation of a spanning tree

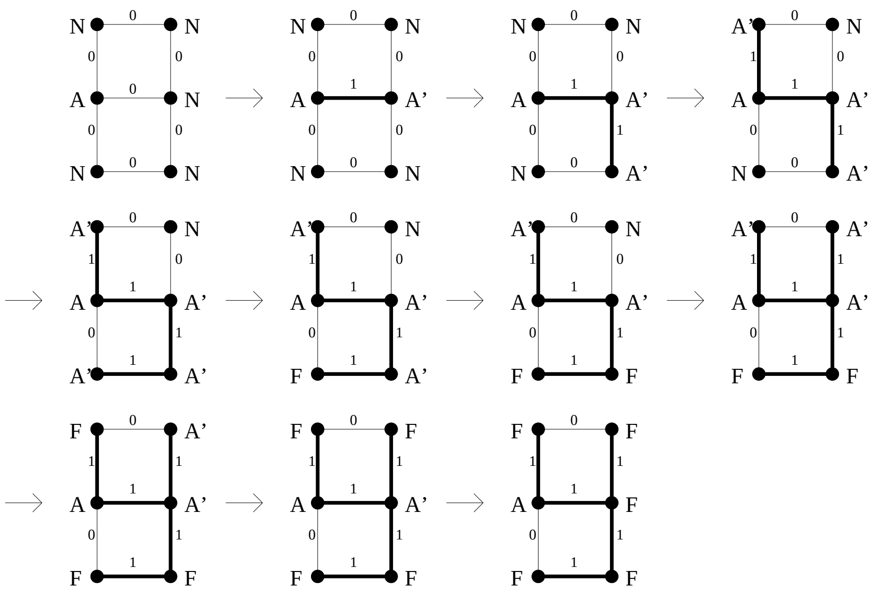


Figure 3: Distributed computation of a spanning tree with local detection of the global termination

As before, we assume that a unique vertex has initially label A, all other vertices having label N and all edges having label 0.

At each step of the computation, an A-labelled vertex u may activate any of its neutral neighbours, say v. In that case, u keeps its label, v becomes A-labelled and the edge $\{u, v\}$ becomes 1-labelled.

Hence, several vertices may be active at the same time. Concurrent steps will be allowed provided that two such steps involve distinct vertices. The computation stops as soon as all the vertices have been activated. As before, the spanning tree is given by the 1-labelled edges.

Figure 2 describes a sample computation using this algorithm. According to the previous discussion, the reader should keep in mind that some of the relabelling steps *may* be applied concurrently.

3.3 Distributed Computation of a Spanning Tree with Local Detection of the Global Termination

The sequential algorithm described before is such that the active vertex is able to *locally* detect the global termination of the algorithm: if all the neighbours of the active vertex are F-labelled, then the computation is terminated. However, it is not difficult to observe that the distributed version of this algorithm does not have this property. We will give here a new distributed version of this algorithm having the property of local detection of the global termination.

As before, we assume that a unique vertex has initially label A, all other vertices having label N and all edges having label 0. The main idea is that the unique initially A-labelled vertex will keep its label until the end of the computation, while other activated vertices will be A'-labelled. As soon as an A'-labelled vertex is no longer "useful" for the computation, it will reach its final state (with label F). More precisely, we will use the following computation rules:

At each step of the computation, an active vertex (with label A or A'), say u, will act as follows:

1. If u has a N-labelled neighbour v, then u will activate this neighbour: u keeps its label, v becomes active (with label A') and the edge $\{u, v\}$ becomes 1-labelled.

2. If u is A'-labelled, has no N-labelled neighbour and is such that all its neighbours to which it is linked by a 1-labelled edge except one of these neighbours are F-labelled, then u becomes F-labelled.

At any time, the subgraph induced by the 1-labelled edges and the A- or A'-labelled vertices is a tree. Intuitively speaking, the second rule means that the vertex u is a leaf in this tree.

Thus, this algorithm runs in two phases (that may overlap): in the first phase, the tree is growing until all vertices are reached; in the second phase, it will decrease (by loosing its leaves) until it is reduced to the initially A-labelled vertex. This vertex is then able to detect that the algorithm has terminated since all its neighbours are F-labelled.

Figure 3 describes a sample computation using this algorithm.

4 Graph Relabelling Systems

4.1 Labelled Graphs

We consider now $L-$*labelled graphs*, that is graphs whose vertices and edges are labelled with labels from a possibly infinite alphabet L. A $L-$labelled graph will be denoted by (G, λ), where G is a graph and $\lambda: V(G) \cup E(G) \to L$ is the *labelling function*. The graph G is called the *underlying graph* of (G, λ), and λ is a *labelling* of G. The class of $L-$labelled graphs will be denoted by $\mathcal{G}_L$, or simply $\mathcal{G}$ if the alphabet L is clear from the context.

Let (G, λ) and (G', λ') be two labelled graphs; (G, λ) is a subgraph of (G', λ'), denoted by $(G, \lambda) \subseteq (G', \lambda')$, if G is a subgraph of G' and λ is the restriction of λ' to $V(G) \cup E(G)$.

A mapping $\varphi: V(G) \cup E(G) \to V(G') \cup E(G')$ is a homomorphism of (G, λ) to (G', λ') if φ is a homomorphism of G to G' which preserves the labelling, that is such that $\lambda'(\varphi(x)) = \lambda(x)$ holds for every $x \in V(G) \cup E(G)$. An *occurrence* of (G, λ) in (G', λ') is an isomorphism φ between (G, λ) and some subgraph (H, η) of (G', λ').

4.2 Graph Relabelling Systems

We introduce in this section the formal notion of graph relabelling systems.

Definition 2 *A* (graph) relabelling rule *is a triple* $R = (G_R, \lambda_R, \lambda'_R)$ *such that* (G_R, λ_R) *and* (G_R, λ'_R) *are two labelled graphs. The labelled graph* (G_R, λ_R) *is the* left-hand side *and the labelled graph* (G_R, λ'_R) *is the* right-hand side *of* R.

Definition 3 *A* graph relabelling system *(GRS for short) is a triple* $\mathcal{R} = (L, I, P)$ *where* L *is a set of labels,* I *a subset of* L *called the set of* initial labels *and* P *a finite set of relabelling rules.*

The intuitive notion of computation step will then correspond to the notion of relabelling step:

Definition 4 *A* $\mathcal{R}-$relabelling step *is a 5-tuple* $(G, \lambda, R, \varphi, \lambda')$ *such that* R *is a relabelling rule in* P *and* φ *is both an occurrence of* (G_R, λ_R) *in* (G, λ) *and an occurrence of* (G_R, λ'_R) *in* (G, λ').

Intuitively speaking, the labelling λ' of G is obtained from λ by modifying all the labels of the elements of $\varphi(G_R, \lambda_R)$ according to the labelling λ'_R. Such a relabelling step will be denoted by $(G, \lambda) \longrightarrow_{R,\varphi} (G, \lambda')$.

The notion of computation then corresponds to the notion of relabelling sequence:

Definition 5 *A* $\mathcal{R}-$relabelling sequence *is a tuple* $(G, \lambda_0, R_0, \varphi_0, \lambda_1, R_1, \varphi_1, \lambda_2, \ldots, \lambda_{n-1}, R_{n-1}, \varphi_{n-1}, \lambda_n)$ *such that for every* i, $0 \le i < n$, $(G, \lambda_i, R_i, \varphi_i, \lambda_{i+1})$ *is a* $\mathcal{R}-$*relabelling step. The existence of such a relabelling sequence will be denoted by* $(G, \lambda_0) \longrightarrow^*_{\mathcal{R}} (G, \lambda_n)$.

The computation stops when the graph is labelled in such a way that no relabelling rule can be applied:

Definition 6 *A labelled graph* (G, λ) *is said to be* $\mathcal{R}-$irreducible *if there exists no occurrence of* (G_R, λ_R) *in* (G, λ) *for every relabelling rule* R *in* P.

For every labelled graph (G, λ) in $\mathcal{G_I}$ we denote by $Irred_{\mathcal{R}}(G, \lambda)$ the set of all $\mathcal{R}-$irreducible labelled graphs (G, λ') such that $(G, \lambda) \longrightarrow^*_{\mathcal{R}} (G, \lambda')$. Intuitively speaking, the set $Irred_{\mathcal{R}}(G, \lambda)$ contains all the final labellings that can be obtained from a $I-$labelled graph (G, λ) by applying relabelling rules in P and may be viewed as the set of all the possible results of the computation encoded by the system $\mathcal{R}$.

Example 7 *The algorithm introduced in Subsection 3.2 may be encoded by the graph relabelling system* $\mathcal{R}_1 = (L_1, I_1, P_1)$ *defined by* $L_1 = \{N, A, 0, 1\}$, $I_1 = \{N, A, 0\}$, *and* $P_1 = \{R\}$ *where* R *is the following relabelling rule:*

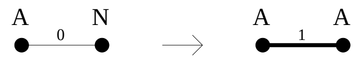
$R:$ 

Figure 1 thus describes a sample $\mathcal{R}_1-$relabelling sequence.

4.3 Local Control Mechanisms

In order to reach a satisfactory expressive power, we introduce some *local control mechanisms*. These mechanisms allow us to restrict in some sense the applicability of relabelling rules.

4.3.1 Graph Relabelling Systems with Priorities

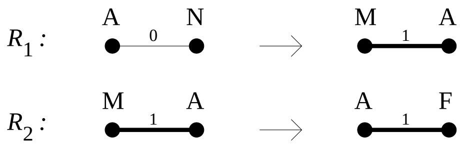
The first mechanism we will consider is obtained by introducing some priority relation on the set of relabelling rules:

Definition 8 *A graph relabelling system* with priorities *(PGRS for short) is a 4-tuple* $\mathcal{R} = (L, I, P, >)$ *such that* (L, I, P) *is a graph relabelling system and* $>$ *is a partial order defined on the set* P, *called the* priority relation.

A $\mathcal{R}-$relabelling step *is then defined as a 5-tuple* $(G, \lambda, R, \varphi, \lambda')$ *such that* R *is a relabelling rule in* P, φ *is both an occurrence of* (G_R, λ_R) *in* (G, λ) *and an occurrence of* (G_R, λ'_R) *in* (G, λ') *and there exists no occurrence* φ' *of a relabelling rule* R' *in* P *with* $R' > R$ *such that* $\varphi(G_R)$ *and* $\varphi(G_{R'})$ *intersect in* G *(that is* $V(\varphi(G_R)) \cap V(\varphi(G_{R'})) = \emptyset$*).*

The notion of relabelling sequence is defined as previously.

Example 9 *The algorithm introduced in Subsection 3.2 may be encoded by the PGRS* $\mathcal{R}_2 = (L_2, I_2, P_2, >_2)$ *defined by* $L_2 = \{N, A, M, F, 0, 1\}$, $I_2 = \{N, A, 0\}$, $P_2 = \{R_1, R_2\}$ *where* R_1 *and* R_2 *are the following relabelling rules:*



with the priority relation: $R_1 >_2 R_2$.

Figure 1 thus describes a sample $\mathcal{R}_2-$relabelling sequence.

4.3.2 Graph Relabelling Systems with Forbidden Contexts

The idea we develop here is to prevent the application of a relabelling rule whenever the corresponding occurrence is "included" in some special configuration, called a context. More formally, we have:

Definition 10 *Let* (G, λ) *be a labelled graph. A* context *of* (G, λ) *is a triple* (H, μ, ψ) *such that* (H, μ) *is a labelled graph and* ψ *an occurrence of* (G, λ) *in* (H, μ).

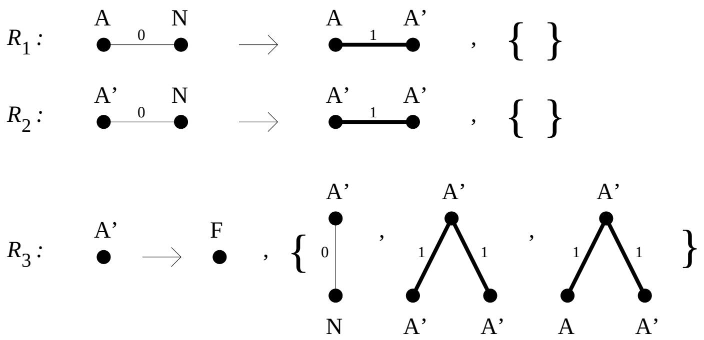
Definition 11 *A* relabelling rule with forbidden contexts *is a 4-tuple* $R = (G_R, \lambda_R, \lambda'_R, F_R)$ *such that* $(G_R, \lambda_R, \lambda'_R)$ *is a relabelling rule and* F_R *is a finite set of contexts of* (G_R, λ_R).

Definition 12 *A* graph relabelling system with forbidden contexts *(FCGRS for short) is a triple* $\mathcal{R} = (L, I, P)$ *defined as a GRS except that the set* P *is a set of relabelling rules with forbidden contexts.*

A relabelling rule with forbidden contexts may be applied on some occurrence if and only if this occurrence is not "included" in an occurrence of some of its forbidden contexts. More formally:

Definition 13 *A* $\mathcal{R}-$relabelling step *is a 5-tuple* $(G, \lambda, R, \varphi, \lambda')$ *such that* R *is a relabelling rule with forbidden contexts in* P, φ *is both an occurrence of* (G_R, λ_R) *in* (G, λ) *and an occurrence of* (G_R, λ'_R) *in* (G, λ'), *and for every context* (H_i, μ_i, ψ_i) *of* (G_R, λ_R), *there is no occurrence* φ_i *of* (H_i, μ_i) *in* (G, λ) *such that* $\varphi_i(\psi_i(G_R, \lambda_R)) = \varphi(G_R, \lambda_R)$.

Example 14 *The algorithm introduced in Subsection 3.3 may be encoded by the FCGRS* $\mathcal{R}_3 = (L_3, I_3, P_3)$ *defined by* $L_3 = \{N, A, M, F, 0, 1\}$, $I_3 = \{N, A, 0\}$, $P_3 = \{R_1, R_2, R_3\}$ *where* R_1, R_2 *and* R_3 *are the following relabelling rules with forbidden contexts:*



The unique vertex of the left-hand side of the rule R_3 *is associated with the top vertex of its forbidden contexts.*

Figure 3 thus describes a sample $\mathcal{R}_3-$relabelling sequence.

4.3.3 PGRS's versus FCGRS's

Due to the control mechanism on the applicability of relabelling rules in PGRS's and FCGRS's, only relabelling steps concerning "far enough" occurrences may be applied concurrently [8]. Roughly speaking, in order to check whether a relabelling rule may be applied on a given occurrence or not it is necessary to consider some "control area" surrounding this occurrence. Two relabelling steps are then "independant" if their corresponding control areas do not intersect. The reader should note here that the diameter of this control area is bounded by some constant only depending on the graph relabelling system.

The comparison between the expressive power of PGRS's and FCGRS's, together with some other types of GRS's (where the occurrences are defined as *induced* subgraphs), has been done in [8]. The main result is the following:

Theorem 15 *PGRS's and FCGRS's are equivalent.*

By "equivalent" we mean that for every PGRS (resp. FCGRS) $\mathcal{R}$ there exists a FCGRS (resp. PGRS) $\mathcal{R}'$ such that for every labelled graph (G, λ), the sets $Irred_{\mathcal{R}}(G, \lambda)$ and $Irred_{\mathcal{R}'}(G, \lambda)$ coincide.

