Theory Comput. Systems **37**, 249–293 (2004)
DOI: 10.1007/s00224-003-1062-1

Theory of Computing Systems

Characterizations of Classes of Graphs Recognizable by Local Computations

Emmanuel Godard,[1] Yves Métivier,[2] and Anca Muscholl[3]

[1]LIF, Université de Provence,
39 rue Joliot-Curie, 13453 Marseille cedex 13, France
egodard@cmi.univ-mrs.fr

[2]LaBRI, Université Bordeaux I, ENSEIRB,
351 cours de la Libération, 33405 Talence, France
metivier@labri.fr

[3]LIAFA, Université Paris VII,
2 place Jussieu, case 7014, 75251 Paris cedex 05, France
anca@liafa.jussieu.fr

Abstract. This paper investigates the power of local computations on graphs, by considering a classical problem in distributed algorithms, the recognition problem. Formally, we want to compute some topological information on a network of processes, possibly using additional knowledge about the structure of the underlying graph. We propose the notion of recognition with structural knowledge by means of local computations. Then we characterize the graph classes that are recognizable with or without structural knowledge. The characterizations use graph coverings and a distributed enumeration algorithm proposed by A. Mazurkiewicz. Several applications are presented, in particular concerning minor-closed classes of graphs.

1. Introduction

The Framework. Local computations on graphs which are described by graph rewriting systems [LMS2] are a formal model for distributed computing. Rewriting systems provide a general tool for describing and analysing distributed algorithms and for proving properties, like correctness or termination. In this paper we consider local computations having some additional knowledge of the network.

We consider networks of processors with arbitrary topology. A network is represented as a connected, undirected labelled graph where vertices denote processors and

edges denote direct communication links. Labels are attached to vertices and edges; they represent the current processor knowledge. Examples are node identities, a distinguished vertex, the number of vertices, the graph diameter or its topology. Edge labels can be weights, the encoding of a spanning tree, etc. Labels are modified locally, that is, on the subgraph consisting of a node and its neighbours, according to certain rules depending on the subgraph only (*local computations*). The relabelling is performed until no more transformation is possible, i.e., until a normal form is obtained. Note that this acceptance condition is global, in the sense that it does not ensure that termination can be detected locally, by each node. We discuss this issue below.

We focus on a classical problem for distributed algorithms, the recognition problem. We consider graphs which are uniformly labelled by some additional initial label (which can encode some structural knowledge on the graph) and the presence or the absence of certain final labels determine whether the graph is accepted or not. The recognition problem asks for graph properties like acyclicity, planarity, etc.

Previous Results. Several basic properties, like regularity or acyclicity, can be recognized by local computations without any additional structural knowledge. On the other hand, it is known that we cannot decide by local computations whether a graph is planar, provided the given graph is uniformly labelled without any structural knowledge [LMZ], [CM]. However, the presence of a distinguished vertex allows us to gather information. More precisely, a distinguished vertex allows us to detect a given minor in a graph [B]. In particular, it allows us to determine whether a graph is planar. A natural question is whether some additional structural information encoded in the initial (uniform) labelling of a graph can help for checking properties like, e.g., planarity.

The classical proof techniques showing the non-existence of solutions given by local computations are based on coverings [A], [FLM], which are known from algebraic topology [M1]. Coverings have also been used in network simulation [BL]. A graph G is a covering of another graph H if there is a surjective morphism from G to H which is locally bijective. The general technique for local computations works as follows. If G and H are two graphs and G covers H, then every local computation on H induces a local computation on G. As a consequence, every graph class which is recognizable by local computations (without additional knowledge) must be closed under coverings. Using this fact it has been proved that the class of planar graphs is not recognizable by local computations [LMZ]. More generally, it has been shown that up to a few exceptions, no minor-closed class of graphs (i.e., characterized by a finite set of forbidden minors by the Graph Minor Theorem) is closed under coverings, implying that it is not recognizable by local computations [CM]. In particular, one cannot decide by local computations whether a given graph is included as a minor in an arbitrary graph.

Main Results. The first main result of this paper (Theorem 22) shows that the well-known necessary condition for recognizability without structural knowledge given by Angluin [A] is also a sufficient condition. Thus we obtain that a class of labelled graphs is recognizable if and only if it is a recursive set closed under the covering relation and its inverse.

A graph G is called covering-minimal if every graph covered by G is isomorphic to G. Covering-minimal graphs play an important role for local computations. For instance,

there exists an election algorithm for covering-minimal graphs [M3], assuming that the size of the graph is known. Moreover, it is easy to see that the property of being covering-minimal is not recognizable without some structural knowledge of the graph. Using [M3] we note that this property is recognizable if we have the size of the graph as additional knowledge. Having an odd number of vertices or having exactly one vertex with a certain label are other examples of properties which are not recognizable without structural knowledge, but become recognizable if the size is known. Thus, local computations assuming the knowledge of the size of the graph are strictly more powerful than without any structural knowledge.

For recognizing graph properties assuming some structural knowledge we cannot apply the covering argument directly. In this paper we introduce a new construction. For a given function ι, which encodes the structural knowledge, we define a binary relation between two graphs G and G' by requiring that $\iota(G) = \iota(G')$ and that G and G' both cover some graph H. Let $\sim^\iota$ denote the reflexive and transitive closure of this relation. We show that any recognizable graph class must be closed under this relation, and that the converse also holds. The characterization (Theorem 48) of recognizability with structural knowledge is the following: a class $\mathcal{F}$ is recognizable with some structural knowledge ι if and only if it is recursive and closed under $\sim^\iota$. We give several applications of this theorem. In particular, we obtain directly that recognizable classes are closed under union, intersection, and complement. Another important consequence is that minor-closed classes of graphs are in general not recognizable, even with knowledge of the size (the exceptions are some simple classes, like trees).

The proofs of our results use Mazurkiewicz' distributed enumeration algorithm [M3]. A distributed enumeration algorithm on a graph G is an algorithm that labels the vertices in such a way that the labelling is a bijection from $V(G)$ to $\{1, 2, \ldots, |V(G)|\}$. In [M3] Mazurkiewicz proposes a distributed enumeration algorithm for non-ambiguous graphs (i.e., graphs where any locally bijective labelling is a bijection). In this paper we prove that non-ambiguous graphs are precisely the covering-minimal graphs. We also show that Mazurkiewicz' algorithm allows us to compute, at each vertex of the given graph G, another graph H that is covered by G. We extend this algorithm to labelled graphs, where the labels are taken in a totally ordered set.

The recognition notion that we have defined here is a deterministic one. That is, a graph is recognized if every run is accepting. At the end of the paper we discuss the issue of nondeterminism for recognizability. There, a graph is recognized if at least one run is accepting.

Related Work. Some of the models related to our model have been considered by Rosenstiehl et al. [RFH], Angluin [A], Yamashita and Kameda [YK1]–[YK3], Boldi and Vigna [BV], and Naor and Stockmeyer [NS]. In [RFH] computations are synchronous and vertices represent (identical) deterministic finite automata. The basic computation step is to compute the next state of each processor according to its state and the states of its neighbours. In [A] an asynchronous model is considered. A basic computation step means that two adjacent vertices exchange their labels and then compute new ones. In [YK2] an asynchronous model is studied and a basic computation step means that a processor either changes its state and sends a message, or it receives a message. In [BV] networks are directed graphs with coloured edges. Each processor changes its state

depending on its previous state and on the states of its in-neighbours. The activation of processors may be synchronous or asynchronous. In [NS] the aim is a study of distributed computations that can be done in a network within a time independent of the size of the network.

Further Research and Contents. In this paper we do not consider the issue of recognizing graph properties with termination detection. This means that our computations are performed until a normal form is reached, but it is not required that in the normal form each vertex is labelled by a terminal state (implicit termination). With implicit termination, when a normal form is reached some vertices may not be aware of this fact. Termination is explicit if in a normal form every vertex can detect the termination (for a formal definition see [T]). Thus the problem of characterizing recognizable graph classes with explicit termination remains open. Some partial results have been obtained recently in [MT], [GM3], [GM2], and [G]. Nevertheless, the negative results obtained in this paper hold in the most general setting and thus remain true, assuming explicit termination.

The paper is organized as follows. Section 2 reviews the basic definitions of labelled graphs, coverings, and ambiguous graphs. We prove that non-ambiguous graphs, as introduced by Mazurkiewicz, are exactly the covering-minimal graphs. In Section 3 we recall the definition of local computations as well as the relation between such computations and coverings. Section 4 introduces labelled graph recognizers and Section 5 presents a characterization of recognizable classes of labelled graphs without structural knowledge. We then describe Mazurkiewicz' distributed enumeration algorithm and give a complete correctness proof in order to make the paper self-contained. Section 6 gives a necessary condition of recognizability for classes of labelled graphs with structural knowledge. In Section 7 we illustrate this condition by showing that in general minor-closed classes of graphs are not recognizable, even knowing the size of graphs. In Section 8 we show that the condition of Section 6 is sufficient. Section 9 investigates particular cases of structural knowledge. Section 10 (Conclusion) presents some elements on the nondeterministic case and more details on related works.

This paper is an improved version of the extended abstracts [GMM] (necessary condition and applications: Sections 6 and 7) and [GM1] (sufficient condition and applications: Sections 8 and 9).

2. Basic Notions and Notations

2.1. *Graphs*

The notations used here are essentially standard [R2]. We only consider finite, undirected, connected graphs without multiple edges and self-loops. If G is a graph, then $V(G)$ denotes the set of vertices and $E(G)$ denotes the set of edges. Two vertices u and v are said to be adjacent if $\{u, v\}$ belongs to $E(G)$. The distance between two vertices u, v is denoted $d(u, v)$. The set of neighbours of v in G, denoted $N_G(v)$, is the set of all vertices of G adjacent to v. For a vertex v, we denote by $B_G(v)$ the ball of radius 1 with centre v, that is, the graph with vertices $N_G(v) \cup \{v\}$ and edges $\{\{u, v\} \in E(G) \mid u \in V(G)\}$.

A homomorphism between G and H is a mapping $\gamma: V(G) \to V(H)$ such that if $\{u, v\}$ is an edge of G, then $\{\gamma(u), \gamma(v)\}$ is an edge of H. Since we deal only with graphs without self-loops, we have $\gamma(u) \neq \gamma(v)$ whenever $\{u, v\}$ is an edge of G. Note also that $\gamma(N_G(u)) \subseteq N_H(\gamma(u))$. For an edge $\{u, v\}$ of G we define $\gamma(\{u, v\}) = \{\gamma(u), \gamma(v)\}$; this extends γ to a mapping $V(G) \cup E(G) \to V(H) \cup E(H)$. We say that γ is an isomorphism if γ is bijective and γ^{-1} is a homomorphism, too. We write $G \simeq G'$ whenever G and G' are isomorphic. A class of graphs is any set of graphs containing all graphs isomorphic to some of its elements. The class of all graphs is denoted by $\mathcal{G}$. Given a graph G, a graph H whose vertices and edges are all in G is a subgraph of G. An occurrence of G in G' is an isomorphism γ between G and a subgraph H of G'.