Outline of the proof. For any PGRS $\mathcal{R}$, it is not difficult to construct an equivalent FCGRS $\mathcal{R}'$. The basic idea is to transform every relabelling rule R of $\mathcal{R}$ into a relabelling rule with forbidden contexts R' whose forbidden contexts are all possible configurations corresponding to some "overlapping" of R with some relabelling rule R' whith highest priority.

The difficult part of the proof is to provide for every FCGRS $\mathcal{R}$ an equivalent PGRS $\mathcal{R}'$. The main idea is to construct a PGRS $\mathcal{R}'$ that will "simulate" the behaviour of $\mathcal{R}$. Roughly speaking, this simulation is obtained by the "composition" of several PGRS's corresponding to the following "phases" of the simulation:

Phase 1. *The k-election construction.*

The election problem, discussed in Section 8, consists in distinguishing (electing) in a graph a unique vertex (by means of some special label). This problem is known to be unsolvable in the general case. We define here a "weak" version of this problem, namely the $k-$election problem, which consists in electing several vertices, called *capitals*, in such a way that:

1. Every capital is the root of some tree whose depth is at most d, where d is the maximal diameter of a forbidden context in a rule of $\mathcal{R}$.

2. Every vertex belongs to such a tree.

3. The distance between every two capitals is at least $d + 1$.

This first phase allows us to partition the graph into several *countries* correspnding to the above described trees. Every country has a capital which will organize the simulation.

Phase 2. *Activation of vertices.*

Each capital will generate a token and send it along its own tree (using a depth-first search traversal). A vertex is activated when it owns the token.

Phase 3. *Looking for applicability of relabelling rules.*

When a vertex x is active, it first builds a spanning tree of the ball $B(x, d)$. For every relabelling rule with forbidden contexts R whose left-hand side has p vertices, the vertex x enumerates all possible $p-$tuples in $B(x, d)$, in order to check whether the rule R is applicable or not (in particular, it is necessary to verify that the corresponding occurrence is not included in some of the forbidden contexts). When a rule is found to be applicable then, undeterministically, the rule is applied or not (in order to simulate all possible behaviours of $\mathcal{R}$).

The main difficulty is to correctly "coordonate" the activity of the capitals. In particular, it is necessary to ensure that two "near" capitals are not active at the same time to prevent two vertices to process the third phase on overlapping balls.

□

5 Proof Techniques

A major interest of the graph rewriting formalism for expressing distributed algorithms is that we can use proof techniques issued from classical Rewriting Theory to prove properties of these algorithms. The properties which are of special interest are for instance: correctness, termination, termination detection or time complexity. The aim of this section is to illustrate, on the previously introduced examples, how these techniques can be used to prove such properties.

Roughly speaking, the correspondance between properties of distributed algorithms and graph relabelling systems will be the following. Let $\mathcal{A}$ be a distributed algorithm encoded by some graph relabelling system $\mathcal{R}$ (indifferently, $\mathcal{R}$ may be a GRS, a PGRS or a FCGRS). Then :

- The algorithm $\mathcal{A}$ is "correct" if for every labelled graph (G, λ) every labelled graph in $Irred_{\mathcal{R}}(G, \lambda)$ corresponds to a valid solution [1].
- The algorithm $\mathcal{A}$ always terminates if and only if there exists no infinite $\mathcal{R}-$relabelling chain or, equivalently, if the system $\mathcal{R}$ is noetherian.
- A measure of the time complexity of the algorithm $\mathcal{A}$ can be obtained by considering the maximum length of a $\mathcal{R}-$relabelling chain.
- The algorithm $\mathcal{A}$ has the property of "local detection of the global termination" if there exists a property P and a positive integer k such that every labelled graph containing a vertex v such that the ball $B(v, k)$ satisfies P is $\mathcal{R}-$irreducible (in that case, the vertex v is able to detect the termination of $\mathcal{A}$).

The technique we use to prove the correctness of a given algorithm or, equivalently, of a given relabelling system, is the following: we exhibit a set of *invariant properties*, which are satisfied by every initially labelled graph and that are preserved by every relabelling step. Then, considering an $\mathcal{R}-$irreducible graph (G, λ) (and in particular the fact that no rule can be applied to (G, λ)), this set of invariant properties allows to conclude.

In order to prove that a given relabelling system is noetherian we will use the following useful notion:

Definition 16 *Let $\mathcal{R}$ be a binary relation on a set X, $\mathcal{R}'$ be a binary relation on the set X' and φ be any mapping from X to X'. The relation $\mathcal{R}'$ is* compatible *with the relation $\mathcal{R}$ via φ if:*

$$\forall\ a, b \in X,\ a\ \mathcal{R}\ b \quad \Longrightarrow \quad \varphi(a)\ \mathcal{R}'\ \varphi(b).$$

The interest of this notion is given by the following obvious lemma:

Lemma 17 *Let $\mathcal{R}$ be a binary relation on a set X, $\mathcal{R}'$ be a binary relation on the set X' and φ be any mapping from X to X' such that the relation $\mathcal{R}'$ is compatible with the relation $\mathcal{R}$ via φ. If the relation $\mathcal{R}'$ is noetherian, then so is the relation $\mathcal{R}$.*

The relation $\mathcal{R}'$ will generally be chosen as an acyclic and transitive relation and we will say in that case that $\mathcal{R}'$ is a *compatible order*. The set X' will generally be the set N^p , $p > 0$, of p-tuples of positives integers. In this latter case, we will use the following classical lexicographic order:

Definition 18 *Let p be a non-negative integer. The ordering relation $>_p$ on N^p is defined as follows: $(x_1, x_2, \ldots, x_p)\ >_p\ (y_1, y_2, \ldots, y_p)$ if and only if there exists some i, $1 \leq i \leq p$, such that $x_1 = y_1$, $\ldots$, $x_{i-1} = y_{i-1}$ and $x_i > y_i$.*

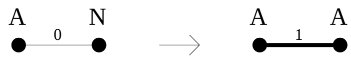
Observe that the ordering relation $>_p$ is noetherian for every p. In the following we will often drop the subscript and $<_p$ will be simply denoted $<$.

We now turn to the study of the previously introduced sample relabelling systems.

[1] We will say that an algorithm $\mathcal{A}$, *undeterministically* solves a given problem if for every labelled graph (G, λ), there exists a labelled graph (G', λ') in $Irred_{\mathcal{R}}(G, \lambda)$ which corresponds to a valid solution.

5.1 The graph relabelling system $\mathcal{R}_1$

Recall first that the GRS $\mathcal{R}_1$ is defined by $\mathcal{R}_1 = (L_1, I_1, P_1)$ with $L_1 = \{N, A, 0, 1\}$, $I_1 = \{N, A, 0\}$, and $P_1 = \{R\}$ where R is the following relabelling rule:

$R :$ 

Then we have:

Theorem 19 *1. The system $\mathcal{R}_1$ is noetherian.*

2. Let (G, λ) be an I_1-labelled graph such that exactly one vertex is $A-$labelled, and (G, λ') be any L_1-labelled graph in $Irred_{\mathcal{R}_1}(G, \lambda)$. Then the set of 1-labelled edges in (G, λ') induces a spanning tree of G. Moreover, the length of a maximal $\mathcal{R}_1-$relabelling chain starting from (G, λ) is at most $|V(G)| - 1$.

Proof. Let $\varphi : \mathcal{G}_{L_1} \to N$ be the mapping which associates with each L_1-labelled graph the number of its $N-$labelled vertices. The usual ordering relation $>$ on N is clearly compatible with $\mathcal{R}_1$ since every application of the rule R strictly decreases the number of $N-$labelled vertices. Thus the system $\mathcal{R}_1$ is noetherian.

To prove the second part of the theorem, we use the following invariant properties:

P_1 Every edge incident with an $N-$labelled vertex is 0-labelled.

P_2 Every $A-$labelled vertex is incident with at least one 1-labelled edge except when there is no 1-labelled edge at all.

P_3 The subgraph induced by the set of 1-labelled edges is a tree.

These three properties are clearly satisfied for the initial graph (G, λ) and preserved by the application of the rule R.

Let now (G, λ') be any labelled graph in $Irred_{\mathcal{R}_1}(G, \lambda)$. Since (G, λ') is irreducible, it contains no $N-$labelled vertex. Thus, thanks to properties P_2 and P_3, the subgraph of (G, λ') induced by the 1-labelled edges is a spanning tree of G.

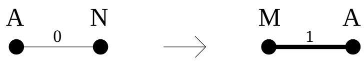
Moreover, the maximal length of an $\mathcal{R}_1-$relabelling chain starting from (G, λ) is equal to the number of $N-$labelled vertices in (G, λ), that is $|V(G)| - 1$, since each application of the rule R replaces an $N-$label by an $A-$label. This concludes the proof. $\square$

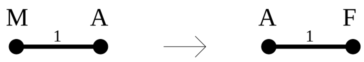
Moreover, the maximal length of an $\mathcal{R}_1-$relabelling chain starting from (G, λ) is equal to the number of $N-$labelled vertices in (G, λ), that is $|V(G)| - 1$, since each application of the rule R replaces an $N-$label by an $A-$label.

The reader should observe that if the initial graph has several $A-$labelled vertices, say p, then the system $\mathcal{R}_1$ constructs a *spanning forest* of G containing exactly p trees.

5.2 The graph relabelling system with priorities $\mathcal{R}_2$

Recall that the PGRS $\mathcal{R}_2$ is defined by $\mathcal{R}_2 = (L_2, I_2, P_2, >_2)$ with $L_2 = \{N, A, M, F, 0, 1\}$, $I_2 = \{N, A, 0\}$, $P_2 = \{R_1, R_2\}$ where R_1 and R_2 are the following relabelling rules:

$R_1 :$ 

$R_2 :$ 

with the priority relation: $R_1 >_2 R_2$.
Then we have:

Theorem 20 *1. The system $\mathcal{R}_2$ is noetherian.*

2. Let (G, λ) be an I_2-labelled graph such that exactly one vertex is $A-$labelled, and (G, λ') be any L_2-labelled graph in $Irred_{\mathcal{R}_2}(G, \lambda)$. Then the set of 1-labelled edges in (G, λ') induces a spanning tree of G. Moreover, the length of a maximal $\mathcal{R}_1-$relabelling chain starting from (G, λ) is at most $2(|V(G)| - 1)$.

3. For every $\mathcal{R}_2-$relabelling chain from (G, λ) to some labelled graph (G, λ''), the graph (G, λ'') is irreducible if and only if it contains an $A-$labelled vertex whose all neighbours are $F-$labelled.

Proof. Let $\varphi : \mathcal{G}_{L_2} \to N^2$ be the mapping which associates with each L_2-labelled graph (H, ν) the couple (ν_N, ν_M), where for every label X, ν_X denotes the cardinal of the set $\nu^{-1}(X)$. The ordering relation $>$ on N^2 is clearly compatible with $\mathcal{R}_2$ since every application of the rules R_1 or R_2 strictly decreases the couple (ν_N, ν_M). Thus the system $\mathcal{R}_2$ is noetherian.

To prove the second part of the theorem, we use the following invariant properties:

P_1 Every edge incident with an $N-$labelled vertex is 0-labelled.

P_2 Every $A-$, $M-$ or $F-$labelled vertex is incident with at least one 1-labelled edge except when there is no 1-labelled edge at all.

P_3 The subgraph induced by the set of 1-labelled edges is a tree.

P_4 There is exactly one $A-$labelled vertex.

P_5 The subgraph induced by the $A-$ and $M-$labelled vertices and the 1-labelled edges is a path such that one of its endpoints is $A-$labelled.

P_6 Every $F-$labelled vertex has no $N-$labelled neighbour.

These properties are clearly satisfied for the initial graph (G, λ). Let us check that all of them and preserved by the application of the rules R_1 and R_2:

P_1 : Only the rule R_1 creates 1-labelled edges and in that case none of its endpoints is $N-$labelled.

P_2 : Every time an $A-$, $M-$ or $F-$labelled vertex appears, thanks to the rule R_1 or R_2, it is clearly incident with a 1-labelled edge.

P_3 : Only the rule R_1 creates a $1-$labelled edge. One of the endpoints of this edge was $N-$labelled and, by P_1, had no other 1-labelled incident edge. Thus this new edge cannot create a cycle. The other endpoint was $A-$labelled and, by P_2, it was already adjacent to at least one 1-labelled edge. Thus the subgraph induced by the set of 1-labelled edges is still connected.

P_4 : Obvious, since no $A-$labelled vertex is created.

P_5 : This is clearly preserved by the rule R_1 (a vertex is added to the path) and by the rule R_2 (a vertex is removed from the path) thanks to the induction hypothesis and property P_3.

P_6 : This follows from the priority mechanism: if an $A-$labelled vertex has an $N-$labelled neighbour then the rule R_2 cannot be applied.

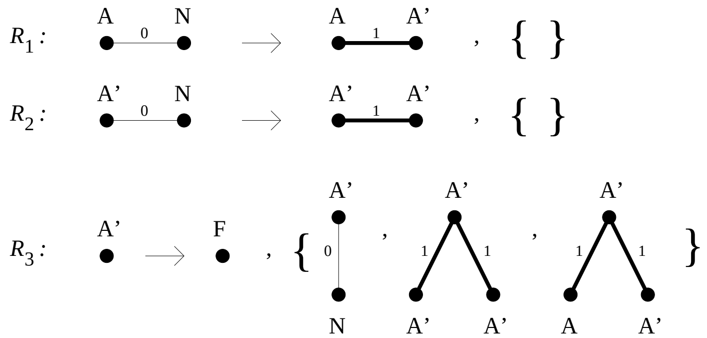
Let now (G, λ') be any labelled graph in $Irred_{\mathcal{R}_2}(G, \lambda)$. Since (G, λ') is irreducible, it contains no $M-$labelled vertex (otherwise, by property P_5, the rule R_2 would be applicable). Thus it contains an $A-$labelled vertex and all other vertices may only be $F-$ or $N-$labelled. By property P_6 and by the irreducibility of (G, λ') other vertices are necessarily $F-$labelled. Finally, thanks to properties P_2 and P_3, the subgraph of (G, λ') induced by the 1-labelled edges is a spanning tree of G.

Moreover, it is easy to observe that every 1-labelled edge in an irreducible graph has been relabelled twice, one time with the rule R_1, one time with the rule R_2. The maximal length of an $\mathcal{R}_2-$relabelling chain is thus twice the number of such edges, that is exactly $2(|V(G) - 1|)$.

We already observed that all the vertices of every irreducible graph are $F-$labelled except one vertex which is $A-$labelled. To complete the proof of the third assertion of the theorem, it suffices to observe that if the $A-$labelled vertex (this vertex is unique by property P_4) has only $F-$labelled neighbours then no rule is applicable. This concludes the proof. □

5.3 The graph relabelling system with forbidden contexts $\mathcal{R}_3$

Recall that the FCGRS $\mathcal{R}_3$ is defined by $\mathcal{R}_3 = (L_3, I_3, P_3)$ with $L_3 = \{N, A, M, F, 0, 1\}$, $I_3 = \{N, A, 0\}$, $P_3 = \{R_1, R_2, R_3\}$ where R_1, R_2 and R_3 are the following relabelling rules with forbidden contexts:



The unique vertex of the left-hand side of the rule R_3 is associated with the top vertex of its forbidden contexts.

Then we have:

Theorem 21 *1. The system $\mathcal{R}_3$ is noetherian.*

2. Let (G, λ) be an I_3-labelled graph such that exactly one vertex is $A-$labelled, and (G, λ') be any L_3-labelled graph in $Irred_{\mathcal{R}_3}(G, \lambda)$. Then the set of 1-labelled edges in (G, λ') induces a spanning tree of G. Moreover, the length of a maximal $\mathcal{R}_3-$relabelling chain starting from (G, λ) is at most $2(|V(G)| - 1)$.

3. For every $\mathcal{R}_3-$relabelling chain from (G, λ) to some labelled graph (G, λ''), the graph (G, λ'') is irreducible if and only if it contains an $A-$labelled vertex whose all neighbours are $F-$labelled.

Proof. Let $\varphi : \mathcal{G}_{L_3} \rightarrow N^2$ be the mapping which associates with each L_3-labelled graph (H, ν) the couple $(\nu_N, \nu_{A'})$. The ordering relation $>$ on N^2 is clearly compatible with $\mathcal{R}_3$ since every application of the rules R_1 or R_2 strictly decreases the couple $(\nu_N, \nu_{A'})$. Thus the system $\mathcal{R}_3$ is noetherian.

To prove the second part of the theorem, we use the following invariant properties:

P_1 Every edge incident with an $N-$labelled vertex is 0-labelled.

P_2 Every $A-$, $A'-$ or $F-$labelled vertex is incident with at least one 1-labelled edge except when there is no 1-labelled edge at all.

P_3 The subgraph induced by the set of 1-labelled edges is a tree.

P_4 There is exactly one $A-$labelled vertex.

P_5 The subgraph induced by the $A-$ and $A'-$labelled vertices and the 1-labelled edges is a tree.

P_6 Every $F-$labelled vertex has no $N-$labelled neighbour.

These properties are clearly satisfied for the initial graph (G, λ). Let us check that all of them and preserved by the application of the rules R_1, R_2 and R_3:

P_1 : Only the rules R_1 and R_2 create 1-labelled edges and in that case none of its endpoints is $N-$labelled.

P_2 : Every time an $A-$ or $A'-$labelled vertex appears, thanks to the rule R_1 or R_2, it is clearly incident with a 1-labelled edge. Every $F-$labelled vertex was previously $A'-$labelled and in that case the result follows from the induction hypothesis.

P_3 : Only the rules R_1 and R_2 create a 1$-$labelled edge. One of the endpoints of this edge was $N-$labelled and, by P_1, had no other 1-labelled incident edge. Thus this new edge cannot create a cycle. The other endpoint was $A-$ or $A'-$labelled and, by P_2, it was already adjacent to at least one 1-labelled edge. Thus the subgraph induced by the set of 1-labelled edges is still connected.

P_4 : Obvious, since no $A-$labelled vertex is created.

P_5 : This is clearly preserved by the rules R_1 and R_2 (a leaf is added to the tree) and by the rule R_3 (a leaf is removed from the tree since, thanks to the forbidden contexts mechanism, the $A'-$labelled vertex has exactly one $A-$ or $A'-$labelled neighbour) thanks to the induction hypothesis and property P_3.

P_6 : This follows from the forbidden contexts mechanism: if an $A'-$labelled vertex has an $N-$labelled neighbour then the rule R_2 cannot be applied.

Let now (G, λ') be any labelled graph in $Irred_{\mathcal{R}_3}(G, \lambda)$. Since (G, λ') is irreducible, it contains no $A'-$labelled vertex (otherwise, by property P_5, the rule R_3 would be applicable on some leaf). Thus it contains an $A-$labelled vertex and all other vertices may only be $F-$ or $N-$labelled. By property P_6 and by the irreducibility of (G, λ') other vertices are necessarily $F-$labelled. Finally, thanks to properties P_2 and P_3, the subgraph of (G, λ') induced by the 1-labelled edges is a spanning tree of G.

Moreover, it is easy to observe that every 1-labelled edge in an irreducible graph has been relabelled once, either with the rule R_1 or with the rule R_2. Then, every $A'-$labelled vertex is relabelled once with the rule R_3. The maximal length of an $\mathcal{R}_2-$relabelling chain is thus the number of 1-labelled edges plus the number of initially $N-$labelled vertices, that is exactly $2(|V(G) - 1|)$.

We already observed that all the vertices of every irreducible graph are $F-$labelled except one vertex which is $A-$labelled. To complete the proof of the third assertion of the theorem, it suffices to observe that if the $A-$labelled vertex (this vertex is unique by property P_4) has only $F-$labelled neighbours then no rule is applicable. This concludes the proof. □

6 Local Computations

6.1 Definitions

Graph relabelling systems, as introduced in the previous section, are in fact an illustration of a more general mechanism called local computations. Local computations as considered here can be described in the following general framework. Recall that $\mathcal{G}_L$ stands for the class of $L-$labelled graphs.

Definition 22 *A* graph rewriting relation *is a binary relation $\mathcal{R} \subseteq \mathcal{G}_L \times \mathcal{G}_L$ closed under isomorphism. The transitive closure of $\mathcal{R}$ is denoted $\mathcal{R}^*$. A $\mathcal{R}-$*rewriting chain *is a sequence $(G_1, \lambda_1), (G_2, \lambda_2), \dots, (G_n, \lambda_n)$ such that for every i, $1 \leq i < n$, $(G_i, \lambda_i)\ \mathcal{R}\ (G_{i+1}, \lambda_{i+1})$.*

By "closed under isomorphism" we mean that if $(G_1, \lambda_1) \simeq (G, \lambda)$ and $(G, \lambda)\ \mathcal{R}\ (G', \lambda')$, then there exists a labelled graph (G'_1, λ'_1) such that $(G_1, \lambda_1)\ \mathcal{R}\ (G'_1, \lambda'_1)$ and $(G'_1, \lambda'_1) \simeq (G', \lambda')$.

Definition 23 *Let $\mathcal{R} \subseteq \mathcal{G}_L \times \mathcal{G}_L$ be a graph rewriting relation and k be a non-negative integer.*

1. *$\mathcal{R}$ is a* relabelling relation *if whenever two labelled graphs are in relation then their underlying graphs are equal (not only isomorphic):*

$$(G, \lambda)\ \mathcal{R}\ (H, \lambda') \quad \Longrightarrow \quad G = H.$$

   *When $\mathcal{R}$ is a relabelling relation we will speak about $\mathcal{R}-$*relabelling chains *instead of $\mathcal{R}-$rewriting chains.*

2. *A relabelling relation $\mathcal{R}$ is $k-$*local *if whenever $(G, \lambda)\ \mathcal{R}\ (G, \lambda')$, the labellings λ and λ' only differ on some ball of radius k:*

$$\exists\ v \in V(G) \text{ such that } \forall\ x \notin V(B_G(v,k)) \cup E(B_G(v,k)),\ \lambda(x) = \lambda'(x).$$

   The relation $\mathcal{R}$ is local *if it is $k-$local for some $k > 0$.*

3. *An $\mathcal{R}-$*normal form *of $(G, \lambda) \in \mathcal{G}_L$ is a graph (G, λ') such that $(G, \lambda)\ \mathcal{R}^*\ (G, \lambda')$, and $(G, \lambda')\ \mathcal{R}\ (G, \lambda'')$ holds for no (G, λ'') in $\mathcal{G}_L$. We say that $\mathcal{R}$ is* noetherian *if for every graph (G, λ) in $\mathcal{G}_L$ there exists no infinite $\mathcal{R}-$relabelling chain starting from (G, λ). Thus, if a relabelling relation $\mathcal{R}$ is noetherian, then every labelled graph has an $\mathcal{R}-$normal form.*

We now define the notion of $k-$locally generated relabelling relation. Roughly speaking, a relabelling relation $\mathcal{R}$ is $k-$locally generated if the knowledge of its restriction on centered balls of radius k suffices to completely determine $\mathcal{R}$. In other words, the relabelling of a ball of radius k does not depend on the rest of the graph:

Definition 24 *Let $\mathcal{R}$ be a relabelling relation and k be a non-negative integer. The relation $\mathcal{R}$ is $k-$*locally generated *if for every labelled graphs (G, λ), (G, λ'), (H, η), (H, η') and every vertices $v \in V(G)$, $w \in V(H)$ such that the balls $B_G(v,k)$ and $B_H(w,k)$ are isomorphic via $\varphi\colon V(B_G(v,k)) \longrightarrow V(B_H(w,k))$ and $\varphi(v) = w$, the following three conditions:*

1. $\forall\ x \in V(B_G(v,k)) \cup E(B_G(v,k)),\ \lambda(x) = \eta(\varphi(x))$ *and* $\lambda'(x) = \eta'(\varphi(x))$,
2. $\forall\ x \notin V(B_G(v,k)) \cup E(B_G(v,k)),\ \lambda(x) = \lambda'(x)$,
3. $\forall\ x \notin V(B_H(w,k)) \cup E(B_H(w,k)),\ \eta(x) = \eta'(x)$,

imply that $(G, \lambda)\ \mathcal{R}\ (G, \lambda')$ if and only if $(H, \eta)\ \mathcal{R}\ (H, \eta')$.

The relation $\mathcal{R}$ is locally generated *if it is $k-$locally generated for some $k > 0$.*

6.2 Distributed Computations of Local Computations

The notion of relabelling sequence defined above obviously corresponds to a notion of *sequential* computation. Let us also note that a $k-$locally generated relabelling relation allows parallel rewritings, since non-overlapping $k-$balls may be relabelled independently. Thus we can define a distributed way of computing by saying that two consecutive relabelling steps concerning non-overlapping $k-$balls may be applied in any order. We say that such relabelling steps *commute* and they may be applied concurrently. More generally, every two relabelling sequences such that the latter one may be obtained from the former one by a succession of such commutations lead to the same resulting labelled graph. Hence, our notion of relabelling sequence may be regarded as a *serialization* [9] of some distributed computation. This model is clearly asynchronous: several relabelling steps *may* be done at the same time but we do not require that all of them have to be performed. In the sequel we will essentially deal with sequential relabelling sequences but the reader should keep in mind that such sequences may be done in a distributed way.

7 Coverings and k−Coverings

The notion of covering is well-known in algebraic topology [10] and has also been studied in Graph Theory [11, 12] where it is in particular related to the notion of uniform emulation [13, 14]. Concerning the theory of distributed computations, coverings of graphs have been used in particular for deriving impossibility results [2, 15].

In the first subsection we introduce this notion of covering and give some basic properties. We then present some standard construction, the Kronecker product, which allows to build coverings of graphs. In order to be used within our framework this notion needs to be particularized to that of *k-coverings*. This will be done in the third subsection and we will finally show how k−coverings are related to local computations.

7.1 Coverings

Definition 25 *A graph G is a* covering *of a graph G' if there exists a surjective homomorphism γ from G onto G' such that for every vertex v of $V(G)$ the restriction of γ to $N_G(v)$ is a bijection onto $N_{G'}(\gamma(v))$. Such a covering is* strict *if G and G' are not isomorphic.*

From this definition, we easily get the following:

Observation 26 *If γ is a surjective homomorphism of G to G' and for every vertex v in $V(G)$ we have $d_G(v) = d_{G'}(\gamma(v))$ then G is a covering of G' via γ.*

Example 27 *Let R_n, $n > 2$, denote the ring on n vertices defined by $V(R_n) = [0, n-1]$ and $E(R_n) = \{\{x, y\} : y = x + 1 \ (mod\ n)\}$. Let now $m \geq n$ and $\gamma_{m,n} : [0, m] \longrightarrow [0, n]$ be the mapping defined by $\gamma_{m,n}(i) = i \ (mod\ n)$, for every $i \in [0, m]$. It is then easy to check that for every $n > 2$, the ring R_{2n} is a covering of the ring R_n via the mapping $\gamma_{2n,n}$.*

The notion of covering is extended in a natural way to labelled graphs. A labelled graph (G, λ) is a covering of (G', λ') via γ if G is a covering of G' via γ and γ preserves the vertex and edge labels.

From the definition, we may observe that if a vertex v in $V(G)$ has two distinct neighbours v_1 and v_2 then these two neighbours must have distinct images in G' by the mapping γ. Thus we have:

Proposition 28 *Let G be a covering of G' via γ and let v_1, v_2 be two distinct vertices of $V(G)$. If $\gamma(v_1) = \gamma(v_2)$ then $N_G(v_1) \cap N_G(v_2) = \emptyset$ and thus $d(v_1, v_2) > 2$.*

Moreover, we have:

Proposition 29 *Let G' be a connected graph and let G be a covering of G' via γ. Then there exists an integer q such that $\forall\ v \in V(G')$, $Card(\gamma^{-1}(v)) = q$.*

Proof. Let $v \in V(G')$, $v' \in N_{G'}(v)$ and $q = card(\gamma^{-1}(v))$. By Proposition 28 we get that the inverse image of $N_{G'}(v)$ is a family of q pairwise disjoint sets in G such that there is a bijection between each of these sets and $N_{G'}(v)$. Thus $q = card(\gamma^{-1}(v)) = card(\gamma^{-1}(v'))$. Now let $w \in V(G')$ be any vertex of G'. Since G' is connected there exists a simple path $v = v_1, ..., v_i = w$ from v to w in G'. From the previous considerations it follows that for every j, $1 \leq j < i$, $card(\gamma^{-1}(v_j)) = card(\gamma^{-1}(v_{j+1}))$, which yields the desired result. □

The integer q is called the number of *sheets* of the covering G. In this case we say that G is a q−*sheeted covering* of G'. The reader should observe that if $q = 1$ then G and G' are isomorphic.

The following proposition states that the structure of trees is preserved by the mapping γ^{-1}:

Proposition 30 *Let G be a q−sheeted covering of G' via γ and T be a subgraph of G'. If T is a tree then $\gamma^{-1}(T)$ is a set of q disjoint trees, each being isomorphic to T.*

The proof of this result can be obtained using a simple inductive argument on the size of T. If we consider now a spanning tree T of G' then $\gamma^{-1}(T)$ is a spanning forest of G, whose all connected components are trees isomorphic to T. The inverse image of an edge $\{x', y'\}$ of G' which does not belong to T is a set of distinct edges of the form $\{x_i, y_i\}$ such that $\gamma(x_i) = x$ and $\gamma(y_i) = y$. Hence, all the x_i's (resp. all the y_i's) belong to distinct components of the spanning forest of G. In [16], Reidemeister proved that all the coverings of G' can be obtained in this way:

Theorem 31 (Reidemeister, 1932) *Let G' be a graph and T a spanning tree of G'. A graph G is a covering of G' if and only if there exist a non-negative integer q and a set $\Sigma = \{\sigma_e,\ e \in E(G') \setminus E(T)\}$ of permutations on $[1, q]$ such that G is isomorphic to the graph $G'_{T,\Sigma}$ defined by:*

$$\begin{aligned} V(G'_{T,\Sigma}) &= \{\ (x,i) :\ x \in V(G'),\ i \in [1,q]\ \}, \\ E(G'_{T,\Sigma}) &= \{\ \{(x,i),(y,i)\} :\ \{x,y\} \in E(T),\ i \in [1,q]\ \} \ \cup \\ &\quad\ \{\ \{(x,i),(y,\sigma_{\{x,y\}}(i))\} :\ \{x,y\} \in E(G') \setminus E(T),\ i \in [1,q]\ \}. \end{aligned}$$

Proof. Let $\gamma : V(G'_{T,\Sigma}) \longrightarrow V(G)$ be the mapping defined by $\gamma(x,i) = x$ for every $(x,i) \in V(G'_{T,\Sigma})$. It is not difficult to check that $G'_{T,\Sigma}$ is a covering of G' via the mapping γ.