For any set S, card(S) (or $|S|$) denotes the cardinality of S. For any integer q, we denote by $[1, q]$ the set $\{1, 2, \ldots, q\}$.

2.2. *Labelled Graphs*

Throughout the paper we consider graphs where vertices and edges are labelled with labels from a recursive alphabet L. A graph labelled over L is 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 and the mapping λ is a labelling of G. For a labelled graph (G, λ), $lab((G, \lambda))$ is the set of labels that occur in (G, λ), i.e.,

$$lab((G, \lambda)) = \{\lambda(x) \mid x \in V(G) \cup E(G)\}.$$

The class of labelled graphs over some fixed alphabet L is denoted by $\mathcal{G}_L$. Note that since L is recursive, $\mathcal{G}_L$ is also recursive.

Let (G, λ) and (G', λ') be two labelled graphs. Then (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 the labelling λ' to $V(G) \cup E(G)$.

A mapping $\gamma: V(G) \to V(G')$ is a homomorphism from (G, λ) to (G', λ') if γ is a graph homomorphism from G to G' which preserves the labelling, i.e., such that $\lambda'(\gamma(x)) = \lambda(x)$ holds for every $x \in V(G) \cup E(G)$.

An *occurrence* of (G, λ) in (G', λ') is an isomorphism γ between (G, λ) and a subgraph (H, η) of (G', λ'). It is denoted by $\gamma: (G, \lambda) \hookrightarrow (G', \lambda')$.

Labelled graphs are designated by bold letters like $\mathbf{G}$, $\mathbf{H}$, ... If $\mathbf{G}$ is a labelled graph, then G denotes the underlying graph.

2.3. *Coverings*

We say that a graph G is a *covering* of a graph H via γ (and we write $G \to H$) if γ is a surjective homomorphism from G onto H such that for every vertex v of $V(G)$ the restriction of γ to $B_G(v)$ is a bijection onto $B_H(\gamma(v))$. The covering is proper if G and H are not isomorphic.

The notion of covering extends to labelled graphs in an obvious way. The labelled graph (H, λ') is *covered* by (G, λ) via γ, if γ is a homomorphism from (G, λ) to (H, λ') whose restriction to $B_G(v)$ is an isomorphism from $(B_G(v), \lambda)$ to $(B_H(\gamma(v)), \lambda')$.

A graph G is called *covering-minimal* if every covering from G to some H is a bijection. Note that a graph covering is exactly a covering in the classical sense of algebraic topology, see [M1]. We have the following basic property of coverings [R1]:

Lemma 1. *Suppose that G is a covering of H via γ. Let T be a subgraph of H. If T is a tree, then $\gamma^{-1}(T)$ is a set of disjoint trees, each isomorphic to T.*

By considering simple paths between any two vertices, the previous lemma implies:

Lemma 2. *For every covering γ from G to H there exists an integer q such that $card(\gamma^{-1}(v)) = q$, for all $v \in V(H)$.*

The integer q in the previous lemma is called the number of *sheets* of the covering. We also refer to γ as a *q-sheeted covering*.

In [R1] it is shown that all coverings of H can be obtained from a given spanning tree of H:

Theorem 3 [R1]. *Let H be a graph and let T be a spanning tree of H. A graph G is a covering of H if and only if there exist a non-negative integer q and a set $\Sigma = \{\sigma_{(x,y)} \mid x, y \in V(H), \{x, y\} \in E(H)\backslash E(T)\}$ of permutations[1] on $[1, q]$ such that G is isomorphic to the graph $H_{T,\Sigma}$ defined by*

$$V(H_{T,\Sigma}) = \{(x, i) \mid x \in V(H) \mid i \in [1, q]\},$$
$$E(H_{T,\Sigma}) = \{\{(x, i), (y, i)\} \mid \{x, y\} \in E(T),\ i \in [1, q]\}$$
$$\cup \{\{(x, i), (y, \sigma_{(x,y)}(i))\} \mid \{x, y\} \in E(H)\backslash E(T),\ i \in [1, q]\}.$$

Lemma 4. *Let G be a covering of H via γ and let $v_1, v_2 \in V(G)$ be such that $v_1 \neq v_2$. If $\gamma(v_1) = \gamma(v_2)$ then $B_G(v_1) \cap B_G(v_2) = \emptyset$.*

Example 5. A simple example of a two-sheeted covering is given in Figure 1. The image of each vertex of G is given by the corresponding Roman letter. Furthermore, we note that the image of each vertex is also given by its position on the H pattern (the spanning tree of H suggested in the figure). All examples of coverings below are implicitly described by this geometric scheme, that is based on Theorem 3.

2.4. *Ambiguous Graphs and Coverings*

In this part we give the definition of ambiguous graphs introduced by Mazurkiewicz in [M3] and we show that the non-ambiguous graphs are precisely the covering-minimal graphs.

A labelling is said to be locally bijective if vertices with the same label are not in the same ball and have isomorphic labelled neighbourhoods. Formally, we have:

Definition 6 [M3]. Let L be a set of labels and let (G, λ) be a labelled graph. The labelling λ is *locally bijective* if it verifies the following two conditions:

(1) For each $v \in V$ and for all $v', v'' \in B_G(v)$ we have $\lambda(v') = \lambda(v'')$ if and only if $v' = v''$.

[1] With the convention that $\sigma_{(x,y)} = \sigma^{-1}_{(y,x)}$.

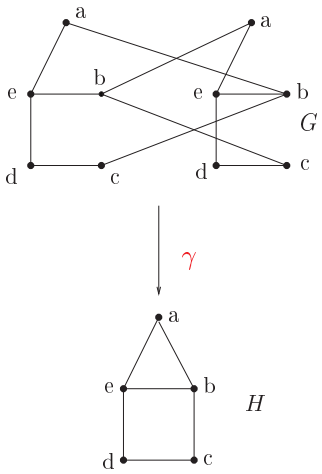


Fig. 1. The morphism γ is a covering from G to H.

(2) For all $v', v'' \in V$ such that $\lambda(v') = \lambda(v'')$, the labelled balls $(B_G(v'), \lambda)$ and $(B_G(v''), \lambda)$ are isomorphic.

A graph G is *ambiguous* if there exists a non-bijective labelling of G which is locally bijective.

The labelling of the graph G in Figure 1 proves that G is ambiguous.

Locally bijective labellings and coverings are closely related through quotient graphs.

Definition 7. Let λ be a labelling of the graph G. We define the *quotient graph* G/λ by letting

- $V(G/\lambda) = \lambda(V(G))$ and
- $E(G/\lambda) = \{\{\alpha, \alpha'\} \mid \exists v, v' \in V(G)$ such that $\{v, v'\} \in E(G), \alpha = \lambda(v), \alpha' = \lambda(v')\}$.

Lemma 8. *Let G be a graph*:

(1) *If λ is a locally bijective labelling of G, then the quotient mapping $G \longrightarrow G/\lambda$ is a covering.*
(2) *Every covering $\gamma\colon G \longrightarrow H$ defines a locally bijective labelling of G.*

Proof. (1) Using condition (1) of Definition 6 we note that G/λ has no self-loop. Moreover, conditions (1) and (2) imply that λ is a bijection from $B_G(v)$ to $B_{G/\lambda}(\lambda(v))$, for each $v \in V(G)$. Hence $B_G(v)$ and $B_{G/\lambda}(\lambda(v))$ are isomorphic.

(2) We consider $V(H)$ as a set of labels and we label a vertex $v \in V(G)$ by $\gamma(v)$. It is straightforward to verify that this labelling is locally bijective. □

Using the previous lemma we obtain:

Corollary 9. *A graph is non-ambiguous if and only if it is covering-minimal.*

3. Local Computations

In this section we recall the definition of local computations and their relation with coverings [LMZ]. They model networks of processors of arbitrary topology. The network is represented as a connected, undirected graph where vertices denote processors and edges denote direct communication links. Labels (or states) are attached to vertices and edges.

Graph relabelling systems and more generally local computations satisfy the following constraints, that arise naturally when describing distributed computations with decentralized control:

(C1) They do not change the underlying graph but only the labelling of its components (edges and/or vertices), the final labelling being the result of the computation.
(C2) They are *local*, that is, each relabelling step changes only a connected subgraph of a fixed size in the underlying graph.
(C3) They are *locally generated*, that is, the applicability of a relabelling rule only depends on the *local context* of the relabelled subgraph.

The relabelling is performed until no more transformation is possible, i.e., until a normal form is obtained.

3.1. *Local Computations*

Local computations as considered here can be described in the following general framework. Let $\mathcal{G}_L$ be the class of L-labelled graphs and let $\mathcal{R} \subseteq \mathcal{G}_L \times \mathcal{G}_L$ be a binary relation on $\mathcal{G}_L$. Then $\mathcal{R}$ is called a *graph rewriting relation*. We assume that $\mathcal{R}$ is closed under isomorphism, i.e., if $\mathbf{G}\mathcal{R}\mathbf{G}'$ and $\mathbf{H} \simeq \mathbf{G}$, then $\mathbf{H}\mathcal{R}\mathbf{H}'$ for some labelled graph $\mathbf{H}' \simeq \mathbf{G}'$. In the remainder of the paper $\mathcal{R}^*$ stands for the reflexive-transitive closure of $\mathcal{R}$. The labelled graph $\mathbf{G}$ is *$\mathcal{R}$-irreducible* (or just irreducible if $\mathcal{R}$ is fixed) if there is no $\mathbf{G}'$ such that $\mathbf{G}\mathcal{R}\mathbf{G}'$. For $\mathbf{G} \in \mathcal{G}_L$, $\mathrm{Irred}_{\mathcal{R}}(\mathbf{G})$ denotes the set of $\mathcal{R}$-irreducible graphs obtained from $\mathbf{G}$ using $\mathcal{R}$, i.e., $\mathrm{Irred}_{\mathcal{R}}(\mathbf{G}) = \{\mathbf{H} \mid \mathbf{G}\,\mathcal{R}^*\,\mathbf{H}$ and $\mathbf{H}$ is $\mathcal{R}$-irreducible$\}$.

Definition 10. Let $\mathcal{R} \subseteq \mathcal{G}_L \times \mathcal{G}_L$ be a graph rewriting relation.