Conversely, let G be a $q-$sheeted covering of G' via some mapping γ. We will first construct the required set of permutations Σ and then show that there exists an isomorphism φ from G to $G'_{T,\Sigma}$. By Proposition 30 we know that $\gamma^{-1}(T)$ is a disjoint union of q trees, denoted by $T_1, T_2, \ldots, T_q$, which covers the graph G. Let $\{x, y\}$ be any edge of G' not belonging to T. The inverse image of $\{x, y\}$ is a disjoint set of q edges, denoted by $\{x_1, y_1\}, \ldots, \{x_q, y_q\}$. Since we have $\gamma(x_i) = x$ and $\gamma(y_i) = y$ for every $i \in [1, q]$, no two distinct x_i's (resp. y_i's) can belong to the same tree T_j. For every vertex v in $V(G)$, let us denote by $t(v)$ the index such that v belongs to $T_{t(v)}$. The permutation $\sigma_{\{x,y\}}$ is then defined by $\sigma_{\{x,y\}}(t(x_i)) = t(y_i)$, for every $i \in [1, q]$. Consider now the corresponding graph $G'_{T,\Sigma}$ and let $\varphi : V(G) \longrightarrow V(G'_{T,\Sigma})$ be the mapping defined by $\varphi(v) = (\gamma(v), t(v))$ for every $v \in V(G)$. It is then not difficult to check that φ is an isomorphism from G to $G'_{T,\Sigma}$, which concludes the proof. □

Leighton considered in [17] the problem of deciding whether two graphs admit a common covering or not. The *degree partition* of a graph G is the partition of the vertices of G into the minimal number of blocks $B_0, B_1, \cdots, B_{t-1}$ for which there are constants $r_{i,j}$, $0 \le i, j < t$, such that every vertex v in B_i is incident to $r_{i,j}$ edges linking v to vertices in B_j. The *degree refinement* of G is then the $t \times t$ matrix $R = (r_{i,j})$. Two degree refinements R_1 and R_2 are considered to be the same if they have the same size and if there is a permutation matrix P such that $R_1 = P^T R_2 P$. Then we have:

Theorem 32 (Leighton, 1982) *Given any two finite connected graphs G and H, G and H share a common finite covering if and only if they have the same degree refinement.*

This result will be used in subsection 9.2 for deriving some impossibility results.

7.2 The Kronecker Product

Among methods for producing coverings of a given graph, the Kronecker product by K_2 (the complete graph of size 2) is a standard construction. The Kronecker product was firstly defined on matrices. We deal here with its natural extension to graphs as it was considered in [18].

Definition 33 *Let G and H be two connected graphs. The* Kronecker product *of G by H, denoted $G \wedge H$, is the graph defined by:*

$$\begin{aligned} V(G \wedge H) &= V(G) \times V(H) \\ E(G \wedge H) &= \{\ \{(v,w),(v',w')\}\ :\ \{v,v'\} \in E(G),\ \{w,w'\} \in E(H)\ \} \end{aligned}$$

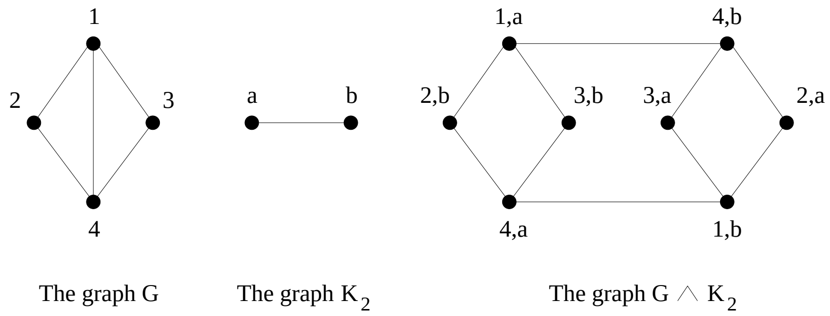


Figure 4: The kronecker product by K_2

The Kronecker product of graphs is a commutative and associative operation through isomorphisms. It has been used in [19] as a simple way of getting coverings of graphs since we have:

Proposition 34 *For every connected graph G, $G \wedge K_2$ is a covering of G.*

Proof. Let γ be the mapping from $V(G \wedge K_2)$ to $V(G)$ defined by $\gamma(v,i) = v$ for every $(v,i) \in V(G \wedge K_2)$. From the definition of the Kronecker product, γ is a surjective homomorphism of $G \wedge K_2$ onto G. Furthermore, the degree of every vertex (v,i) in $G \wedge K_2$ is equal to the degree of $\gamma(v,i)$ in G. Therefore, by Observation 26, $G \wedge K_2$ is a covering of G via γ. □

Example 35 *Figure 4 shows a sample graph G and the corresponding graph $G \wedge K_2$.*

7.3 $k-$Coverings

In this section, we particularize the notion of coverings to that of $k-$coverings by requiring isomorphisms between balls of radius k:

Definition 36 *Let G and G' be two graphs, γ be a surjective homomorphism of G onto G' and k be a non-negative integer. The graph G is a* $k-$covering *of G' via γ if for every vertex $v \in V(G)$, the restriction of γ to $B_G(v,k)$ is an isomorphism between $B_G(v,k)$ and $B_{G'}(\gamma(v),k)$. Such a $k-$covering is* strict *if G and G' are not isomorphic.*

As before, this notion can be extended in a natural way to the case of labelled graphs. The reader should observe that if G is a $k-$covering of G' then G is also a $k'-$covering of G' for every k', $0 < k' < k$.

Example 37 *Let R_n and $\gamma_{m,n}$ be the ring graphs and the mappings defined as in Example 27. It is not difficult to check that for every $k > 0$ and every n, $n > min(3, 2k)$, the graph R_{kn} is a $k-$covering of R_n via the mapping $\gamma_{kn,n}$. However, this is no longer true if $n \leq 2k$ since in that case, for every vertex $v \in V(R_{kn})$, the balls $B_{R_{kn}}(v,k)$ and $B_{R_n}(\gamma_{kn}(v),k)$ are not isomorphic.*

Proposition 28 can naturally be extended to the case of $k-$coverings and we get:

Proposition 38 *Let G be a $k-$covering of G' via γ and let v_1, v_2 be two distinct vertices of $V(G)$. If $\gamma(v_1) = \gamma(v_2)$ then $B_G(v_1,k) \cap B_G(v_2,k) = \emptyset$ and thus $d(v_1,v_2) > 2k$.*

Remark 39 *From the definitions, every $k-$covering of a graph G' is also a covering of G'. However, Example 37 shows for instance that C_6 is a covering of C_3 by γ_6 but not a $1-$covering of C_3.*

The following proposition states which additional requirements a covering much satisfy to be a $k-$covering:

Proposition 40 *Let G and G' be two graphs, γ be a surjective homomorphism of G onto G' and k be a non-negative integer. Then G is a $k-$covering of G' via γ if and only if G is a covering of G' via γ and for every cycle $C = (v_1, e_1, v_2, e_2, \dots, e_i, v_{i+1} = v_1)$ of length $i \leq 2k+1$ in G' the inverse image $\gamma^{-1}(C)$ is a disjoint union of cycles isomorphic to C.*

Proof. We already observed that every $k-$covering is a covering. Moreover, if G is a $k-$covering of G' then every cycle C of length at most $2k+1$ in G' is contained in some centered ball of radius k. The proof of the "if" part then follows from Proposition 38.

Conversely, suppose G is a covering of G' via γ verifying the above property. Let $v' \in V(G')$, $v \in \gamma^{-1}(v')$ and H' be a breadth-first spanning tree of $B_{G'}(v', k)$ rooted at v' (the tree H' has depth at most k). By Proposition 30 the inverse image $\gamma^{-1}(H')$ is a disjoint union of graphs isomorphic to H'. Let $H \simeq H'$ be the connected component of $\gamma^{-1}(H')$ containing v (thus H is a tree rooted at v). Every edge $e' = \{x', y'\}$ such that $x', y' \in B_{G'}(v', k)$ and $e' \notin E(H')$ belongs to some cycle C' contained in $B_{G'}(v', k)$ whose all edges except e' belong to the spanning tree H'. Moreover, since $|C'| \leq 2k+1$, $\gamma^{-1}(C')$ is a disjoint union of copies of C'. We thus get that those vertices $x, y \in V(H)$ with $\gamma(x) = x'$ and $\gamma(y) = y'$ are such that the edge $\{x, y\}$ belongs to $E(G)$. Therefore, the subgraph induced by $V(H)$, that is $B_G(v, k)$, is isomorphic to $B_{G'}(v', k)$ and G is a $k-$covering of G' via γ. □

The question whether a graph has a non-trivial finite or infinite connected $k-$covering is undecidable [20]. However, we can answer positively to this question in the following simple case [21]:

Theorem 41 *Let k be a non-negative integer, G' be a graph and e' an edge in $E(G')$ such that the graph $G' - e'$ is still connected and e' does not belong to any cycle of length at most $2k+1$. Then for every $q \geq 1$ there exists a connected $q-$sheeted $k-$covering G_q of G'.*

Proof. Let $e' = \{x, y\}$ and G_e be the graph defined as the disjoint union of q copies of $G' - e'$. Without loss of generality, we may identify $V(G_e)$ with the set $V(G') \times \{1, \dots, q\}$. Let now G_q be the graph defined by $V(G_q) = V(G_e)$ and $E(G_q) = E(G_e) \cup \{\{(x, i), (y, i+1)\}, 0 \leq i < q\}$ (with addition taken modulo q). The graph G_q is clearly connected and, by Proposition 40, is a $q-$sheeted $k-$covering of G' via the mapping γ given by $\gamma(x, i) = x$ for every $(x, i) \in V(G_q)$. □

7.4 Local Computations and $k-$Coverings

The next proposition establishes the connection between $k-$coverings and $k-$locally generated relabelling relations: if (G, λ) is a $k-$covering of (G', λ') then for every $k-$locally relabelling relation $\mathcal{R}$, a $\mathcal{R}-$relabelling chain starting from (G', λ') induces a $\mathcal{R}-$relabelling chain starting from (G, λ) which "preserves" the $k-$covering relation, as indicated in the following diagram:

$$\begin{array}{cccccc} & (G,\lambda) & \underset{R^*}{\longrightarrow} & (G,\mu) & \\ \text{k-cov} & \downarrow & & \downarrow & \text{k-cov} \\ & (G',\lambda') & \underset{R^*}{\longrightarrow} & (G',\mu') & \end{array}$$

Intuitively speaking, it means that every computation on the graph G' can be "duplicated" on G, by applying the same rules on all the inverse images of the corresponding occurrences. More formally, we have:

Proposition 42 *Let $\mathcal{R}$ be a $k-$locally generated relabelling relation and let (G,λ) be a $k-$covering of (G',λ') via some mapping γ. Moreover, let (G',μ') be a labelled graph such that $(G',\lambda')\ \mathcal{R}^*\ (G',\mu')$. Then there exists a labelling μ of G such that $(G,\lambda)\ \mathcal{R}^*\ (G,\mu)$ and (G,μ) is a $k-$covering of (G',μ').*

Proof. It suffices to prove the result for a one-step relabelling chain. Thus suppose that $(G',\lambda')\ \mathcal{R}\ (G',\mu')$ and that the corresponding relabelling step changes labels only in some ball $B_{G'}(v,k)$, for some vertex $v \in V(G')$. We may then apply this relabelling step to each of the (disjoint) labelled balls of $\gamma^{-1}(B_{G'}(v,k))$, since they are all isomorphic to $B_{G'}(v,k)$. We get in this way the required labelling μ. □

8 The Election Problem

The aim of an election in a graph is to choose exactly one vertex among the set of all vertices. This vertex becomes *elected* and is called the *leader* of the graph.

In our framework, this problem can be formalized in the following way. We say that a relabelling relation $\mathcal{R}$ solves the election problem for a class $\mathcal{C}$ of unlabelled graphs if it is noetherian and if the following condition holds : there exist two labels N and T such that for every graph G in $\mathcal{C}$ whose vertices are initially all $N-$labelled (by some labelling function λ), if $(G,\lambda') \in Irred_{\mathcal{R}}(G,\lambda)$ then there is exactly one vertex $v \in V(G)$ such that $\lambda'(v) = T$ (the vertex v is then the elected vertex).

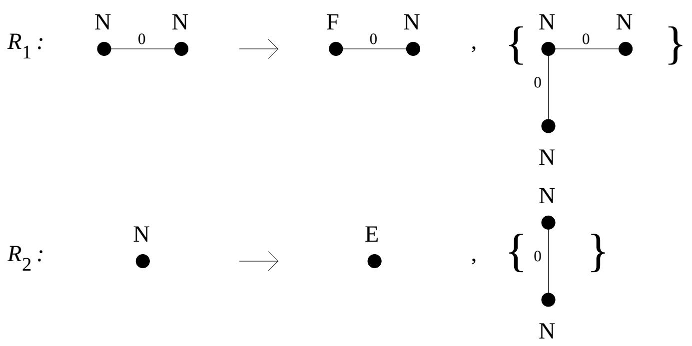
We shall first give some sample election algorithms and then discuss in the following subsections the election problem when vertices of the graph have, or have not, some knowledge on the graph itself, like its size or its topology.

8.1 Examples

We give in this subsection some examples of graph relabelling systems encoding an election algorithm.

We first consider the case of trees. The following FCGRS (Graph Rewriting System with Forbidden Context) allows to elect a vertex in a tree (recall that a relabelling rule with forbidden contexts can be applied to some occurrence if and only if this occurrence is not included in an occurrence of some of its forbidden contexts).

Example 43 *Let $\mathcal{R}_4 = (\mathcal{L}_4, \mathcal{I}_4, P_4)$ be the FCGRS defined by $\mathcal{L}_4 = \{N, F, E,\ 0\}$, $\mathcal{I}_4 = \{N, 0\}$ and $P_4 = \{R_1, R_2\}$ where R_1, R_2 are the following relabelling rules with forbidden contexts:*



Let us call a pendant *vertex any $N-$labelled vertex having exactly one $N-$labelled neighbour. The rule R_1 then consists in "cutting" a pendant vertex in the tree (since the forbidden context ensures that this vertex has no other $N-$labelled neighbour) by giving it a $F-$label. Thus, if (G,λ) is a labelled tree whose all vertices are $N-$labelled and all edges are $0-$labelled then this cutting procedure leads to a unique $N-$labelled vertex which becomes elected thanks to the rule R_2. It is not difficult to observe that every vertex in the tree may be elected by this algorithm.*

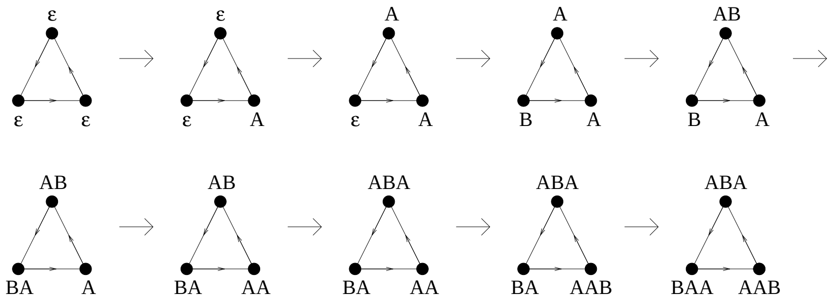
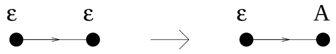


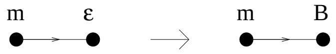
Figure 5: Election on a prime oriented ring

The following algorithm is due to Mazurkiewicz [22] and works on oriented rings having a prime number of vertices. Let n be the number of vertices of the ring; vertex labels are all words over the alphabet $\{A, B\}$ of length at most n. Initially all labels are set to the empty word ε. This algorithm may be encoded by the three rules given below.

Example 44 *The first rule is the following:*

R_1 :


In the next rule we assume that the word m is not the empty word:

R_2 :


For the last rule we assume that $0 < |x| < n$ and that $|x| \leq |m|$. We denote by $m_{|x|}$ the $|x|^{th}$ letter of m.

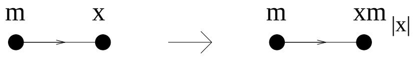
R_3 :


Figure 5 shows a sample computation using these rules (the elected vertex is marked).

Mazurkiewicz has shown that on rings of size n, with n prime, a normal form is always obtained. In this case, vertices are labelled by words of length n which are all different and conjugated (recall that two words f and g are conjugated if $f = uv$ and $g = vu$). The vertex whose label is minimal with respect to the lexicographic ordering is then considered as elected. Of course, a vertex labelled by a word of length n knows whether it is elected or not by determining if its labels is minimal within the set of its conjugates. Hence the elected vertex knows the result. Nevertheless, no vertex can locally detect that the algorithm has terminated. We can note that this rewriting system needs to know the size of the ring. If the size of the ring is not known then there is no election algorithm in the class of prime rings [21].

8.2 Election without any Knowledge

In [2], Angluin proves the following:

Theorem 45 (Angluin, 1980) *There exists no election algorithm for a class of graphs containing both a graph H and a strict covering of H.*

The proof goes as follows. Let G be a strict covering of H via the morphism γ. If a step is performed on an edge e of H then the same step can be performed on each edge of $\gamma^{-1}(e)$. Thus, if H and G have the same uniform initial labelling, every label which appears once in H appears at least twice in G. Hence, no algorithm can elect exactly one vertex both in G and H. This argument can be used in the same way for local computations.

We will prove in the next subsection that the election problem cannot be solved for such a class of graphs by local computations even if the vertices have some knowledge on the size or on the topology of the graph.

8.3 Election Knowing the Size or the Topology

In [22], Mazurkiewicz gave an election algorithm for the class of rings having a prime size; the ring is oriented, anonymous, and its size is known. In every basic computation step, two adjacent vertices may exchange their labels and then compute new ones. To prove that there exists no election algorithm when the size is composite, the author proves that it is always possible to go from some symmetric configuration to another symmetric configuration. From that, he deduces that the algorithm may never terminate. More precisely, let (G, λ) be a labelled ring and let $n = mk$ be the size of G. The vertices of G are denoted by $[0, n-1]$. The labelling λ of G is $m-$symmetric for some $m > 0$ if $\lambda(i) = \lambda(i+m)$ for every $i \in [0, n-1]$ (addition is taken modulo n). In that case, any transformation concerning vertices i and $i+1$ may be applied to vertices $i+jm$ and $i+jm+1$, for every j. Therefore, every $m-$symmetric labelling may be transformed in a new $m-$symmetric labelling. Since the initial labelling is uniform, and thus $m-$symmetric for every m, the algorithm will not terminate if we keep $m-$symmetric configurations.

We will generalize this result to the case of general relabelling rules on graphs such that every vertex has an initial knowledge like the size or the topology. Let G be a graph; a vertex v of G *knows the topology of* G if v has an intial label which encodes a graph G' isomorphic to G but does not allow v to know to which vertex in G' it corresponds. A graph G is *c-minimal* if there is no graph H such that G is a strict covering of H. Let G be a graph which is not c-minimal, H be a graph such that G is a strict covering of H via the morphism γ. A subgraph K of G is *free modulo* γ, or simply $\gamma-$*free*, if $\gamma^{-1}(\gamma(K))$ is a collection of disjoint subgraphs of G, all of them being isomorphic to K. A labelling λ is then $\gamma-$*coherent* if $\gamma(x) = \gamma(y) \Longrightarrow \lambda(x) = \lambda(y)$. By saying that an algorithm *operates on the labels of a subgraph* we mean that the algorithm only needs the knowledge of this subgraph in order to modify its labels and that it modifies nothing else in the graph.

Then we have [23]:

Theorem 46 *Let G be a graph which is not c-minimal. Then there exists no algorithm for the election problem in G verifying the two following conditions:*

1. *each rewriting step operates on labels of a $\gamma-$free subgraph,*
2. *the topology of G is known by each vertex of G.*

This theorem uses the following lemma [23]:

Lemma 47 *Let G and H be two graphs such that G is a strict covering of H via some mapping γ. Let λ be a $\gamma-$coherent labelling of G. If there exists a rewriting step of some algorithm that modifies the labels of some $\gamma-$free subgraph K, then there exists a $\gamma-$coherent labelling λ' of G, $\lambda' \neq \lambda$, that can be obtained after q steps of the algorithm.*

In [24], the class of non-ambiguous graphs is introduced and an election algorithm is given for this class. We will prove that a graph is non-ambiguous if and only if it is c-minimal. From this, several impossibility results can be obtained concerning the election problem for classes of ambiguous graphs (e.g. trees or complete graphs). We also deduce that there exists no election algorithm for an ambiguous graph even knowing its topology.

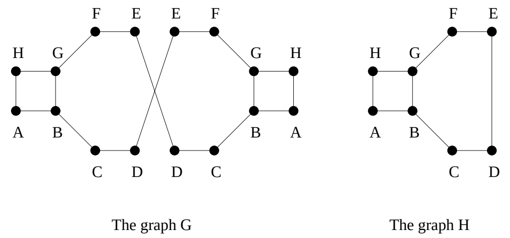


Figure 6: The graph G is ambiguous, it is a 2-covering of the graph H

Let (G, λ) be a labelled graph; the labelling function λ is *bijective* if $\lambda(v) = \lambda(v') \implies v = v'$ for every $v, v' \in V(G)$.

Definition 48 *Let (G, λ) be a labelled graph. The labelling function λ is* locally bijective *if it satisfies the two conditions:*

1. $\forall\ v \in V(G),\ \forall\ v', v'' \in N_G(v),\ \lambda(v') = \lambda(v'') \implies v' = v''$,
2. $\forall\ v', v'' \in V(G),\ \lambda(v') = \lambda(v'') \implies \lambda(N_G(v')) = \lambda(N_G(v''))$.

Definition 49 *A labelling function is* ambiguous *if it is locally bijective and not bijective. A graph G is* ambiguous *if there exists an ambiguous labelling of G.*

In other words, a graph G is ambiguous if there exists a locally bijective labelling λ of G such that $|\lambda(V)| < |V|$.

The next lemma gives a link between ambiguity and coverings [23]:

Lemma 50 *A graph G is ambiguous if and only if it is not c-minimal.*

Proof. Let G be an ambiguous graph and λ be an ambiguous labelling of G. We define the graph H by

$$\begin{aligned} V(H) &= \lambda(V(G)) \\ E(H) &= \{\ \{\lambda(v), \lambda(v')\} :\ \{v, v'\} \in E(G)\ \} \end{aligned}$$

Because of condition 1, there is no self-loop in H. Let γ be the canonical mapping given by $\gamma(v) = \lambda(v)$ and $\gamma(\{v, v'\}) = \{\lambda(v), \lambda(v')\}$. Now, for every $v \in V(G)$, conditions 1 and 2 imply that γ is a bijection from $N_G(v)$ to $N_G(\gamma(v))$. Hence $B_G(v)$ and $B_G(\gamma(v))$ are isomorphic.

Conversely, if G is not c-minimal then there exists some graph H properly covered by G via γ. The labelling λ defined by $\lambda(v) = \gamma(v)$ for every $v \in V(G)$ clearly satisfies conditions 1 and 2 and is not bijective since $|V(H)| < |V(G)|$. Hence the graph G is ambiguous. $\square$

Using this lemma and Theorem 46, we get that there exists no election algorithm for an ambiguous graph even knowing its topology.

Figure 6 shows an ambiguous graph G which is a 2-covering of the graph H.

9 The Recognition Problem

Here we study how local relabelling relations can be used to recognize graph classes. Let L be any fixed set of labels. All the labelled graphs considered in this section are supposed to be $L-$labelled. Recall that the set of all $L-$labelled graphs is denoted by $\mathcal{G}_L$.

Definition 51 *A* graph recognizer *is a pair $(\mathcal{R}, \mathcal{K})$, where $\mathcal{R}$ is a graph relabelling relation and $\mathcal{K}$ is a class of labelled graphs whose elements are called* terminal graphs*. The set of labelled graphs recognized by $(\mathcal{R}, \mathcal{K})$, denoted by $L(\mathcal{R}, \mathcal{K})$ is defined by*

$$L(\mathcal{R}, \mathcal{K}) = \{ \ G \in \mathcal{G}_L : \ Irred_{\mathcal{R}}(G) \cap \mathcal{K} \neq \emptyset \ \}.$$

Definition 52 *A graph recognizer $(\mathcal{R}, \mathcal{K})$ is* deterministic *if $\mathcal{R}$ is noetherian and for every graph G we have either $Irred_{\mathcal{R}}(G) \subseteq \mathcal{K}$ or $Irred_{\mathcal{R}}(G) \cap \mathcal{K} = \emptyset$.*

Note that if $(\mathcal{R}, \mathcal{K})$ is deterministic then $\mathcal{G}_L \setminus L(\mathcal{R}, \mathcal{K}) = L(\mathcal{R}, \mathcal{G}_L \setminus \mathcal{K})$.

These notions apply to unlabelled graphs as follows. A recognizer for unlabelled graphs is defined by a triple $(\mathcal{R}, \mathcal{K}, l_0)$, where $(\mathcal{R}, \mathcal{K})$ is a recognizer for labelled graphs and $l_0 \in L$ is an initial label. An unlabelled graph G is recognized by $(\mathcal{R}, \mathcal{K}, l_0)$, if the labelled graph (G, λ_{l_0}) is recognized by $(\mathcal{R}, \mathcal{K})$ where λ_{l_0} denote the labelling assigning the label l_0 to all vertices and edges. We denote by $L(\mathcal{R}, \mathcal{K}, l_0)$ the class of (unlabelled) graphs recognized in this way.

To be of practical use, the set of terminal graphs must be specified in a "simple" way. In the sequel we assume that $\mathcal{K}$ is specified by some special conditions, called *final conditions*, defined on the labelling functions. These conditions are defined by means of propositional formulas defined inductively in the following way:

1. for every label $l \in L$, l is a formula,

2. if φ and ψ are formulas then $\neg\varphi$, $\varphi \vee \psi$ and $\varphi \wedge \psi$ are formulas.

Now, for $l \in L$, a labelled graph satisfies the formula l if $\lambda^{-1}(l) \neq \emptyset$, and by induction it satisfies $\neg\varphi$ if it does not satisfy φ, it satisfies $\varphi \vee \psi$ if it satisfies φ or ψ, and it satisfies $\varphi \wedge \psi$ if it satisfies φ and ψ.

Thus, such a final condition allows only to verify the presence or the absence of some labels in the labelling of a graph. For instance, the final condition $N \vee \neg Y$ is satisfied by (G, λ) if and only if $\lambda^{-1}(N) \neq \emptyset \vee \lambda^{-1}(Y) = \emptyset$, that is if (G, λ) contains no $Y-$labelled vertex or at least one $N-$labelled vertex. Similarly, the fact that the labelling λ of G contains exactly all the labels of some subset U of L can be specified by the final condition $\bigwedge_{l \in U} l \wedge \bigwedge_{l' \in L \setminus U} \neg l'$.

The reader should observe that such final conditions does not allow to count vertices or edges with given labels, nor to verify their relative positions. For instance, it is not possible to specify that a graph contains exactly one $T-$labelled vertex or two adjacent $T-$labelled vertices.

Definition 53 *Let φ be a final condition; the set of terminal graphs $\mathcal{K}(\varphi)$ is defined by*

$$\mathcal{K}(\varphi) = \{(G, \lambda) : \ (G, \lambda) \ \textit{satisfies} \ \varphi\}.$$

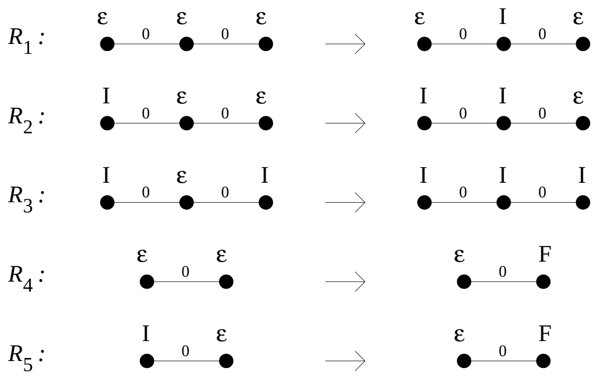
From now on we assume that every set of terminal graphs is of the form $\mathcal{K}(\varphi)$ for some propositional formula φ.

The reader should notice that our notion of recognizability does not coincide with the algebraic notion of recognizability introduced by Mezei and Wright.

9.1 Examples

We give in this subsection examples of graph relabelling systems which allow us to recognize the trees and the complete graphs.

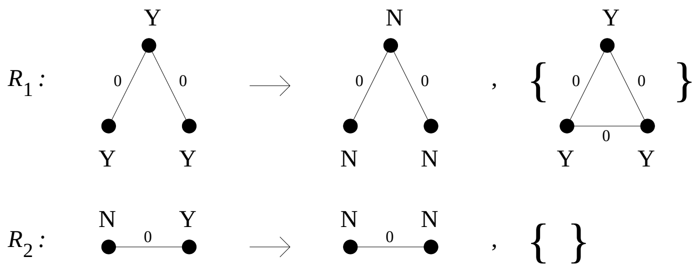
Example 54 *Let $\mathcal{R}_5 = (\mathcal{L}_5, \mathcal{I}_5, P_5, >_5)$ be the PGRS defined by $\mathcal{L}_5 = \{\varepsilon, I, F, 0\}$, $\mathcal{I}_5 = \{\varepsilon, 0\}$ and $P_5 = \{R_1, R_2, R_3, R_4, R_5\}$ with the following rules:*



and the priority relation: $\{R_1, R_2, R_3\} >_5 \{R_4, R_5\}$.

Let now φ *be the final condition* $\varphi = \neg I$. *It can be shown that if* (G, λ) *is a labelled graph whose all vertices are* $\varepsilon-$*labelled and all edges are* $0-$*labelled then every labelled graph* (G, λ') *in* $Irred_{\mathcal{R}_5}(G, \lambda)$ *has no* $I-$*labelled vertex, and thus satisfies* φ, *if and only if* G *has no cycle. Hence, the pair* $(\mathcal{R}_5, \mathcal{K}(\varphi))$ *is a deterministic recognizer for the class of trees.*

Example 55 *Let* $\mathcal{R}_6 = (\mathcal{L}_6, \mathcal{I}_6, P_6)$ *be the FCGRS defined by* $\mathcal{L}_6 = \{N, Y, 0\}$, $\mathcal{I}_6 = \{N, 0\}$ *and* $P_6 = \{R_1, R_2\}$ *with the following relabelling rules with forbidden contexts:*



It can be shown that if (G, λ) *is a labelled graph whose all vertices are* $Y-$*labelled and all edges are* $0-$*labelled then* (G, λ) *is irreducible if and only if* G *is a complete graph. If* G *is not complete then the rule* R_1 *can be applied and, thanks to the rule* R_2, *all the vertices of* G *will become* $N-$*labelled. Thus, if* φ *denotes the final condition* $\varphi = \neg N$, $(\mathcal{R}_6, \mathcal{K}(\varphi)$ *is a deterministic recognizer for the class of complete graphs.*

9.2 Recognition without any Knowledge

We say that a class of graphs is closed under $k-$coverings (resp. connected $k-$coverings), if it contains all $k-$coverings (resp. connected $k-$coverings) of its elements. We first prove that a class of graphs recognized by a $k-$locally generated relabelling relation is closed under $k-$coverings.