(1) $\mathcal{R}$ is a *relabelling relation* if whenever two labelled graphs are in relation, then the underlying graphs are equal (we say equal, not just isomorphic), i.e.,

$$\mathbf{G}\mathcal{R}\mathbf{H} \qquad \text{implies that} \quad G = H.$$

(2) $\mathcal{R}$ is *local* if it can only modify balls of radius 1, i.e., $(G, \lambda)\mathcal{R}(G, \lambda')$ implies that there exists a vertex $v \in V(G)$ such that

$$\lambda(x) = \lambda'(x) \qquad \text{for every} \quad x \notin V(B_G(v)) \cup E(B_G(v)).$$

The labelled ball $(B_G(v), \lambda)$ is a *support* of the relabelling relation.

The next definition states that a local relabelling relation $\mathcal{R}$ is *locally generated* if the applicability of any relabelling depends only on the balls of radius 1.

Definition 11. Let $\mathcal{R}$ be a relabelling relation. Then $\mathcal{R}$ is *locally generated* if it is local and the following is satisfied: For all labelled graphs (G, λ), (G, λ'), (H, η), (H, η') and all vertices $v \in V(G)$, $w \in V(H)$ such that the balls $B_G(v)$ and $B_H(w)$ are isomorphic via φ: $V(B_G(v)) \longrightarrow V(B_H(w))$ and $\varphi(v) = w$, the following three conditions,

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

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

We only consider recursive relabelling relations such that the set of irreducible graphs is recursive. The purpose of all assumptions about recursiveness done throughout the paper is to have "reasonable" objects with respect to the computational power. By definition, local computations on graphs are computations on graphs corresponding to locally generated relabelling relations.

A sequence $(\mathbf{G}_i)_{0 \leq i \leq n}$ is called an $\mathcal{R}$-*relabelling sequence* (or a *relabelling sequence*, when $\mathcal{R}$ is clear from the context) if $\mathbf{G}_i \mathcal{R} \mathbf{G}_{i+1}$ for every $0 \leq i < n$ (with n being the length of the sequence). A relabelling sequence of length 1 is a *relabelling step*. The relation $\mathcal{R}$ is called *noetherian* on a graph $\mathbf{G}$ if there is no infinite relabelling sequence $\mathbf{G}_0 \mathcal{R} \mathbf{G}_1 \mathcal{R} \cdots$, with $\mathbf{G}_0 = \mathbf{G}$. The relation $\mathcal{R}$ is noetherian on a set of graphs if it is noetherian on each graph of the set. Finally, the relation $\mathcal{R}$ is called noetherian if it is noetherian on each graph.

3.2. *Graph Relabelling Systems*

We now present graph relabelling systems as used for modelling distributed algorithms, by describing the exact form of the relabelling steps. Each step modifies a *star-graph*, i.e., a graph with a distinguished centre vertex connected to all other vertices (and having no other edge besides these edges). As any ball of radius 1 is isomorphic to a star-graph, the support (or precondition) of any relabelling rule is supposed to be a labelled star-graph.

Graph Relabelling Rules. A *graph relabelling rule* is a triple $r = (B_r, \lambda_r, \lambda'_r)$, where B_r is a star-graph and λ_r, λ'_r are two labellings of B_r. We refer to (B_r, λ) as the *precondition* of rule r, whereas (B_r, λ') is referred to as the *relabelling* through r.

Let $r = (B_r, \lambda_r, \lambda'_r)$ be a relabelling rule, let H be an (unlabelled) graph, and let η, η' be two labellings of H. We say that (H, η') is obtained from (H, η) by applying rule r to the occurrence φ of B_r in H (and we write $(H, \eta) \Longrightarrow_{r,\varphi} (H, \eta')$) if the following conditions are satisfied, with v_0 denoting the centre of B_r:

1. φ induces both an isomorphism from (B_r, λ_r) to $B_{(H,\eta)}(\varphi(v_0))$ and from (B_r, λ'_r) to $B_{(H,\eta')}(\varphi(v_0))$.
2. $\eta'(x) = \eta(x)$ for all $x \in (V(H) \backslash V(B_H(\varphi(v_0)))) \cup (E(H) \backslash E(B_H(\varphi(v_0))))$.

In this case we also say that φ is an occurrence of rule r in (H, η) and the image of B_r under φ is called the image of r under φ.

The relabelling relation $\Longrightarrow_r$ induced by rule r is defined by letting $(H, \eta) \Longrightarrow_r (H, \eta')$ if there exists an occurrence φ of r in (H, η) with $(H, \eta) \Longrightarrow_{r,\varphi} (H, \eta')$.

Let $r = (B_r, \lambda_r, \lambda'_r)$ and $s = (B_s, \lambda_s, \lambda'_s)$ be two (not necessarily distinct) relabelling rules and let

$$\varphi_r \colon (B_r, \lambda_r) \hookrightarrow (H, \eta), \qquad \varphi_s \colon (B_s, \lambda_s) \hookrightarrow (H, \eta)$$

be two occurrences of r and s respectively in (H, η). We say that these two occurrences *overlap* if

(i) the images of B_r by φ_r and B_s by φ_s have a common vertex, and
(ii) either $r \neq s$ or ($r = s$ and $\varphi_r \neq \varphi_s$).

Graph Relabelling Systems. A *graph relabelling system* is a recursive set R of graph relabelling rules, such that the set of labelled star-graphs that are preconditions of a rule in R is also recursive.

The relabelling relation $\Longrightarrow_R$ is defined by $(G, \lambda) \Longrightarrow_R (G, \lambda')$ if there is a rule $r \in R$ such that $(G, \lambda) \Longrightarrow_r (G, \lambda')$.

Examples of graph relabelling systems are presented in [LMS2] and [LMZ].

Clearly, graph relabelling systems represent locally generated relabelling relations. Conversely, any locally generated relabelling relation can be represented by a graph relabelling system.

Proposition 12. *Let $\mathcal{R}$ be a relabelling relation. Then $\mathcal{R}$ is both locally generated and a recursive relation such that the set of irreducible graphs is recursive if and only if there exists a graph relabelling system R such that $\mathcal{R}$ equals $\Longrightarrow_R$.*

Proof. Given a locally generated relabelling relation $\mathcal{R}$, we have to find a graph relabelling system R that generates $\mathcal{R}$.

We define

$$R = \{(B, \lambda, \lambda') \mid B \text{ is a star-graph, } (B, \lambda)\ \mathcal{R}\ (B, \lambda')\}.$$

First, R is obviously recursive since $\mathcal{R}$ is. The set of preconditions of R is also recursive, since one can check whether a precondition does not belong to the set of $\mathcal{R}$-irreducible graphs. It is then straightforward to verify that R generates exactly $\mathcal{R}$ from Definition 11. □

In the following we do not discriminate between a locally generated relabelling relation and a graph relabelling system that generates it.

Generic Rules. We explain here the convention under which we describe graph relabelling systems later. If the number of rules is finite, then we describe all rules by their preconditions and relabellings. We also describe a family of rules by a generic rule ("meta-rule"). In this case we consider a generic star-graph of generic centre v_0 and of a generic set of vertices $B(v_0)$. Within these conventions, we refer to a vertex v of the

star-graph by writing $v \in B(v_0)$. If $\lambda(v)$ is the label of v in the precondition, then $\lambda'(v)$ will be its label in the relabelling. In the description we omit labels that are not modified by the rule. This means that if $\lambda(v)$ is a label such that $\lambda'(v)$ is not explicitly described in the rule for a given v, then $\lambda'(v) = \lambda(v)$. In all the examples of graph relabelling systems that we consider in this paper the edge labels are never changed.

With these conventions, the only point we have to care about is to verify that the set of graph relabelling rules and the set of preconditions described by the generic rule are recursive.

Example. Our first example is a $(d+1)$-coloring of regular graphs of degree d. This example allows us to use the above-described conventions.

Example 13. We consider the graph relabelling system COLO$_d$. The value of the label of a vertex v is denoted by $c(v)$. The "colours" used here are integers from $[1, d+1]$, all vertices are initially labelled by 0. The following generic rule means that if v_0 is labelled by 0, then v_0 is relabelled by the smallest value that does not occur as label of one of its neighbours. The edge labels are not used in this example.

COLO$_d$: $(d+1)$**-Coloring**

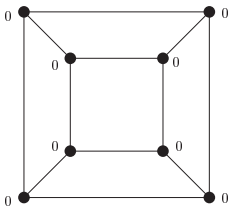
Precondition:

- $c(v_0) = 0$.

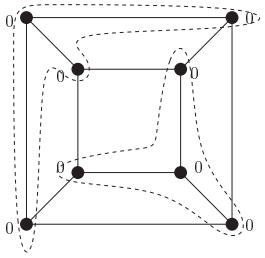
Relabelling:

- $c'(v_0) := \min([1, d+1] \backslash \{c(v) \mid v \in B(v_0), c(v) \neq 0\})$.

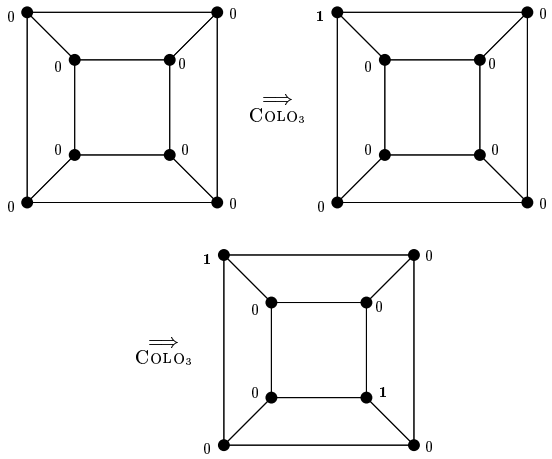
The figures below show an execution of COLO$_3$.
The initial labelling is the following:



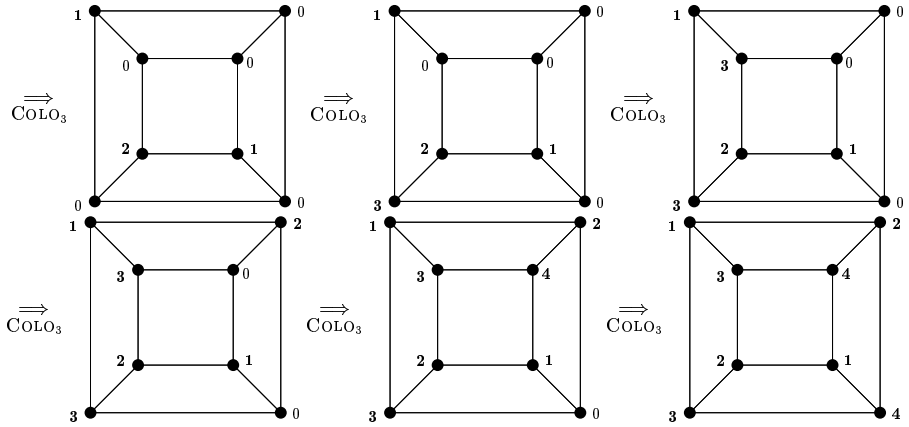
Two non-overlapping occurrences where a rule can be applied are:



A corresponding relabelling sequence is:



The remaining part of the relabelling sequence is, for instance:



One can note that the correctness of the algorithm follows from the fact that the set upon which the minimum is taken is never empty.

3.3. *Distributed Computations of Local Computations*

The notion of the relabelling sequence defined above obviously corresponds to a notion of *sequential* computation. Clearly, a locally generated relabelling relation allows parallel relabellings too, since non-overlapping balls may be relabelled independently. Thus we can define a distributed way of computing by saying that two consecutive relabelling steps with disjoint supports may be applied in any order (or concurrently). More generally, any two relabelling sequences such that one can be obtained from the other by exchanging successive concurrent steps, lead to the same result.

Hence, our notion of a relabelling sequence associated to a locally generated relabelling relation may be regarded as a *serialization* [M2] of a distributed computation. This model is asynchronous, in the sense that several relabelling steps *may* be done at the same time but we do not require that all of them have to be performed. In what follows we essentially handle sequential relabelling sequences, but the reader should keep in mind that such sequences may be done in parallel.

3.4. *Local Computations and Coverings*

We now present the fundamental lemma connecting coverings and locally generated relabelling relations [A]. It states that whenever $\mathbf{G}$ is a covering of $\mathbf{H}$, every relabelling step in $\mathbf{H}$ can be lifted to a relabelling sequence in $\mathbf{G}$, which is compatible with the covering relation.

Lemma 14 (Lifting Lemma). *Let $\mathcal{R}$ be a locally generated relabelling relation and let $\mathbf{G}$ be a covering of $\mathbf{H}$ via γ. If $\mathbf{H}\ \mathcal{R}^*\ \mathbf{H}'$, then there exists $\mathbf{G}'$ such that $\mathbf{G}\ \mathcal{R}^*\ \mathbf{G}'$ and $\mathbf{G}'$ is a covering of $\mathbf{H}'$ via γ.*

Proof. It suffices to show the claim for the case $\mathbf{H}\mathcal{R}\mathbf{H}'$. Suppose that the relabelling step changes labels in $B_H(v)$, for some vertex $v \in V(H)$. We may apply this relabelling step to each of the disjoint labelled balls of $\gamma^{-1}(B_H(v))$, since they are isomorphic to $B_H(v)$. This yields $\mathbf{G}'$ which satisfies the claim. □

This is depicted in the following commutative diagram:

$$\begin{array}{ccc} \mathbf{G} & \xrightarrow[\mathcal{R}^*]{} & \mathbf{G}' \\ {\scriptstyle\text{covering}}\downarrow & & \downarrow{\scriptstyle\text{covering}} \\ \mathbf{H} & \xrightarrow[\mathcal{R}^*]{} & \mathbf{H}' \end{array}$$

4. Labelled Graph Recognizers

Let $\mathcal{F}$ be some class of labelled graphs. We say that $\mathcal{F}$ is recognizable if there exists some locally generated relabelling relation such that starting from any labelled graph $\mathbf{G}$ some final labelling can be reached, allowing us to decide whether $\mathbf{G}$ belongs to $\mathcal{F}$ or not. We require that this decision depends only upon the set of final labels. We give the formal definition of this notion below, we explain how it may be applied to (unlabelled) graphs, and we present an example.

Definition 15. A *nondeterministic labelled graph recognizer* is a pair $(\mathcal{R}, \mathcal{K})$ where $\mathcal{R}$ is a noetherian, locally generated relabelling relation and there exists a recursive set K of finite subsets of L such that

$$\mathcal{K} = \{\mathbf{G} \in \mathcal{G}_L \mid lab(\mathbf{G}) \in K\}.$$

The set $\mathcal{K}$ is called the *final condition*.

Definition 16. Let $(\mathcal{R}, \mathcal{K})$ be a nondeterministic labelled graph recognizer. A labelled graph $\mathbf{G}$ is *recognized* by $(\mathcal{R}, \mathcal{K})$ if $\mathrm{Irred}_{\mathcal{R}}(\mathbf{G}) \cap \mathcal{K} \neq \emptyset$.

In the following sections we focus on deterministic recognizers (recognizers, for short). Section 10.1 considers the extension to nondeterministic recognizers.

Definition 17. A *labelled graph recognizer* $(\mathcal{R}, \mathcal{K})$ satisfies, in addition to Definition 15, the following condition: for each $\mathbf{G} \in \mathcal{G}_L$ we have either $\mathrm{Irred}_{\mathcal{R}}(\mathbf{G}) \cap \mathcal{K} = \emptyset$ or $\mathrm{Irred}_{\mathcal{R}}(\mathbf{G}) \subseteq \mathcal{K}$.

We have directly from the definitions:

Proposition 18. *Let $(\mathcal{R}, \mathcal{K})$ be a labelled graph recognizer. A labelled graph* $\mathbf{G}$ *is* recognized *by $(\mathcal{R}, \mathcal{K})$ if and only if* $\mathrm{Irred}_{\mathcal{R}}(\mathbf{G}) \subseteq \mathcal{K}$.

Definition 19. A class $\mathcal{F}$ of labelled graphs is *recognizable* if there exists a labelled graph recognizer $(\mathcal{R}, \mathcal{K})$ such that the set of recognized labelled graphs is $\mathcal{F}$.

4.1. *Unlabelled Graphs*

The notions above 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 labelled graph recognizer and l_0 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} is the labelling associating l_0 with all vertices and edges. Here l_0 can be any label and has no specific meaning.

A simple example of an unlabelled graph recognizer is given below.

Example 20 (Tree Recognition). The following graph relabelling system recognizes all trees. The set of labels is $L = \{N, \text{PRUNED}, \text{TREE}\}$. The initial label is $l_0 = N$. The set $\mathcal{K}$ is the set of labelled graphs having at least one vertex (this means the same as "exactly one vertex" because the graphs are connected) with the label TREE and the others having the label PRUNED, i.e., $\mathcal{K}$ is specified by $K = \{\{\mathit{TREE}\}, \{\mathit{PRUNED}, \mathit{TREE}\}\}$.

RECOG_TREE1: **Pruning rule**

Precondition:
- $\lambda(v_0) = N$.
- There exists a unique $v \in B(v_0)$ with $v \neq v_0$, $\lambda(v) = N$.

Relabelling:
- $\lambda'(v_0) :=$ PRUNED.

RECOG_TREE2: **Recognition rule**

Precondition:
- $\lambda(v_0) = N$.
- For all $v \in B(v_0)$, $v \neq v_0$: $\lambda(v) \neq N$.

Relabelling:
- $\lambda'(v_0) :=$ TREE.

We call any vertex labelled N having exactly one neighbour with the label N a pendant vertex. There are two meta-rules: RECOG_TREE1 and RECOG_TREE2. Meta-rule RECOG_TREE1 consists in cutting a pendant vertex by giving it the label PRUNED. The label N of a vertex v becomes TREE by meta-rule RECOG_TREE2 if the vertex v has no neighbour labelled N. A complete proof of this system may be found in [LMS2].

5. Characterizing Recognizable Classes of Labelled Graphs without Structural Knowledge

5.1. *The Necessary Condition*

Let $\overset{*}{\leftrightarrow}$ be the reflexive-symmetric-transitive closure of the covering relation $\rightarrow$. It is well known from Angluin's work [A] that recognizable graph classes must be closed under coverings (this follows directly from Lemma 14). A class $\mathcal{F}$ that is closed under $\overset{*}{\leftrightarrow}$ is said to be closed under the covering relation. We have from Lemma 14:

Lemma 21. *Let $(\mathcal{R}, \mathcal{K})$ be a labelled graph recognizer. Assume that* $\mathbf{G}$, $\mathbf{H}$ *are labelled graphs and* $\mathbf{G}$ *is a covering of* $\mathbf{H}$. *Then* $\mathbf{H}$ *is recognized by $(\mathcal{R}, \mathcal{K})$ if and only if* $\mathbf{G}$ *is recognized by $(\mathcal{R}, \mathcal{K})$.*

Proof. Consider a maximal relabelling sequence starting with $\mathbf{H}$. The relabelling system $\mathcal{R}$ is noetherian, hence the sequence is finite. Moreover, by Lemma 14 the relabelling sequence on $\mathbf{H}$ can be lifted to $\mathbf{G}$. Since $\mathcal{R}$ is locally generated, the lifting of the final form of $\mathbf{H}$on the relabelling sequence is also irreducible. By this way we obtain two irreducible graphs $\mathbf{G}'$ and $\mathbf{H}'$ such that $\mathbf{G}\ \mathcal{R}^*\ \mathbf{G}'$, $\mathbf{H}\ \mathcal{R}^*\ \mathbf{H}'$, and $\mathbf{G}'$ is a covering of $\mathbf{H}'$.

Suppose first that $\mathbf{H}$is recognized by $(\mathcal{R}, \mathcal{K})$. By definition, the irreducible graph $\mathbf{H}'$ belongs to K. Since $\mathbf{G}'$ has the same set of labels as $\mathbf{H}'$, it also belongs to K. Hence, $\mathbf{G}$ is recognized, too.

Suppose now that $\mathbf{G}$ is recognized. By definition, any maximal relabelling sequence on $\mathbf{G}$ yields an irreducible graph that belongs to $\mathcal{K}$. Hence, $\mathbf{G}' \in \mathcal{K}$ and $\mathbf{H}' \in \mathcal{K}$, too. Hence $\mathbf{H}$ is recognized, too. □

5.2. *Angluin's Condition Suffices for Recognizability*

The aim of this section is to prove the converse of Angluin's result, obtaining a characterization for recognizability:

Theorem 22. *Let $\mathcal{F}$ be a class of labelled graphs. The following statements are equivalent*:

(1) *$\mathcal{F}$ is recognizable.*
(2) *$\mathcal{F}$ is recursive and closed under $\overset{*}{\leftrightarrow}$.*

The proof of the implication (1) $\Longrightarrow$ (2) is simple. The closure follows from Lemma 21. Let $\mathcal{F}$ be recognized by the labelled graph recognizer $(\mathcal{R}, \mathcal{K})$, and let $\mathbf{G}$ be a labelled graph. Since $\mathcal{R}$ is recursive and the set of irreducible labelled graphs is recursive, an element $\mathbf{H}$ of $\mathrm{Irred}_{\mathcal{R}}(\mathbf{G})$ can be computed as follows: test whether $\mathbf{G}$ is irreducible; if

not, then (recursively) enumerate $\mathcal{R}$ to find a pair $\mathbf{G}\,\mathcal{R}\,\mathbf{G}'$ and continue with $\mathbf{G}'$. Then test whether $lab(\mathbf{H})$ is in K, the recursive set of finite subsets of L that specifies $\mathcal{K}$.