Proposition 56 *If a graph* G *is a* $k-$*covering of a graph* H *via some mapping* γ *then every recognizer* $(\mathcal{R}, \mathcal{K})$ *such that* $\mathcal{R}$ *is a* $k-$*locally generated relabelling relation which recognizes* H *recognizes* G *as well. Moreover, if* $(\mathcal{R}, \mathcal{K})$ *is deterministic then the inverse also holds, that is* G *is recognized if and only if* H *is recognized.*

Proof. Let C be a relabelling chain accepting H, that is a relabelling chain leading to some $(H, \mu) \in \mathcal{K}$. Assume that G is a $q-$sheeted covering of H. For each relabelling step of C on the graph H there exists a ball $B_H(v, k)$ such that only labels of $B_H(v, k)$ are modified during this step. This relabelling step may thus be applied to the q disjoint balls of $\gamma^{-1}(B_H(v, k))$. In this way, via γ^{-1}, we construct from C a relabelling chain accepting the graph G. Note that this is due to the fact that a final condition may only check the presence or the absence of labels and that a label appears in H if and only if it appears in G during the simulation of the computation on H.

Suppose now that $(\mathcal{R},\mathcal{K})$ is deterministic. Then $H \notin L(\mathcal{R},\mathcal{K})$ if there exists a relabelling chain rejecting H. As previously, this chain can be simulated on G, giving a relabelling chain rejecting G. But since $(\mathcal{R},\mathcal{K})$ is deterministic, if G is rejected by one relabelling chain then it is rejected by all relabelling chains and thus $G \notin L(\mathcal{R},\mathcal{K})$. □

We then get:

Corollary 57 *Let $\mathcal{R}$ be a $k-$locally generated relabelling relation and $(\mathcal{R},\mathcal{K})$ a deterministic recognizer. If two graphs G and H have a common $k-$covering then $G \in L(\mathcal{R},\mathcal{K})$ if and only if $H \in L(\mathcal{R},\mathcal{K})$.*

In the following, we will use these facts to derive impossibility proofs for the recognition problem by means of locally generated computations. In particular, we shall prove that the class of graphs having exactly one $\ell-$labeled vertex (for some label ℓ), the class of graphs having an odd number of vertices, the class of series-parallel graphs, and some minor-closed classes of graphs cannot be recognized by locally generated relabelling relations.

Our first result shows that counting conditions are not in general $k-$locally verifiable.

Proposition 58 *Let ℓ be a label. There exists no locally generated relabelling relation which recognizes (in a deterministic or non-deterministic way) the class of labelled graphs having exactly one $\ell-$labelled vertex.*

Proof. Let (G,λ) be a labelled ring with exactly one $\ell-$labelled vertex. We know (see Example 37) that G admits a connected strict $k-$covering (G',λ'). By Proposition 29 (G',λ') contains at least two $\ell-$labelled vertices and, by Proposition 56, is recognized (in a deterministic or non-deterministic way) whenever (G,λ) is recognized. □

This result can easily be extended to any class of labelled graphs which contains at least one graph with a cycle and to the case where the condition "having exactly one $\ell-$labelled vertex" is replaced by "with at most i $\ell-$labelled vertices" where i is any non-negative integer.

From Example 37, we also get that a ring having an odd number of vertices always admits a $k-$covering with an even number of vertices; from this fact and Proposition 56 we get:

Proposition 59 *There exists no locally generated relabelling relation which recognizes (in a deterministic or non-deterministic way) the class of graphs having an odd number of vertices.*

This result easily be extended to any class of graphs having an odd number of vertices which contains at least one graph with a cycle.

In [25] it has been proved that class of graphs having an even number of vertices is non-deterministically recognizable by a Graph Relabelling System, thus:

Proposition 60 *For every k, the classes of unlabelled graphs non-deterministically recognized by $k-$locally generated relabelling relations does not form a boolean algebra and does not coincide with the classes of graphs deterministically recognized by $k-$locally generated relabelling relations.*

Consider the series-parallel graph S and the graph C, which is not series-parallel, depicted by Figure 7. These two graphs are not regular but they have the same degree partition, consisting of two blocks $\{3,5\}$ and $\{1,2,4,6\}$. The corresponding degree refinement is $\begin{pmatrix} 0 & 2 \\ 1 & 2 \end{pmatrix}$. Thus, by Theorem 32, C and S share a common finite covering and we get:

Proposition 61 *No locally generated relabelling relation deterministically recognizes series-parallel graphs.*

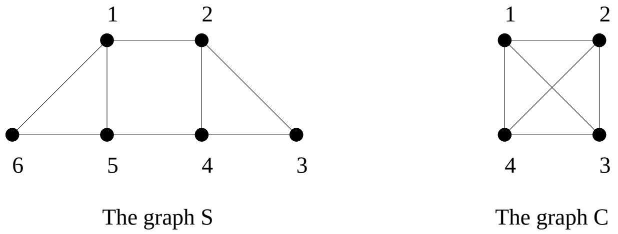


Figure 7: Two graphs having the same degree partition

A graph G is a *minor* of a graph H, denoted by $G \trianglelefteq H$, if G can be obtained from a subgraph of H by a sequence of edge contractions (contracting an edge linking vertices v and v' consists in identifying v and v', deleting the resulting loop, and simplifying, by deleting the multiple edges, the graph thus obtained). A class of graphs is *minor-closed* if it contains all minors of its elements. By a minor-closed class of connected graphs we mean a class of connected graphs containing all connected minors of its elements. Minor-closed classes of connected graphs can be characterized by finite sets of connected forbidden minors, also called obstructions [26, 27]. Then we have [28]:

Theorem 62 *No minor-closed class of connected graphs, which is not the class of all connected graphs and contains at least one graph with at least two cycles, can be recognized by locally generated relabelling relations.*

Before proving this result, we need two technical lemmas.

Lemma 63 *For every planar connected graph G, for every connected graph H having at least two cycles, there exists a connected covering K of H that contains G as minor.*

Proof. A torus is a graph of the form $G(m,n)$ with set of vertices $V(G(m,n)) = [0, m-1] \times [0, n-1]$, and all edges of the form $\{(x_1, y_1), (x_2, y_2)\}$ where either $x_1 = x_2$ and $y_2 = y_1 + 1 \pmod n$ or $y_1 = y_2$ and $x_2 = x_1 + 1 \pmod m$. Since the graph G is planar, there exist m and n such that $G \trianglelefteq G(m,n)$. Since H has at least 2 cycles, there exist two edges $e = \{x, y\}$ and $e' = \{x', y'\}$ in H, with possibly $y = x'$, such that $H \setminus \{e, e'\}$ is connected. Let H' be the graph obtained from H by deleting e and e'. We shall construct a connected graph K such that $G \trianglelefteq K$ and K is a covering of H.

For every $(i,j) \in V(m,n)$ we let $H'(i,j)$ be an isomorphic copy of H' such that $V(H'(i,j)) \cap V(H'(i',j')) = \emptyset$ if $(i,j) \neq (i',j')$. We denote by $w(i,j)$, the vertex in $H'(i,j)$ corresponding to the vertex w of H'. Let now K consists of the union of the graphs $H'(i,j)$ together with the edges $\{x(i,j), y(i,j{+}1 \pmod n)\}$ and $\{z(i,j), u(i{+}1 \pmod m, j)\}$ for every i, j, $0 \le i \le m{-}1$, $0, \le j \le n{-}1$. It is not difficult to observe that, by contracting simultaneously all edges of the subgraphs $H'(i,j)$ of K, we get $G(m,n)$. Hence $G \trianglelefteq G(m,n) \trianglelefteq K$. Moreover, K is a covering of H via the mapping that assigns w to every vertex $w(i,j)$ in $H'(i,j)$. □

Lemma 64 *For every connected graph G, for every connected graph H having at least three cycles, there exists a connected covering K of H that contains G as minor.*

Proof. The proof is an easy extension of that of the previous lemma. Instead of embedding G as a minor in a torus, we embed it as a minor in a large enough "toroidal" parallelepiped $P(\ell, m, n)$ with set of vertices $V(P(\ell, m, n)) = [0, \ell - 1] \times [0, m-1] \times [0, n-1]$ and edges $\{(i,j,k), (i',j',k')\}$ with $i = i'$, $j = j'$, $k' = k + 1 \pmod n$, or $i = i'$, $j' = j + 1 \pmod m$, $k = k'$, or $i' = i + 1 \pmod \ell$, $j = j'$, $k = k'$.

We now select three edges $e = \{x, y\}$, $e' = \{x', y'\}$, $e'' = \{x'', y''\}$ such that the graph H' obtained from H by deleting e, e' and e'' is connected. We construct K as the union of disjoint copies $H'(i,j,k)$ of H' for $(i,j,k) \in V(P(\ell,m,n))$ together with the edges $\{x(i,j,k), y(i,j,k+1 \pmod n)\}$, $\{z(i,j,k), u(i,j+1 \pmod m), k)\}$ and $\{v(i,j,k), w(i+1 \pmod \ell), j, k)\}$.

Similarly, we get $G \trianglelefteq P(l,m,n) \trianglelefteq K$ and K is a covering of H. □

Recall that a $k-$*tree* is a graph that can be obtained by starting with a complete graph on k vertices and repeatedly adding a new vertex linked to k existing vertices which induce a complete graph. We then say that a graph has *tree-width* at most k if and only if it is a partial subgraph of some $k-$tree. We can now prove Theorem 62.

Proof of Theorem 62 Let $\mathcal{C}$ be a minor-closed class of connected graphs that is not the class of all connected graphs and contains at least one graph with at least two cycles. We have two cases to consider:

Case 1. The class $\mathcal{C}$ has bounded tree-width. It follows from Robertson and Seymour [29] that some connected planar graph G is not in $\mathcal{C}$. Let $H \in \mathcal{C}$ with at least two cycles. By Lemma 63, there exists a covering K of H such that $G \trianglelefteq K$. We cannot have $K \in \mathcal{C}$, because we would have $G \in \mathcal{C}$, contradicting the choice of G. Hence $\mathcal{C}$ is not closed under connected coverings.

Case 2. The class $\mathcal{C}$ has unbounded tree-width. Let G be a connected graph not in $\mathcal{C}$. Since every graph with at most two cycles has tree-width at most 2, there is in $\mathcal{C}$ a graph H with at least three cycles. By Lemma 64 we conclude as in Case 1 that $\mathcal{C}$ is not closed under connected coverings. □

In [30], we study some properties of the Kronecker product in relation with graphs minors, planar graphs, graphs with cut-vertices or cut-edges, and graphs having non-trivial automorphism groups. In particular, we prove that for every connected graph G, the graph $G \wedge K_2$ is a bipartite graph with a non-trivial automorphism group. By Proposition 34, the graph $G \wedge K_2$ is a covering of G. Moreover, we give a graph with a cut-vertex (resp. cut-edge) such that its Kronecker product by K_2 is without cut-vertex (resp. cut-edge). We find a nonplanar graph obtained from $K_{3,3}$ which has a planar Kronecker product by K_2. Using previous constructions we get results concerning local computations and recognizability: the classes of graphs having a cut-vertex or a cut-edge are not recognizable, the class of graphs with trivial automorphism group, the class of nonbipartite graphs and the class of nonplanar graphs are not recognizable by locally generated relabelling relations in a deterministic or nondeterministic way.

9.3 Recognition Knowing the Size

The results presented in this subsection have been established in [23]. Let μ be a label. We say that an unlabelled graph G is $\mu-$recognized by some graph recognizer $(\mathcal{R}, \mathcal{K})$ if, starting from the uniformly labelled graph (G, μ) (that is all vertices and edges have the same label μ), $\mathcal{R}$ leads to some final labelling of G belonging to the class $\mathcal{K}$, that is if $\text{Irred}_{\mathcal{R}}(G, \lambda) \cap \mathcal{K} \neq \emptyset$.

The set of labelled graphs *recognized without initial knowledge* (*recognized* for short) by $(\mathcal{R}, \mathcal{K})$ is then defined as the set of unlabelled graphs $\varepsilon-$recognized by $(\mathcal{R}, \mathcal{K})$, where ε is the empty word. We are also interested in locally recognizing graphs which have certain initial knowledge encoded in the label μ. For example, we define an *encoding* of the size of the graph to be a function σ assigning to every graph a label such that for every graph G and H,

$$(\sigma(G) = \sigma(H)) \Leftrightarrow (|V(G)| = |V(H)|)\,.$$

We say that a graph G is recognized *knowing the size* if it is $\sigma(G)-$recognized. We say that a set $\mathcal{G}$ of connected graphs is *recognizable knowing the size* (*s-recognizable* for short) if there exist a relabelling relation $\mathcal{R}$ and a final condition $\mathcal{K}$ such that the set of graphs recognized by $(\mathcal{R}, \mathcal{K})$ knowing the size is exactly the set $\mathcal{G}$.

9.4 Double $k-$Covering Technique

In [31] it was shown that locally recognizable classes of graphs must be closed under $k-$coverings for some k and, as a corollary, that certain classes of graphs were not recognizable without initial knowledge (since they are not closed under $k-$covering). On the other hand, if G is a strict covering of H then the size of G is strictly greater than the size of H. Hence, s-recognizable classes of graphs are not necessarily closed under $k-$coverings. Using these constructions, it is thus not possible to introduce in proofs parameters like the size of the graphs. In this part, we give a very simple relation between graph recognizers and $k-$coverings from which we derive results on graph recognizers for graphs having some initial knowledge. Some sample applications will be given.

We first have:

Lemma 65 *Let $(\mathcal{R}, \mathcal{K})$ be a deterministic graph recognizer, where $\mathcal{R}$ is $k-$locally generated and $\mathcal{K}$ is defined by some final condition. Suppose the graph G is a $k-$covering of a graph K, and let μ be any label. Then, G is $\mu-$recognized by $(\mathcal{R}, \mathcal{K})$ if and only if K is $\mu-$recognized by $(\mathcal{R}, \mathcal{K})$.*

From that we get:

Corollary 66 *Let $(\mathcal{R}, \mathcal{K})$ be a deterministic graph recognizer, where $\mathcal{R}$ is $k-$locally generated and $\mathcal{K}$ is defined by a final condition. Suppose G and H are $k-$coverings of a graph K, and let μ be any label. Then G is $\mu-$recognized by $(\mathcal{R}, \mathcal{K})$ if and only if H is $\mu-$recognized by $(\mathcal{R}, \mathcal{K})$.*

What is interesting in applying the previous very simple result is that μ can encode some common characteristics of G and H such as the number of vertices and/or edges. Thus, if G and H are $k-$coverings of the same graph K and have the same size, then G is recognized knowing the size if and only if H is recognized knowing the size. Note that we cannot extend this fact to K since μ encodes the number of vertices of G and not the number of vertices of K. In the next proposition we apply this technique to some particular classes of graphs:

Proposition 67 *The following graph classes are not deterministically locally s-recognizable: the class of bipartite graphs, the class of graphs having a cut-edge, the class of graphs having a cut-vertex, the class of hamiltonian graphs.*

Remark 68 *Note that using Lemma 30, we get that if G and H are both coverings of some graph K and have the same number of vertices then they have the same number of edges and thus their cycle spaces have the same dimension.*

9.5 Minors

Even knowing the size, minor closed classes of graphs are not recognizable in general. The proof of that is based on some special construction which, starting from two graphs G and H such that G is not a minor of H, allows us to obtain a graph K such that (i) K is a covering of H and (ii) G is a minor a K.