To prove the converse, we describe a labelled graph recognizer that recognizes a given $\overset{*}{\leftrightarrow}$-closed recursive class of graphs. Our result uses an enumeration algorithm given by Mazurkiewicz in [M3]. It is worth noting that despite its simplicity, this algorithm provides all the information that a distributed algorithm can be required to compute [G].

This proof is addressed in the following way. First we present Mazurkiewicz' enumeration algorithm. Then we characterize the final labellings obtained with this algorithm.

We prove that Mazurkiewicz' algorithm applied to any graph $\mathbf{G}$ makes it possible to compute a graph $\mathbf{H}$ such that $\mathbf{G}$ is a covering of $\mathbf{H}$. Since $\mathcal{F}$ is closed under $\overset{*}{\leftrightarrow}$, we have $\mathbf{G} \in \mathcal{F}$ if and only if $\mathbf{H} \in \mathcal{F}$.

5.3. *Mazurkiewicz' Enumeration Algorithm*

A distributed enumeration algorithm on a graph $\mathbf{G}$ is a distributed algorithm such that the result of any computation is a labelling of the vertices that is a bijection from $V(G)$ to $\{1, 2, \ldots, |V(G)|\}$. In particular, an enumeration of the vertices where vertices know whether the algorithm has terminated solves the election problem. In [M3] Mazurkiewicz presents a distributed enumeration algorithm for covering-minimal (non-ambiguous) graphs.

The computation model in [M3] consists exactly in relabelling balls of radius 1 and the initial graph is unlabelled.

Mazurkiewicz' algorithm is denoted by $\mathcal{M}$. By abuse of language we still speak of an enumeration algorithm, even when it is applied to ambiguous graphs (for which no enumeration algorithm exists [M3]). The final labellings that are incorrect from the enumeration point of view have interesting properties in the context of recognition. Namely, they determine a graph that is covered by the input graph.

In the following we describe Mazurkiewicz' algorithm including its extension to labelled graphs.

Enumeration Algorithm. We first give a general description of the algorithm $\mathcal{M}$ applied to a graph $\mathbf{G}$. Let $\mathbf{G} = (G, \lambda)$ and consider a vertex v_0 of G, and the set $\{v_1, \ldots, v_d\}$ of neighbours of v_0. The label of the vertex v_0 used by $\mathcal{M}$ is the pair $(\lambda(v_0), c(v_0))$ where $c(v_0)$ is a triple $(n(v_0), N(v_0), M(v_0))$ representing the following information obtained during the computation (formal definitions are given below):

- $n(v_0) \in \mathbb{N}$ is the *number* of the vertex v_0 computed by the algorithm.
- $N(v_0) \in \mathcal{N}$ is the *local view* of v_0, and it is either empty or a family of triples defined by

  $$\{(n(v_i), \lambda(v_i), \lambda(\{v_0, v_i\}))|1 \leq i \leq d\}.$$

- $M(v_0) \subseteq L \times \mathbb{N} \times \mathcal{N}$ is the *mailbox* of v_0 and contains the whole information received by v_0 at any step of the computation.

Each vertex v attempts to get its own number $n(v)$, which will be an integer between 1 and $|V(G)|$. A vertex chooses a number and broadcasts it together with its label and

its labelled neighbourhood all over the network. If a vertex u discovers the existence of another vertex v with the same number, then it compares its label and its local view, i.e., its number-labelled ball, with the local view of its rival v. If the label of v or the local view of v is "stronger", then u chooses another number. Each new number, with its local view, is broadcast again over the network. At the end of the computation it is not guaranteed that every vertex has a unique number, unless the graph is covering-minimal. However, all vertices with the same number will have the same label and the same local view.

The crucial property of the algorithm is based on a total order on local views such that the local view of any vertex cannot decrease during the computation. We assume for the rest of this paper that the set of labels L is totally ordered by $<_L$. Consider a vertex v_0 with neighbourhood $\{v_1, \ldots, v_d\}$ and assume that

- $n(v_1) \geq n(v_2) \geq \cdots \geq n(v_d)$,
- if $n(v_i) = n(v_{i+1})$, then $\lambda(v_i) \geq_L \lambda(v_{i+1})$,
- if $n(v_i) = n(v_{i+1})$ and $\lambda(v_i) = \lambda(v_{i+1})$, then $\lambda(\{v_0, v_i\}) \geq_L \lambda(\{v_0, v_{i+1}\})$.

Then the local view $N(v)$ is the d-tuple

$$((n(v_1), \lambda(v_1), \lambda(\{v_0, v_1\})), \ldots, (n(v_d), \lambda(v_d), \lambda(\{v_0, v_d\}))).$$

Let $\mathcal{N}_>$ be the set of all such ordered tuples. We define a total order $\prec$ on $\mathcal{N}_>$ by comparing the numbers, then the vertex labels, and finally the edge labels. Formally, for two elements

$$((n_1, l_1, e_1), \ldots, (n_d, l_d, e_d)) \quad \text{and} \quad ((n'_1, l'_1, e'_1), \ldots, (n'_{d'}, l'_{d'}, e'_{d'}))$$

of $\mathcal{N}_>$ we define

$$((n'_1, l'_1, e'_1), \ldots, (n'_{d'}, l'_{d'}, e'_{d'})) \prec ((n_1, l_1, e_1), \ldots, (n_d, l_d, e_d))$$

if one of the following conditions holds:

1. $n_1 = n'_1, \ldots, n_{i-1} = n'_{i-1}$ and $n'_i < n_i$ for some i.
2. $d' < d$ and $n_1 = n'_1, \ldots, n_{d'} = n'_{d'}$.
3. $d = d', n_1 = n'_1, \ldots, n_d = n'_d$ and $l_1 = l'_1, \ldots, l_{i-1} = l'_{i-1}$ and $l'_i <_L l_i$ for some i.
4. $d = d'$ and $n_1 = n'_1, \ldots, n_d = n'_d$ and $l_1 = l'_1, \ldots, l_d = l'_d$ and $e_1 = e'_1, \ldots, e_{i-1} = e'_{i-1}$, and $e'_i <_L e_i$ for some i.

If $N(u) \prec N(v)$, then we say that the local view $N(v)$ of v is stronger than the one of u. The order $\prec$ is a total order on $\mathcal{N} = \mathcal{N}_> \cup \{\emptyset\}$, with, by definition, $\emptyset \prec N$ for every $N \in \mathcal{N}_>$.

We now describe the algorithm through a graph relabelling system. The initial labelling of the vertex v_0 is $(\lambda(v_0), (0, \emptyset, \emptyset))$.

The rules are described below for a given ball $B(v_0)$ with centre v_0. The vertices v of $B(v_0)$ have labels $(\lambda(v), (n(v), N(v), M(v)))$. The labels obtained after applying a rule are $(\lambda(v), (n'(v), N'(v), M'(v)))$. We recall that we omit labels that are unchanged.

$\mathcal{M}$-1: **Diffusion rule**

Precondition:

- There exists $v \in B(v_0)$ such that $M(v) \neq M(v_0)$.

Relabelling:

- For all $v \in B(v_0)$, $M'(v) := \bigcup_{w \in B(v_0)} M(w)$.

$\mathcal{M}$-2: **Renaming rule**

Precondition:

- For all $v \in B(v_0)$, $M(v) = M(v_0)$.
- $(n(v_0) = 0)$ or $(n(v_0) > 0$ and there exists $(l, n(v_0), N) \in M(v_0)$ such that $(\lambda(v_0) <_L l)$ or $((\lambda(v_0) = l)$ and $(N(v_0) \prec N)))$.

Relabelling:

- $n'(v_0) := 1 + \max\{n \in \mathbb{N} \mid (l, n, N) \in M(v_0)$ for some $l, N\}$.
- For every $v \in B(v_0)$, $N'(v)$ is obtained from $N(v)$ by replacing the value of $n(v_0)$ by $n'(v_0)$.
- For every $v \in B(v_0)$, the mailbox contents $M(v)$ changes to $M'(v) := M(v) \cup \{(\lambda(w), n'(w), N'(w)) \mid w \in B(v_0)\}$.

5.4. *Properties of Mazurkiewicz' Algorithm*

We present a complete proof of Mazurkiewicz' algorithm in our framework following the ideas developed in [M3].

Let $\mathbf{G}$ be a labelled graph. If v is a vertex of G, then the label of v after a run ρ of Mazurkiewicz' algorithm is denoted $(\lambda(v), c_\rho(v))$ with $c_\rho(v) = (n_\rho(v), N_\rho(v), M_\rho(v))$ and (λ, c_ρ) denotes the final labelling.

Theorem 23 [M3]. *Any run ρ of Mazurkiewicz' enumeration algorithm on a connected labelled graph $\mathbf{G} = (G, \lambda)$ terminates and yields a final labelling (λ, c_ρ) verifying the following conditions for all vertices v, v' of G:*

(1) *Let m be the maximal number in the final labelling, $m = \max_{v \in V(G)} n_\rho(v)$. Then for every $1 \leq p \leq m$ there is some $v \in V(G)$ with $n_\rho(v) = p$.*
(2) $M_\rho(v) = M_\rho(v')$.
(3) $(\lambda(v), n_\rho(v), N_\rho(v)) \in M_\rho(v')$.
(4) *Let $(l, n, N) \in M_\rho(v')$. Then $\lambda(v) = l$, $n_\rho(v) = n$, and $N_\rho(v) = N$ for some vertex v if and only if there is no triple $(l', n, N') \in M_\rho(v')$ with $l <_L l'$ or $(l = l'$ and $N \prec N')$.*
(5) *$n_\rho(v) = n_\rho(v')$ implies $(\lambda(v) = \lambda(v')$ and $N(v) = N(v'))$.*
(6) *n_ρ induces a locally bijective labelling of G.*

The proof below follows the lines of [M3]. It needs the following lemmas. We say that a number m is known by v if $(l, m, N) \in M(v)$ for some l and some N. In the following i is an integer denoting a computation step. Let $(\lambda(v), (n_i(v), N_i(v), M_i(v)))$ be the label of vertex v after the ith step of the computation.

Lemma 24. *For each v, i:*

- $n_i(v) \leq n_{i+1}(v)$.
- $N_i(v) \preceq N_{i+1}(v)$.
- $M_i(v) \subseteq M_{i+1}(v)$.

Proof. The property is obviously true for the vertices that are not involved in the rule applied at step i. For the other vertices we note that the *renaming rule* applied to v_0 increments $n_i(v_0)$, adds elements to some mailboxes, and makes some $N(u)$ stronger. Moreover, the *diffusion rule* only adds elements to mailboxes.

The fact that $N_i(v) \preceq N_{i+1}(v)$ comes from the definition of $\prec$. In other words, this order ensures that the past local views of a vertex are always weaker than its present one.

Furthermore, one of the inequalities is strict for at least one vertex, namely the one for which the previous rule was applied. □

Lemma 25. *For every $v \in V(G)$ and $(l, m, N) \in M_i(v)$ there exists a vertex $w \in V(G)$ such that $n_i(w) = m$.*

Proof. Assume that the number m is known by v and let $U = \{u \in V(G) \mid \exists j < i, n_j(u) = m\}$. Obviously U is not empty. Let $w \in U$ and let $j < i$ such that

1. $n_j(w) = m$,
2. for any $u \in U$ and for any $k < i$ verifying $n_k(u) = m$ we have $N_k(u) \preceq N_j(w)$.

Clearly, the *renaming rule* cannot be applied to w, hence $n_i(w) = m$. □

Next, we claim that whenever a number is known, all positive smaller numbers are assigned to some vertex.

Lemma 26. *For every vertex $v \in V(G)$ such that $n_i(v) \neq 0$ and for every $m \in [1, n_i(v)]$, there exists some vertex $w \in V(G)$ such that $n_i(w) = m$.*

Proof. We show this claim by induction on i. At the initial step ($i = 0$) the assertion is true. Suppose that it holds for $i \geq 0$. If the *diffusion rule* is used, the assertion is true for $i + 1$. If the *renaming rule* is applied to v_0, then we just have to verify it for v_0, and more precisely for all numbers m in the interval $\{n_i(v_0), n_i(v_0) + 1, \ldots, n_{i+1}(v_0)\}$. The property holds obviously for $n_{i+1}(v_0)$ and, being known by v_0 at step $i + 1$, the property for $n_i(v_0)$ is a consequence of Lemma 25.

If the interval $\{n_i(v_0)+1, \ldots, n_{i+1}(v_0)-1\}$ is empty, then the condition is obviously satisfied. Otherwise by definition of the *renaming rule*, $n_{i+1}(v_0)-1$ is known by v_0 at step i and thus Lemma 25 implies that there exists $w \neq v_0$ such that $n_i(w) = n_{i+1}(v_0) - 1$. For every $m \in \{n_i(v_0) + 1, \ldots, n_{i+1}(v_0) - 1\}$, we have, by the induction hypothesis on w that there exists a vertex $x \in V(G)$ such that $n_i(x) = m$. For every such x, because v_0 is the only vertex changing its name from step i to $i + 1$, $n_i(x) = n_{i+1}(x)$, which proves the assertion for step $i + 1$. □

We now show Theorem 23:

Proof. As before, we denote by $(\lambda(v), (n_i(v), N_i(v), M_i(v)))$ the label of vertex v after the ith step of the computation.

As there are no more than $|V(G)|$ different numbers assigned it follows from Lemmas 24 and 26 that the algorithm terminates.

Properties 1–6 of the final labelling are easily derived from the above part of the proof:

1. By Lemma 26 applied to the final labelling.
2. Otherwise, the *diffusion rule* could be applied.
3. A direct corollary of the previous property.
4. We have obtained a final labelling, thus it is a direct consequence of the diffusion rule and of the precondition of the renaming rule.
5. A direct consequence of the previous point.
6. The first part of Definition 6 is a consequence of the rewriting mechanism: when a vertex v is numbered, its number is put in mailboxes of adjacent vertices. Thus vertices at distance 2 of v cannot have the same number as v. The second part of Definition 6 is a consequence of the precondition of the renaming rule: the *renaming rule* could have been applied to vertices having the same number and non-isomorphic local views.

This ends the proof of the theorem. □

Remark 27. By conditions (1) and (6) of Theorem 23, and similarly to [M3], the algorithm computes, for non-ambiguous graphs (and thus for minimal graphs by Corollary 9), a one-to-one correspondence n_ρ between the set of vertices of G and the set of integers $\{1, \ldots, |V(G)|\}$.

5.5. *Mazurkiewicz' Algorithm and Network Cartography*

In this section we prove that even by applying Mazurkiewicz' algorithm to a graph **G** that is not covering-minimal, we can get some relevant information. In this case we prove that we can interpret the mailbox of the final labelling as a graph **H** that each vertex can compute and such that **G** is a covering of **H**.

For a mailbox M, we define the graph of the "strongest" vertices as follows. First, for $l \in L, n \in \mathbb{N}, N \in \mathcal{N}, M \subseteq L \times \mathbb{N} \times \mathcal{N}$, we define the predicate $\underline{\text{Strong}}(l, n, N, M)$ that is true if $(l, n, N) \in M$ and there is no $(l', n, N') \in M$ verifying

$$l' >_L l \quad \text{or} \quad (l = l' \text{ and } N \prec N').$$

The graph H_M of strongest vertices of M is then defined by

$$V(H_M) = \{n \mid \exists N, l : \underline{\text{Strong}}(l, n, N, M)\},$$
$$E(H_M) = \{\{n, n'\} \mid \exists N, l : \underline{\text{Strong}}(l, n, N, M), \text{ and } \exists l', l'' : N = (\ldots, (n', l', l''), \ldots)\}.$$

We also define a labelling on this graph by $\lambda_M(n) = l$, with $\underline{\text{Strong}}(n, l, N, M)$ for some N, and $\lambda_M(\{n, n'\}) = l''$, with $\underline{\text{Strong}}(n, l, N, M)$ and $N = (\ldots, (n', l', l''), \ldots)$.

The uniqueness of this definition comes from the definition of Strong and from Theorem 23(5).

Let ρ be a run of $\mathcal{M}$. Then $(H_{M_\rho(u)}, \lambda_{M_\rho(u)})$ does not depend on u by Theorem 23(2). We then define $\rho(\mathbf{G}) = (H_{M_\rho(u)}, \lambda_{M_\rho(u)})$, for any vertex u. By Theorem 23(4), we have:

Proposition 28. *For a given execution ρ of Mazurkiewicz' algorithm,*

$$V(\rho(\mathbf{G})) = \{n_\rho(v) \mid v \in V(G)\} \quad \textit{and}$$

$$E(\rho(\mathbf{G})) = \{\{n_\rho(v), n_\rho(w)\} \mid \{v, w\} \in E(G)\}.$$

The proposition states that $\rho(\mathbf{G})$ is the quotient graph of G by n_ρ, see Section 2.4.

Remark 29. Before we emphasize the role of $\rho(\mathbf{G})$, note that $\rho(\mathbf{G})$ can be locally computed by every vertex, and that the graph depends only on the label M_ρ.

The next proposition states that we can see a run of $\mathcal{M}$ as computing a graph covered by $\mathbf{G}$. Conversely, as a "translation" from Theorem 5 of [M3], every graph covered by $\mathbf{G}$ can be obtained by a run of the algorithm.

Proposition 30. *Let* $\mathbf{G}$ *be a labelled graph.*

(1) *For all runs ρ of $\mathcal{M}$, $\mathbf{G}$ is a covering of $\rho(\mathbf{G})$.*
(2) (*fairness*) *For all $\mathbf{H}$ such that $\mathbf{G}$ is a covering of $\mathbf{H}$, there exists a run ρ such that $\mathbf{H} \simeq \rho(\mathbf{G})$.*

Proof. (1) Since n_ρ is locally bijective (Theorem 23(6)), we obtain from Lemma 8 that $\mathbf{G}$ is a covering of $\rho(\mathbf{G})$.

(2) We exhibit a run of $\mathcal{M}$ having the required property. Suppose that we have an enumeration of the vertices of $\mathbf{H}$. Let μ be the labelling of G obtained by lifting the enumeration. There is an execution of Mazurkiewicz' algorithm such that each vertex v of G gets $\mu(v)$ as a final n_ρ-labelling.

This is done in the following way. First we apply the renaming rule to all vertices in $\mu^{-1}(1)$. This is possible because there is no overlapping of balls, since $\mathbf{G}$ is a covering of $\mathbf{H}$. Then we apply the diffusion rule as long as we can. After that, we apply the renaming rule to $\mu^{-1}(2)$. Because of the diffusion, the number 1 is known by all the vertices, so the vertices of $\mu^{-1}(2)$ get labelled by 2, and so on, until each vertex v gets labelled by $\mu(v)$. □

From Proposition 30(1), we can see a run of $\mathcal{M}$ as computing a covering. Furthermore, if the underlying graph is covering-minimal, then $\rho(\mathbf{G})$ is an isomorphic copy of $\mathbf{G}$. This copy can be computed from their mailbox by any vertex, providing a "map"—with numbers of identification—of the underlying network. Thus, on minimal networks, the algorithm of Mazurkiewicz can actually be seen as a *cartography algorithm*.

5.6. *Proof of Theorem* 22

In this subsection we complete the proof of Theorem 22 by describing a labelled graph recognizer for recognizing a given recursive, $\overset{*}{\leftrightarrow}$-closed class of labelled graphs $\mathcal{F}$.

In order to recognize the given graph $\mathbf{G}$ we use graph $\rho(\mathbf{G})$ computed by M. We have:

Proposition 31. *Let $\mathcal{F}$ be a $\overset{*}{\leftrightarrow}$-closed class of labelled graphs and let $\mathbf{G}$ be a labelled graph. Then for any run ρ of $\mathcal{M}$,*

$$\mathbf{G} \in \mathcal{F} \qquad \text{if and only if} \quad \rho(\mathbf{G}) \in \mathcal{F}.$$

Proof. From Proposition 30(1), we get that $\mathbf{G} \overset{*}{\leftrightarrow} \rho(\mathbf{G})$ for any run ρ. □

Thus, in order to complete the labelled graph recognizer, we add a label $r \in \{?, \text{TRUE}, \text{FALSE}\}$, that will contain the result of the computation given by the following testing rule:

TEST:
Precondition:
• $r(v_0) = ?$
Relabelling:
• Compute $(\mathbf{H_{M(v_0)}}, \lambda_{\mathbf{M(v_0)}})$.
• Set $r(v_0)$ to the result of the test "$(H_{M(v_0)}, \lambda_{M(v_0)}) \in \mathcal{F}$?".

The labelled graph recognizer will be the following:

– labels: (λ, n, N, M, r);
– rules:
  • an initial relabelling rule that converts, for every vertex v, the $\lambda(v)$ label to $(\lambda(v), 0, \emptyset, \emptyset, ?)$ on all vertices v;
  • the M rules run on $(\lambda, (n, N, M))$, setting $r(v_0)$ to ? each time a rule is applied to v_0;
  • the TEST rule;
– final condition: label r is TRUE on all vertices.

The set of rules expressed by the generic rule TEST defines a valid graph relabelling system because $\mathcal{F}$ is recursive.

6. A Necessary Condition for Recognizability with Structural Knowledge

We are interested in recognizing labelled graphs which have a certain structural knowledge encoded in the initial labelling. Let $\mathbf{G} = (G, \lambda)$ be a labelled graph and let α be a label. Then Λ_α is the uniform labelling on $\mathbf{G}$ with label α, that is, every vertex v is labelled by the pair $(\alpha, \lambda(v))$, the labels of the edges are not changed. The labelled graph $\mathbf{G}$ has some structural information encoded by its labels (a distinguished vertex,

identities) and α can encode some structural properties of the graph, like the size or a bound of the diameter of the graph.