For the $s-$recognizability we cannot use this construction because the sizes of K and H are different. For that, we use another construction which, starting from two graphs G and H such that G is not a minor of H, allows us to obtain a graph K and a graph H' such that (i) G is a minor of K, (ii) K is a $q-$sheeted covering of H, (iii) H' is a $q-$sheeted covering of H, and (v) G is not a minor of H'.

This construction enables us to prove that, even knowing the size, we cannot decide in general by using local computations whether a given graph is minor of a graph or not. In other words, this proves that some minor closed classes of graphs are not s-recognizable.

The main result here is the following:

Theorem 69 *Let $\mathcal{G}$ be a minor-closed class of connected graphs that is closed under homeomorphism, which does not contain all connected graphs and such that there exists a graph with at least three cycles admitting an infinite number of coverings in $\mathcal{G}$. Then $\mathcal{G}$ is not s-recognizable.*

From that we get:

Corollary 70 *The class of connected planar graphs is not recognizable knowing the size.*

9.6 Comparison with Logical Languages

In this section we compare our notion of graph recognition with three main logical languages which allow to express graph properties: First-Order Logic (FOL), Monadic Second-Order Logic (MSOL) and Second-Order Logic (SOL) (see [32] for links between graph properties and graph properties expressible using logical languages).

For short, we will say that "a graph property is locally recognizable" instead of "the class of connected (labelled) graphs verifying a property is recognizable by a locally generated relabelling relation". Similarly, we will speak about a FOL-, MSOL- or SOL-property to say that this property is expressible using the logical language FOL, MSOL or SOL.

In [25], it is proved that the following FOL-properties of a graph G are recognizable by a PGRS: G is simple, G is $k-$regular, G is of degree at most k. On the other hand, we have shown in Proposition 58 that the property of having exactly one given label, which is a FOL-property, is not locally recognizable.

The following properties are not expressible in FOL but are expressible in MSOL: G is $2-$coloriable, G is a tree, G is planar. From the result of [25] we deduce that the first two properties are locally recognizable. On the other hand, we deduce from Theorem 62 that the class of planar graphs is not locally recognizable.

The three following properties are not expressible in MSOL and are expressible in SOL: G has an even number of vertices, G has as many ℓ_1-labelled vertices as ℓ_2-labelled vertices, G has an odd number of vertices. The two first properties are locally recognizable and the last one is not. Therefore, classes of graphs locally recognizable are incomparable with classes of graphs expressible in FOL, MSOL and SOL.

In the case of words and trees, recognizability is equivalent with definability in monadic second order logic. This equivalence does not hold any more for graphs with the notion of recognizability used in our paper. This fact underlines the difference between words and trees on the one hand and general graphs on the other hand. In the case of words and trees, the notion of definability in MSOL and the notion of recognizability by finite automata, which work locally by their very nature, coincide. In the case of graphs, in general, some global computation is necessary. Such global computations on graphs were proposed by Thomas [33], who conjectured that they accept exactly monadic second order definable graphs. In fact, he allows the counting of labels. Thus, each final condition with counting is a boolean combination of atomic conditions of the form $X > k$, where $X \in L$ and k is an fixed integer. This atomic condition is satisfied by (G, λ) if $card(\lambda^{-1}(X)) > k$. The satisfiability relation for boolean combinations of atomic counting conditions is defined inductively in the usual way. The only difference is that, in [33], graphs have a degree uniformely bounded by some constant and the final labelling of a graph is not locally calculated, but is described in terms of tilings of graphs. However, it is not difficult to see that a tiling exists if it can be calculated by a non-deterministic locally generated relabelling relation which rewrites exactly once each vertex. Thus non-deterministic recognition by locally generated relations with counting final conditions captures the Thomas's recognition.

However, we reject counting final conditions since they are not locally verifiable. Roughly speaking, a final condition $\wp$ is $k-$locally verifiable if there exists a $k-$local computation R_f such that (G, λ) satisfies $\wp$ if and only if in the irreducible graph obtained from (G, λ), all vertices are labelled by some label Y. Intuitively, to recognize in a distributed way a graph G by a $k-$locally generated relabelling relation $\mathcal{R}$, we first apply $\mathcal{R}$ to (G, λ_N) (whose all vertices are $N-$labelled) to get some graph (G, λ) irreducible for $\mathcal{R}$. Next, we should be able to verify if (G, λ) satisfies a final condition by means of another $k-$local computation R_f which will notify this fact to all vertices by assigning to them a special label Y if and only if (G, λ) satisfies $\wp$. It is easy to see that final conditions used in Section 9 are $k-$locally verifiable.

10 The Termination Detection Problem

When designing a distributed algorithm, one of the main goals is to ensure that the algorithm has the property of termination detection [6]. Intuitively speaking, this property means that some node in the network is able to detect that the whole algorithm has terminated. This property ensures in particular that it will be possible to execute some algorithm after another algorithm without any "interferences" between them.

In this section, we study this property in the framework of locally generated relabelling relations. Roughly speaking, it means that every vertex v in a graph G, by looking at some ball $B_G(v,k)$, will be able to decide whether the graph is irreducible or not.

In the first subsection we formally introduce this property. We then show how this property is related to the notion of $k-$coverings and, from that, deduce some impossibility results. We then introduce the notion of *quasi $k-$coverings* which allows us to derive more impossibility results. We finally discuss the links between the local detection of the global termination problem, the election problem and the problem of determining the size of a graph.

The results presented in this section have been established in [21].

10.1 The Local Detection of the Global Termination

Let L be any fixed alphabet. We denote by $\mathcal{G} = \mathcal{G}_L$ the set of all $L-$labelled graphs. In this section, we study local computations such that normal forms can be characterized by a set of local configurations:

Definition 71 *Let $\mathcal{R}$ be a $k-$locally generated relabelling relation. Let $\mathcal{I}$ be a subset of $\mathcal{G}$ called the class of* initial graphs *and let T be a subset of connected elements of $\mathcal{G}$. We say that T* characterizes *the normal forms obtained from $\mathcal{I}$ if for every $(G, \lambda) \in \mathcal{I}$ with $(G, \lambda)\ \mathcal{R}^*\ (G, \lambda')$, (G, λ') is a normal form if and only if (G, λ') contains a subgraph isomorphic to some $(K, \mu) \in T$. In that case we also say that (G, λ') is $(K, \mu)-$*characterized.

*Let r be a positive integer. We say that normal forms are $r-$*locally *(or* locally*) characterized if all the elements of T have radius at most r.*

10.2 Applications of $k-$Coverings to Termination Detection

In this subsection we how we can use the notion of $k-$coverings to derive some impossibility results concerning the local detection of the global termination.

Proposition 72 *Let $\mathcal{I} \subseteq \mathcal{G}$ be a class of connected labelled graphs and let $\mathcal{R}$ be a $k-$locally generated relabelling relation. If $\mathcal{I}$ contains two graphs (G, λ) and (G', λ') such that (G, λ) is a strict $k-$covering of (G', λ') via some mapping γ then the normal forms obtained from $\mathcal{I}$ cannot be $r-$locally characterized, for every $r \leq k$.*

Proof. Suppose that $(G', \lambda')\ \mathcal{R}^{n-1}\ (G'_1, \lambda'_1)\ \mathcal{R}\ (G'_2, \lambda'_2)$. By Proposition 42, we can construct a new labelling of G, say (G, λ_1), such that (G, λ_1) is a $k-$covering of (G', λ'_1) (via the mapping γ) and $(G, \lambda)\ \mathcal{R}^*\ (G, \lambda_1)$. Suppose that $(G', \lambda'_1)\ \mathcal{R}\ (G', \lambda'_2)$ holds and concerns some ball $B_{(G', \lambda'_1)}(v, k)$. Then we may apply this relabelling step to exactly one of the connected components of $\gamma^{-1}(B_{(G',\lambda'_1)}(v,k))$ (being isomorphic to $B_{(G',\lambda'_1)}(v,k)$) and obtain the labelled graph (G, λ_2). Now, if the normal forms are $r-$locally characterized for some $r \leq k$, then (G', λ'_2) is $(K, \mu)-$characterized for some $(K, \mu) \in T$. This implies that (G, λ_2) is also $(K, \mu)-$characterized, which contradicts the fact that (G, λ_2) is not a normal form. $\square$

Proposition 72 can easily be generalized in the following way:

Proposition 73 *Let $\mathcal{I} \subseteq \mathcal{G}$ be a class of labelled graphs, and let $\mathcal{R}$ be a $k-$locally generated relabelling relation. Assume that $\mathcal{I}$ contains two labelled graphs (G, λ) and (G', λ') such that (G, λ) is a connected, strict $q-$sheeted $k-$covering of (G', λ') via some mapping γ. Then the normal forms obtained from (G, λ) cannot be $(K, \mu)-$characterized if the labelled graph (K, μ) is such that $\gamma^{-1}(K, \mu)$ is a disjoint union of graphs isomorphic to (K, μ).*

From Theorem 41 and Proposition 72 we get a more general result:

Theorem 74 *Let $\mathcal{I}$ be a class of connected labelled graphs closed under connected $k-$coverings and let $\mathcal{R}$ be a $k-$locally generated relabelling relation. Assume that there exist a graph $(G, \lambda) \in \mathcal{I}$ and an edge $e \in E(G)$ such that $(V(G), E(G) \setminus \{e\})$ is connected, but e belongs to no cycle of length at most $2k + 1$. Then normal forms obtained from $\mathcal{I}$ cannot be $r-$locally characterized, for every $r \leq k$.*

Recall that a graph G is a homeomorphic image of a graph G' if G can be obtained from G' by a sequence of edge subdivisions. Then we have:

Corollary 75 *Let $\mathcal{I}$ be a class of connected labelled graphs closed under connected $k-$coverings and under homeomorphisms, and containing at least one graph with a cycle. Let $\mathcal{R}$ be a $k-$locally generated relabelling relation. Then normal forms obtained from $\mathcal{I}$ cannot be $r-$locally characterized, for every $r \leq k$.*

This result is quite powerful: local computations are very general, they include relabelling with an infinite number of labels and an infinite number of rules, provided that the diameter of the rules is uniformly bounded by some constant. The relabelling relation may be deterministic or not. For Proposition 72 and Theorem 74, T may be infinite provided that the diameter of the graphs is uniformly bounded.

As an illustration, we give some concrete applications [6]. Recall that the majority problem consists in determining, in a graph with $A-$ or $B-$labelled vertices, whether the number of $A-$labelled vertices is greater than the number of $B-$labelled vertices or not. Then we have:

Corollary 76 *There is no local computation system allowing the local detection of termination which solves one of the following problems on uniformly labelled graphs: computing the size of a graph, computing the sum, the product, the minimum or maximum of the vertex labels of a graph, solving the majority problem.*

10.3 Quasi $k-$Coverings and Local Detection of Normal Forms: the Case of $T-$prime Graphs

In this subsection we introduce the notion of quasi $k-$coverings, which allows to extend the results of the previous subsection to certain classes of graphs, as the class of $T-$prime graphs defined as follows. Let G be a connected graph of size n, and let r be an integer dividing n. We say that G is $r-$factorizable if G admits a spanning forest whose all trees have size r. The graph G is said to be $T-$prime if it is not $r-$factorizable for every integer r, $1 < r < n$.

In [34], an election algorithm is given for the class of $T-$prime graphs. The main idea of the algorithm is to construct a partition of the graph into connected subgraphs. Each subgraph is defined by a spanning tree and has a leader (root) with weight equal to the size of the subgraph; all other vertices of the subgraph have weight zero. Initially we consider a partition such that every subgraph consists of a single vertex. We assume that at least one processor starts the computation. Then, there is at least one duel between two adjacent vertices from which we obtain a new partition, with at least one element containing two vertices (in this case we say that the algorithm has started). A leader L with weight w looks for an adjacent subgraph having a leader L' with weight w' such that $w > w'$. In this case, their spanning trees (that is the two corresponding subgraphs) are combined and L remains the leader with the weight $w + w'$, whereas the weight of L' becomes zero. The algorithm terminates when only one tree is left. Clearly, the elected vertex knows that it has been elected if it knows the size of the graph. From results obtained in this subsection we will deduce that this knowledge is necessary.

Definition 77 *Let (G,λ) and (G',λ') be two labelled graphs, $\gamma: V(G) \to V(G')$ be a graph homomorphism and k be a non-negative integer. The graph (G,λ) is a* quasi $k-$covering *of (G',λ') of size s if there exist a finite or infinite $k-$covering (G_0,λ_0) of (G',λ') via some mapping δ, vertices $v_0 \in V(G_0)$, $v \in V(G)$, and an integer $r > 0$ such that :*

1. *$B_{(G,\lambda)}(v,r)$ is isomorphic to $B_{(G_0,\lambda_0)}(v_0,r)$ via φ,*
2. *$Card(V(B_{(G,\lambda)}(v,r))) \geq s$,*
3. *$\gamma = \delta \circ \varphi$ when restricted to $V(B_{(G,\lambda)}(v,r))$.*

The idea behind the notion of quasi $k-$coverings is to enable the simulation of local computations on a given graph in a restricted area of a larger graph, such that the simulation can lead to false conclusions. The restricted area where we can perform the simulation will shrink while the number of simulated steps increases.

Consider a quasi $k-$covering (G,λ) of (G',λ') via some mapping γ. This means that there exist a vertex $z \in V(G)$ and an integer $r > 0$ such that $B_{(G,\lambda)}(z,r)$ is isomorphic to a subgraph of some $k-$covering (G_0,λ_0) of (G',λ'). More precisely, $B_{(G,\lambda)}(z,r)$ is isomorphic via φ to $B_{(G_0,\lambda_0)}(z_0,r)$, where (G_0,λ_0) is some $k-$covering of (G',λ') via some mapping δ. Moreover, $Card(V(B_{(G,\lambda)}(z,r))) \geq s$ and $\gamma = \delta \circ \varphi$ on $V(B_{(G,\lambda)}(z,r))$. Fix now a spanning tree T of G'; then $\delta^{-1}(T) \subseteq V(G_0)$ is a disjoint union of copies of T. Let $\mathcal{J} = \{T_0, T_1, \ldots, T_q\} \subseteq \gamma^{-1}(T) \subseteq V(G)$ be such that for all vertices $u \in V(T_i)$, $0 \leq i \leq q$, the ball $B_{(G,\lambda)}(u,k)$ is included in $B_{(G,\lambda)}(z,r)$. Suppose also, without loss of generality, that $z \in V(T_0)$.