Recognition with structural knowledge is defined as follows:

Definition 32. A *nondeterministic labelled graph recognizer with structural knowledge* ι is a triple $(\mathcal{R}, \mathcal{K}, \iota)$ where $\mathcal{R}$ is a locally generated relabelling relation that is noetherian on the set $\{(G, \Lambda_{\iota(\mathbf{G})}) \mid \mathbf{G} \in \mathcal{G}_L\}$, $\mathcal{K}$ is the final condition (i.e., there exists a recursive set K of finite subsets of L such that $\mathcal{K} = \{\mathbf{G} \in \mathcal{G}_L \mid lab(\mathbf{G}) \in K\}$), and ι is a computable function which associates with each labelled graph $\mathbf{G}$ a label $\iota(\mathbf{G}) \in L$.

A labelled graph $\mathbf{G}$ is *recognized* by $(\mathcal{R}, \mathcal{K}, \iota)$ if $\mathrm{Irred}_{\mathcal{R}}((G, \Lambda_{\iota(\mathbf{G})})) \cap \mathcal{K} \neq \emptyset$.

As previously stated, we first focus on deterministic recognizers.

Definition 33. A *labelled graph recognizer with structural knowledge* ι satisfies, in addition to Definition 32 for each $\mathbf{G} \in \mathcal{G}_L$, either $\mathrm{Irred}_{\mathcal{R}}((\mathrm{G}, \Lambda_{\iota(\mathbf{G})})) \cap \mathcal{K} = \emptyset$ or $\mathrm{Irred}_{\mathcal{R}}((\mathrm{G}, \Lambda_{\iota(\mathbf{G})})) \subseteq \mathcal{K}$.

For such a recognizer, a labelled graph $\mathbf{G}$ is recognized if and only if $\mathrm{Irred}_{\mathcal{R}}(\mathbf{G}) \subseteq \mathcal{K}$. A class $\mathcal{F}$ of labelled graphs is *recognizable with structural knowledge* ι (or, informally, recognizable knowing ι) if there exists a labelled graph recognizer with structural knowledge $(\mathcal{R}, \mathcal{K}, \iota)$ such that the set of labelled graphs that are recognized by $(\mathcal{R}, \mathcal{K}, \iota)$ is $\mathcal{F}$.

6.1. *Examples of Structural Knowledge*

Some examples of structural knowledge are given below:

- An upper bound b on the size of the graph: this is a function b such that $b(G) \geq |V(G)|$ for all G. We say that a class $\mathcal{F}$ is *upper bound recognizable* if there exists $(\mathcal{R}, \mathcal{K})$ such that $\mathcal{F}$ is recognized by $(\mathcal{R}, \mathcal{K}, b)$ for all upper bounds b.
- A *tight upper bound*: this is an upper bound b such that for all graphs G, $|V(G)| \leq b(G) < 2|V(G)|$.
- The size, i.e., the number of vertices $|V(G)|$. In this case a class of graphs is said to be *size-recognizable*.
- The diameter $\Delta(G)$ of the graph.
- The topology of the graph (e.g., the adjacency matrix is given).

If no structural knowledge at all is available, like in Section 5, then we use the particular label ε.

Remark 34. Knowledge of the topology obviously enables the recognition of any recursive graph class but it does not solve all the problems. In particular, it does not enable us to solve the election problem in all cases [GM2]. For the cartography problem, it remains to compute the correspondence between vertices of the concrete graph and vertices of the abstract graph given by the structural knowledge.

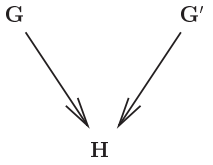


Fig. 2. The new covering technique.

6.2. *A Generalized Covering Technique*

We start with the following proposition stating how recognizability with structural knowledge can be carried over by coverings (see Figure 2). This is similar to the lifting technique used in Lemmas 14 and 21.

Proposition 35. *Let $(\mathcal{R}, \mathcal{K}, \iota)$ be a labelled graph recognizer with structural knowledge ι. Let $\mathbf{G}, \mathbf{G}'$ be graphs such that $\iota(\mathbf{G}) = \iota(\mathbf{G}')$ and there exists $\mathbf{H}$ such that both $\mathbf{G}$ and $\mathbf{G}'$ are coverings of $\mathbf{H}$. Then $\mathbf{G}$ is recognized by $(\mathcal{R}, \mathcal{K}, \iota)$ if and only if $\mathbf{G}'$ is recognized by $(\mathcal{R}, \mathcal{K}, \iota)$.*

Proof. Suppose that $\mathbf{G}$ is recognized by $(\mathcal{R}, \mathcal{K}, \iota)$. Then $\mathrm{Irred}_{\mathcal{R}}(G, \Lambda_{\iota(\mathbf{G})}) \subseteq \mathcal{K}$.

As $\mathbf{G}$ is a covering of $\mathbf{H}$, $(G, \Lambda_{\iota(\mathbf{G})})$ is a covering of $(H, \Lambda_{\iota(\mathbf{G})})$; let γ the corresponding morphism. For the same reason $(G', \Lambda_{\iota(\mathbf{G}')})$ is a covering of $(H, \Lambda_{\iota(\mathbf{G})})$; let γ' the corresponding morphism.

If we apply a relabelling step on $B_{(G,\Lambda_{\iota(\mathbf{G})})}(v)$, then we apply the same relabelling step on $B_{(H,\Lambda_{\iota(\mathbf{G})})}(\gamma(v))$ and on each $B_{(G,\Lambda_{\iota(\mathbf{G})})}(v')$ and $B_{(G',\Lambda_{\iota(\mathbf{G}')})}(w')$ where v' belongs to $\gamma^{-1}(\gamma(v))$ $(v' \neq v)$ and w' belongs to $\gamma'^{-1}(\gamma(v))$.

Since $\mathcal{R}$ is noetherian on $(G, \Lambda_{\iota(\mathbf{G})})$, by applying this strategy an irreducible graph is obtained from $(G, \Lambda_{\iota(\mathbf{G})})$. The corresponding graph obtained from $(G', \Lambda_{\iota(\mathbf{G}')})$ must be irreducible too, for reasons of symmetry; moreover, it has the same set of labels. Hence $\mathrm{Irred}_{\mathcal{R}}(G', \Lambda_{\iota(\mathbf{G}')}) \cap \mathcal{K} \neq \emptyset$. Thus, by definition, $\mathbf{G}'$ is recognized by $(\mathcal{R}, \mathcal{K}, \iota)$. □

Remark 36. The value of $\iota(\mathbf{H})$ above has no importance, and actually we cannot say that $\mathbf{H}$ is recognized by $(\mathcal{R}, \mathcal{K}, \iota)$, for $\iota(\mathbf{H})$ is not necessarily equal to $\iota(\mathbf{G})$ (for example, if ι denotes the size).

The previous result can be generalized as follows. We define a relation σ^ι by letting $\mathbf{G}\,\sigma^\iota\,\mathbf{G}'$ if

- $\iota(\mathbf{G}) = \iota(\mathbf{G}')$,
- there exists a graph $\mathbf{H}$ such that $\mathbf{G}$ and $\mathbf{G}'$ are both coverings of $\mathbf{H}$.

Let $\sim^\iota$ denote the reflexive, transitive closure of σ^ι. A class of graphs $\mathcal{F}$ is said to be closed under $\sim^\iota$ if for any graphs $\mathbf{G}$ and $\mathbf{G}'$ such that $\mathbf{G} \sim^\iota \mathbf{G}'$, $\mathbf{G}$ belongs to $\mathcal{F}$ if and only if $\mathbf{G}'$ belongs to $\mathcal{F}$. With Proposition 35 and the definition of $\sim^\iota$, we can state:

Proposition 37. *Let $(\mathcal{R}, \mathcal{K}, \iota)$ be a labelled graph recognizer with structural knowledge ι. Suppose that $\mathbf{G} \sim^{\iota} \mathbf{G}'$. Then $\mathbf{G}$ is recognized by $(\mathcal{R}, \mathcal{K}, \iota)$ if and only if $\mathbf{G}'$ is recognized by $(\mathcal{R}, \mathcal{K}, \iota)$.*

Corollary 38 (A Necessary Condition for Recognizability). *Let $\mathcal{F}$ be a class of graphs which is recognizable with structural knowledge ι. Then $\mathcal{F}$ is closed under $\sim^{\iota}$.*

Before showing the converse of the previous corollary in Section 8, we present, in Sections 6.3 and 7, some applications.

6.3. *Simple Applications or Easy Impossibility Proofs*

We apply Proposition 35 to the case where $\iota = \texttt{Size}$ represents the size of the unlabelled graph, i.e., $\iota(\mathbf{G}) = |V(G)|$ for every labelled graph $\mathbf{G}$.

Proposition 39. *The following graph classes are not size-recognizable*:

1. *bipartite graphs*,
2. *graphs having a cut-edge*,
3. *graphs having a cut-vertex*,
4. *Hamiltonian graphs*.

Proof. We apply Proposition 35 to the graphs $G \sim^{\texttt{Size}} G'$ (both are coverings of H). Note that only one of the covering graphs is in the given class. Thus, no labelled graph recognizer can recognize these classes.

A graph is bipartite if and only if it contains no odd cycles. Thus G' is bipartite, but G is not bipartite (Figure 3).

Graph G has no cut-edge and G' has two cut-edges (Figure 4).

Graph G has two cut-vertices and G' has no cut-vertex (Figure 5).

Graph G has the Hamilton property, but not G' (Figure 6). □

7. Recognizing Graph Minors

In [CM], it has been shown that no minor-closed class of graphs is recognizable without structural knowledge (with a few exceptions). In this section we prove that this is still the case even knowing the size. The exceptions are simple classes like trees, rings, or the entire class of graphs.

Let us recall some definitions related to minors. Contracting an edge between u and v consists of identifying u and v, deleting the resulting loop, and replacing multiple edges that may arise, by a simple edge. By deleting an edge we mean deleting the edge and the isolated vertices created. A graph G is a minor of a graph H (and we write $G \preceq H$) if there is a sequence of contractions and deletions of edges of H which leads to G. A class of graphs is called 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

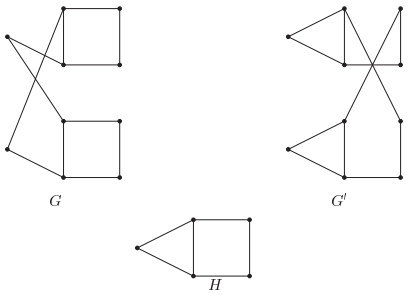


Fig. 3. Bipartite graphs are not size-recognizable.