We consider in the following the undirected graph $H = ([0,q],F)$ with $\{i,j\} \in F$ if and only if for some $x \in V(T_i)$, $y \in V(T_j)$ there is an edge $\{x,y\} \in E(G)$. By means of H we obtain a distance d on $\mathcal{J}$ given by $d(T_i,T_j) = d_H(i,j)$. Note that the degree of vertices of H is bounded by $Card(E(G')) - Card(V(G')) + 1$. Hence, for each $d \geq 1$, by choosing s sufficiently large (depending on G', k and d) we obtain $d(T_0,T_i) \geq d$ for some $T_i \in \mathcal{J}$.

Then we have:

Theorem 78 *Let (G,λ), (G',λ'), γ, $\mathcal{J}$ and d be as above, with $d(T_0,T_i) \geq \ell$ for some $T_i \in \mathcal{J}$, $\ell \geq 2k$. Let $\mathcal{R}$ be a $k-$locally generated relabelling relation and suppose $(G',\lambda')\ \mathcal{R}\ (G',\lambda'_1)$. Moreover, assume that for every $T_i \in \mathcal{J}$ with $d(T_0,T_i) \leq \ell$ and for every vertex $u \in V(T_i)$ the labelled balls $B_{(G,\lambda)}(u,k)$ and $B_{(G',\lambda')}(\gamma(u),k)$ are isomorphic via γ.*

Then there exists a labelled graph (G,λ_1) such that $(G,\lambda)\ \mathcal{R}^*\ (G,\lambda_1)$ holds. Moreover, for every $T_i \in \mathcal{J}$ with $d(T_0,T_i) \leq \ell - 2k$ and for every vertex $v \in V(T_i)$ the labelled balls $B_{(G,\lambda_1)}(v,k)$ and $B_{(G',\lambda'_1)}(\gamma(v),k)$ are isomorphic via γ.

Proof. Let $(G',\lambda')\ \mathcal{R}\ (G',\lambda'_1)$ hold via a relabelling step which changes only the relabelling of $B_{(G',\lambda')}(v',k)$. We can simulate this step on each $v \in \gamma^{-1}(v')$ with $v \in V(T_i)$ and $d(T_0,T_i) \leq \ell$. Let (G,λ_1) denote the graph obtained in this way.

Suppose that $w \in V(T_i)$ and $d(T_0,T_i) \leq \ell - 2k$ holds and let $w' = \gamma(w)$. If $B_{(G',\lambda')}(v',k) \cap B_{(G',\lambda')}(w',k) = \emptyset$ then $B_{(G,\lambda)}(v,k) \cap B_{(G,\lambda)}(w,k) = \emptyset$ holds for all $v \in \gamma^{-1}(v') \cap \cup_{T\in\mathcal{J}} V(T)$, and the result follows by induction. Hence assume that $B_{(G',\lambda')}(v',k) \cap B_{(G',\lambda')}(w',k) \neq \emptyset$ and let v be the unique vertex in $\gamma^{-1}(v')$ such that $B_{(G,\lambda)}(v,k) \cap B_{(G,\lambda)}(w,k) \neq \emptyset$. Moreover, let $v \in V(T_j)$ and note that $d(T_0,T_j) \leq d(T_0,T_i) + 2k \leq \ell$. Therefore, the labelled balls $B_{(G,\lambda_1)}(w,k)$ and $B_{(G,\lambda_1)}(w',k)$ are isomorphic. $\square$

Theorem 78 yields a more general result on the impossibility of locally detecting the termination:

Theorem 79 *Let $\mathcal{I}$ be a class of connected labelled graphs and let $\mathcal{R}$ be a $k-$locally generated relabelling relation. Suppose that some $(G', \lambda') \in \mathcal{I}$ has connected quasi $k-$coverings in $\mathcal{I}$ of arbitrary large size. Then normal forms obtained from $\mathcal{I}$ cannot be $r-$locally characterized, for every $r \leq k$.*

Proof. Let $C = ((G', \lambda') = (G', \lambda'_0), (G', \lambda'_1), \ldots, (G', \lambda'_n))$ be a relabelling chain of length n such that (G', λ'_n) is a normal form. Let (G, λ) be a quasi $k-$covering of (G', λ') of size s. For s sufficiently large we have for some $T_i \in \mathcal{J}$ that $d(T_0, T_i) \geq 2k(n+1)$ (recall the definition of $\mathcal{J}$ and d from Theorem 78).

We can apply Theorem 78 with $\ell = 2k(n+1-m)$ for the $m-$th relabelling step of C. We thus obtain a relabelling (G, λ_n) of (G, λ) with $(G, \lambda)\ \mathcal{R}^*\ (G, \lambda_n)$ such that (G, λ_n) is a normal form. However, we have simulated no step of C on vertices belonging to $V(T_i)$ with $d(T_0, T_i) = 2k(n+1)$ (recall that such vertices still have balls of radius k isomorphic to their image by γ). Hence, this contradicts the fact that (G, λ_n) is a normal form. □

Clearly, we cannot use the results of the previous section for the class of $T-$prime graphs, because no connected non-isomorphic $k-$covering of a $T-$prime graph is $T-$prime, but we can apply Theorem 79. For this, suppose that G' is a connected $T-$prime graph containing an edge e' such that $(V(G'), E(G') \setminus \{e'\})$ is still connected, but e' belongs to no cycle of length at most $2k+1$. The construction from Proposition 41 can be easily modified in such a way that we obtain a quasi $k-$covering of size at least $(q-2)Card(V(G'))$ which is also $T-$prime. For this, it suffices to subdivide the edge $\{x_{q-1}, y_0\}$ until the size of the graph obtained is prime (hence, the graph obtained is $T-$prime). Therefore we get:

Corollary 80 *There is no local computation system with local detection of termination where all input graphs are uniformly labelled by the same label, which solves one of the following problems: computing the size of $T-$prime graphs, computing the sum, the product, the minimum or maximum of the vertex labels of a $T-$prime graph, solving the majority problem for the class of $T-$prime graphs.*

Remark 81 *Recall that there exists an election algorithm for $T-$prime graphs which uses the size of the graph as additional knowledge [34]. The natural question which then arises is whether this knowledge is necessary or not. Theorem 79 provides an indirect positive answer to this question. More precisely, suppose that the election problem could be solved with local detection of termination on uniformly labelled $T-$prime graphs (i.e. labelled by a fixed label). Then we could compute after the election the size of the graph, thus contradicting the previous corollary.*

Moreover, note that the construction used in Theorem 79 can be slightly modified in order to obtain the impossibility result for the election problem for $T-$prime graphs directly. For this, it suffices to increase the size of the quasi $k-$covering in such a way that there exists two vertices v_1, v_2 in $B_{(G,\lambda)}(v, r)$ and integers $r_1, r_2 > 0$ with $B_{(G,\lambda)}(v_i, r_i) \subseteq B_{(G,\lambda)}(v, r_0)$ and $Card(V(B_{(G,\lambda)}(v_i, r_i))) \geq s$ for $i = 1, 2$. Moreover, we require that $B_{(G,\lambda)}(v_1, r_1)$ and $B_{(G,\lambda)}(v_2, r_2)$ are disjoint. Clearly, we can apply the simulation on each $B_{(G,\lambda)}(v_i, r_i)$ in parallel, obtaining therefore two elected vertices, a contradiction.

10.4 Comparison with Others Problems

We consider in this last subsection the following three problems: the election problem (ELECT), the problem of locally detecting of termination (LDT) and the problem of computing the size of the graph (SIZE).

We note that ELECT and LDT are equivalent with respect to local computations: if we can solve the election problem for a class of graphs $\mathcal{I}$, then we can also locally detect the termination of a local computation system on $\mathcal{I}$. Conversely, if we have a class of uniformly labelled graphs $\mathcal{I}$ and a local computation system with local termination detection such that every element of $\mathcal{I}$ is reducible, then we can solve ELECT on $\mathcal{I}$.

The first assertion is easily seen by letting the elected vertex compute a spanning tree and check whether a normal form has been reached. For the other direction assume that normal forms obtained from $\mathcal{I}$ with respect to a local computation system of radius k are characterized by a set of labelled graphs T. Moreover, suppose that every graph in $\mathcal{I}$ is reducible and let r be an upper bound for the radius of each element of T. Consider a normal form (G, λ_n) obtained from $(G, \lambda) \in \mathcal{I}$ and two vertices u, v such that both $B_{(G,\lambda_n)}(u, r)$ and $B_{(G,\lambda_n)}(v, r)$ contain a subgraph isomorphic to an element of T. Since T characterizes exactly normal forms the balls $B_{(G,\lambda_n)}(u, r)$ and $B_{(G,\lambda_n)}(v, r)$ contain each a subgraph from T due to the last step of the relabelling chain from (G, λ) to (G, λ_n). Hence, the distance between u and v is at most $4r$. A simple relabelling system with forbidden contexts of radius $2r$ can now be used in order to elect one of the vertices u having the property that $B_{(G,\lambda_n)}(u, r)$ contains a subgraph isomorphic to an element of T (thus labelled by T): every path of length less or equal $2r$ with extremities labelled by T changes the label of one of its extremities into N. A vertex labelled by T with no further neighbours labelled by T at distance less or equal $4r$ becomes elected.

On the other hand, there is also an easy reduction from Size to Elect, since the size of the graph can be computed along a rooted spanning tree. However, Elect is more difficult than Size, a fact which can be seen by considering the class of hypercubes. Clearly, each vertex in a hypercube can compute locally its degree n, thus also the size 2^n of the graph. However, by symmetry arguments it can be easily shown that no local computation system can solve Elect for the class of hypercubes. To see this, assume $\mathcal{R}$ is a relabelling system of radius k and let H_n be the hypercube with 2^n nodes, $n > 2k$. Then we can define a mapping f_{2k} on H_n by letting $f_{2k}(b_1 \dots b_n) = \bar{b}_1 \dots \bar{b}_{2k} b_{2k+1} \dots b_n$ $(b_i \in \{0, 1\})$. Clearly, for each vertex x the balls $B_{H_n}(x, k)$ and $B_{H_n}(f_{2k}(x), k)$ are disjoint and isomorphic. We can simulate each relabelling step of $\mathcal{R}$ on both balls in parallel. This simulation satisfies the condition that the labelled balls of radius k with center y, resp. $f_{2k}(y)$, are isomorphic via f_{2k}. This can be summarized as follows:

Proposition 82 Elect *is equivalent to* Ldt. Size *is reducible to* Elect *but* Elect *is not reducible to* Size.

References

References

[1] J.-R. Fiksel, A. Holliger, and P. Rosensthiel. Intelligent graphs. In R. Read, editor, *Graph theory and computing*, pages 219–265. Academic Press (New York), 1972.

[2] D. Angluin. Local and global properties in networks of processors. In *Proceedings of the 12th Symposium on theory of computing*, pages 82–93, 1980.

[3] T. Kameda and M. Yamashita. Computing on anonymous networks: Part i - characterizing the solvable cases. *IEEE Transactions on parallel and distributed systems*, 7(1):69–89, 1996.

[4] T. Kameda and M. Yamashita. Computing on anonymous networks: Part ii - decision and membership problems. *IEEE Transactions on parallel and distributed systems*, 7(1):90–96, 1996.

[5] N. A. Lynch. *Distributed algorithms.* Morgan Kaufman, 1996.

[6] G. Tel. *Introduction to distributed algorithms.* Cambridge University Press, 1994.

[7] I. Litovsky and Y. Métivier. Computing trees with graph rewriting systems with priorities. *Tree automata and languages*, pages 115–139, 1992.

[8] I. Litovsky, Y. Métivier, and E. Sopena. Different local controls for graph relabelling systems. *Math. Syst. Theory*, 28:41–65, 1995.

[9] A. Mazurkiewicz. Trace theory. In W. Brauer et al., editor, *Petri nets, applications and relationship to other models of concurrency*, volume 255 of *Lecture notes in computer science*, pages 279–324. Spinger-Verlag, 1987.

[10] W. S. Massey. *A basic course in algebraic topology.* Springer-Verlag, 1991. Graduate texts in mathematics.

[11] J. Kratochvíl, A. Proskurowski, and J.-A. Telle. Covering regular graphs. *Journal of Combinatorial Theory, Series B*, 71:1–16, 1997.

[12] N. Norris. Universal covers of graphs: Isomorphism to depth $n-1$ implies isomorphism to all depths. *Discrete Applied Mathematics*, 56:61–74, 1995.

[13] H.-L. Bodlaender and J. van Leeuwen. Simulation of large networks on smaller networks. *Information and Control*, 71:143–180, 1986.

[14] H.-L. Bodlaender. The classification of coverings of processor networks. *J. Parallel Distrib. Comput.*, 6:166–182, 1989.

[15] M. J. Fisher, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. *Distrib. Comput.*, 1:26–29, 1986.

[16] K. Reidemeister. *Einführung in die Kombinatotorishe Topologie.* Vieweg, Brunswick, 1932.

[17] F. T. Leighton. Finite common coverings of graphs. *J. Combin. Theory, Ser. B*, 33:231–238, 1982.

[18] P.M. Weichsel. The kronecker product of graphs. *Proc. Amer. Math. Soc.*, 8:47–52, 1962.

[19] D. Angluin and A. Gardiner. Finite common coverings of pairs of regular graphs. *J. Combin. Theory, Ser. B*, 30:183–187, 1981.

[20] F. Demichelis and W. Zielonka. Decidability questions for graph k-coverings. *Technical Report, LaBRI*, 1996.

[21] Y. Métivier, A. Muscholl, and P.-A. Wacrenier. About the local detection of termination of local computations in graphs. In *International Colloquium on structural information and communication complexity*, pages 188–200, 1997.

[22] A. Mazurkiewicz. Solvability of the asynchronous ranking problem. *Inf. Processing Letters*, 28:221–224, 1988.

[23] E. Godard, Y. Métivier, and A. Muscholl. The power of local computations in graphs with initial knowledge. submitted.

[24] A. Mazurkiewicz. Distributed enumeration. *Inf. Processing Letters*, 61:233–239, 1997.

[25] I. Litovsky and Y. Métivier. Computing with graph rewriting systems with priorities. *Theoret. Comput. Sci.*, 115:191–224, 1993.

[26] M. R. Fellows. The Robertson-Seymour theorems : a survey of applications. *Contemp. Math.*, 89:1–18, 1989.

[27] N. Robertson and P. Seymour. Graph minors vii : Disjoint paths on a surface. *J. Combin. Theory, Ser. B*, 45:212–254, 1988.

[28] B. Courcelle and Y. Métivier. Coverings and minors : application to local computations in graphs. *Europ. J. Combinatorics*, 15:127–138, 1994.

[29] N. Robertson and P. Seymour. Graph minors v : excluding a planar graph. *J. Combin. Theory, Ser. B*, 35:92–114, 1986.

[30] A. Bottreau and Y. Métivier. The kronecker product and local computations in graphs. In *Colloquium on trees in algebra and programming*, volume 1059 of *Lecture notes in computer science*, pages 2–16. Spinger-Verlag, 1996.

[31] I. Litovsky, Y. Métivier, and W. Zielonka. On the recognition of families of graphs with local computations. *Information and Computation*, 118(1):110–119, 1995.

[32] B. Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. *Handbook of graph grammars and computing by graph transformation*, 1:313–397, 1997.

[33] W. Thomas. On logics, tilings, and automata. In *Proceedings of 18th ICALP*, volume 510 of *Lecture notes in computer science*, pages 441–454. Spinger-Verlag, 1991.

[34] Y. Métivier, N. Saheb, and P.-A. Wacrenier. A probabilistic analysis of a rendez-vous algorithm with an application to the computation of a spanning tree in the family of anonymous t-prime graphs. Submitted, 1998.