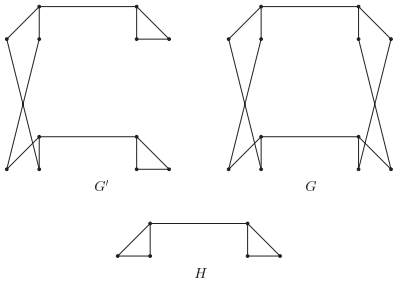


Fig. 4. Cut-edge is not size-recognizable.

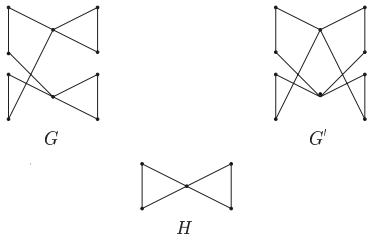


Fig. 5. Cut-vertex is not size-recognizable.

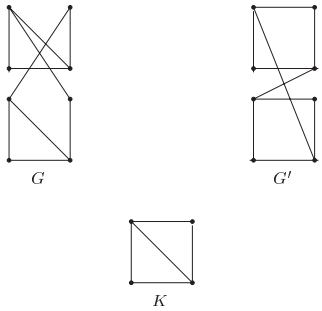


Fig. 6. The Hamilton property is not size-recognizable.

all connected minors of its elements. Minor-closed classes of connected graphs can be characterized by finite sets of connected forbidden minors, which are also called obstruction sets [RS], [F].

7.1. *The Covering Construction*

The first step is to prove the following:

Theorem 40. *Let K and H be two connected graphs and suppose that one of the following conditions holds*:

(1) *K is planar and H has at least two cycles.*
(2) *H contains at least three cycles.*

Then there exists an integer p such that for every $q \geq p$, there exists a connected q-sheeted covering G_1 of H that contains K as a minor.

Theorem 40 consists of two cases, the first one corresponds to Proposition 41 and the second one to Proposition 42. These propositions are similar to [CM].

For positive integers n and i, $\mathrm{mod}_n(i)$ is the unique integer belonging to $[0, n-1]$ such that $i = kn + \mathrm{mod}_n(i)$, for some non-negative integer k.

Proposition 41. *For every planar graph K and every connected graph H having at least two cycles, there exists an integer p such that for every $q \geq p$, there exists a connected q-sheeted covering G_1 of H that contains K as a minor.*

Proof. A torus is a graph of the form $T(m, n)$ with a set of vertices

$$V(m, n) = [0, m-1] \times [0, n-1],$$

and edges of the form

$$\{(x_1, y_1), (x_2, y_2)\} \quad \text{where either} \quad x_1 = x_2 \text{ and } y_2 = \mathrm{mod}_n(y_1 + 1)$$
$$\text{or} \quad y_1 = y_2 \text{ and } x_2 = \mathrm{mod}_m(x_1 + 1).$$

Figure 7 represents the torus $T(4, 5)$.

Since K is planar, there exist m and n such that $K \preceq T(m, n)$. This follows from the fact that every planar graph is a minor of some sufficiently large square grid, and the $m \times n$ grid is a subgraph of $T(m, n)$. More precisely, a planar graph with p vertices can be embedded into a $2p-4$ by $p-2$ grid, see [DFPP] and [vL].

Since H has at least two cycles, there exist two edges e and e' in H such that $H \backslash \{e, e'\}$ is connected. Let $e = \{x, y\}$ and $e' = \{z, u\}$. We may have $y = z$ but not $e = e'$. Let H' be the graph obtained from H by deleting e and e'; let H'' be the graph obtained from H by deleting e'. See Figure 8.

For $q \geq mn$ we construct a connected graph G_1 such that $K \preceq G_1$ and G_1 is a q-sheeted covering of H. It consists of mn disjoint copies of H' linked in a torus-like fashion and adding a "tail" of $q - mn$ disjoint copies of H''.

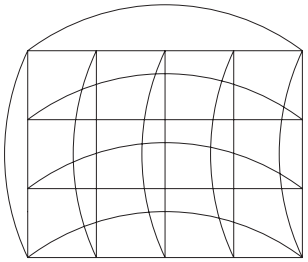


Fig. 7. Torus $T(4, 5)$.

See Figure 9 for an example of this construction with $m = 5$, $n = 4$, and $q = 23$. More precisely, for every $(i, j) \in V(m, n)$, we let $H'(i, j)$ be an isomorphic copy of H' such that

$$V(H'(i, j)) \cap V(H'(i', j')) = \emptyset \qquad \text{if} \quad (i, j) \neq (i', j').$$

For every $0 \leq l \leq (q - p - 1)$ we let $H''(l)$ be an isomorphic copy of H'' such that

$$V(H'(i, j)) \cap V(H''(l)) = \emptyset,$$

$$V(H''(l)) \cap V(H''(l')) = \emptyset \qquad \text{if} \quad l \neq l'.$$

We denote by $w(i, j)$ the copy in $H'(i, j)$ of a vertex w of H', and by $w(l)$ the copy in $H''(l)$ of a vertex w of H''.

We let G_1 consist of:

- The union of the graphs $H'(i, j)$ and $H''(l)$.
- The following edges:
  Torus part:
  - $\{x(i, j), y(i, \mathrm{mod}_n(j + 1))\}$, $1 \leq i \leq m - 1$, $0 \leq j \leq n - 1$,
  - $\{z(i, j), u(\mathrm{mod}_m(i + 1), j)\}$, $0 \leq i \leq m - 1$, $0 \leq j \leq n - 1$, $(i, j) \neq (m - 1, 0)$.

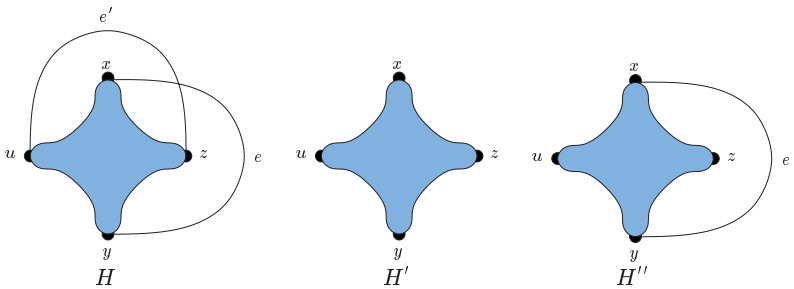


Fig. 8. H, H', and H''.

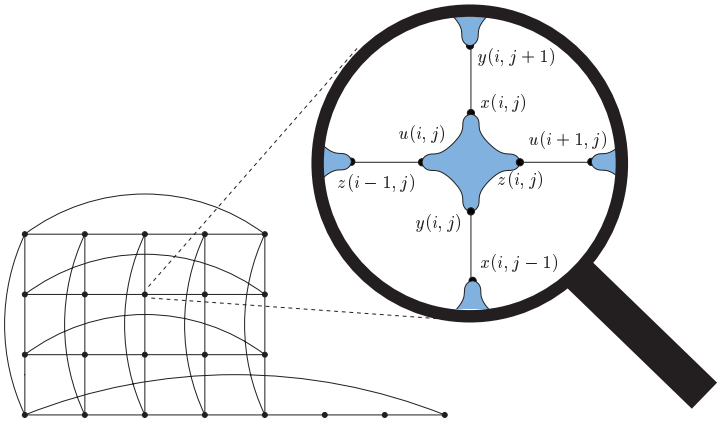


Fig. 9. G_1 seen with a magnifying glass.

Tail part:

- $\{z(m-1, 0), u(0)\}$,
- $\{z(i), u(i+1)\},\ 0 \le i \le q - mn - 2$,
- $\{z(q - mn - 1), u(0, 0)\}$.

It is clear that by contracting simultaneously all edges of the subgraphs $H'(i, j)$ and $H''(l)$ of G_1 and then contracting all the edges of the "tail", we get $T(m, n)$. Hence $K \trianglelefteq T(m, n) \trianglelefteq G_1$. The mapping that replaces a vertex $w(i, j)$ in $H'(i, j)$ or a vertex $w(l)$ in $H''(l)$ by w in H is a covering, thus G_1 is a q-sheeted covering of H. □

The other case of Theorem 40 is treated similarly.

Proposition 42. *For every graph K and every connected graph H having at least three cycles, there exists an integer p such that for every $q \ge p$ there exists a connected q-sheeted covering G_1 of H that contains K as a minor.*

Proof. The proof is an easy extension of the previous one. Instead of embedding K as a minor in a torus, we embed it as a minor in a sufficiently large "toroidal" parallelepiped $P(l, m, n)$ with the set of vertices $V(l, m, n) = [0, l-1] \times [0, m-1] \times [0, n-1]$.

We take $p = l * m * n$, and let $q \ge p$. We select three edges $e = \{x, y\}$, $e' = \{z, u\}$, and $e'' = \{v, w\}$ such that the graph H' obtained from H by deleting e, e', and e'' is connected. Let H'' be obtained from H by deleting e. We construct K as the union of disjoint copies $H'(i, j, k)$ of H' for $(i, j, k) \in V(l, m, n)$ and of disjoint copies $H''(l)$ for $0 \le l \le q - p - 1$ augmented with edges ad hoc (similarly to the previous proof). We obtain $K \trianglelefteq P(l, m, n) \trianglelefteq G_1$ and G_1 is a q-sheeted covering of H. □

7.2. *Size-Recognizability*

It is shown in [CM] that we cannot determine by local computations whether a graph does not contain a given (fixed) graph as a minor. The idea of the proof was to choose a graph H such that K is not a minor of H, and then to build a covering G_1 of H that contains K as a minor.

For size-recognizability we cannot use this construction because G_1 and H have different sizes. We introduce another construction (Figure 11) and we show that we cannot decide by local computations whether a given graph is a minor of another graph, even knowing the size of the latter. Thus, we show that, with a few simple exceptions (trees, rings, and chains, ...), minor-closed classes of graphs are not size-recognizable.

Our main result here is:

Theorem 43. *Let $\mathcal{F}$ be a minor-closed class of connected graphs that does not contain all connected graphs. Let K be a connected graph that is not in $\mathcal{F}$, with p vertices. If one of the following holds,*

(1) *$\mathcal{F}$ contains all planar connected graphs,*
(2) *K is planar and there exists a connected graph $H = (V, E)$ such that*
   – *H has a covering $G \in \mathcal{F}$ with at least $2p^2$ sheets, and*
   – *H contains at least two cycles,*

then $\mathcal{F}$ is not size-recognizable.

Proof. Suppose that $\mathcal{F}$ is size-recognizable.

(1) If $\mathcal{F}$ contains all planar connected graphs, then it contains the "three-leafed clover" S_3 and an infinite subclass of its planar coverings $\{S^q \mid q \in \mathbb{N}\}$. Here S^q is the q-sheeted covering of S_3 defined by Figure 10. If we apply Proposition 42 with $H = S_3$, we get G_1, a covering of S_3 such that $K \trianglelefteq G_1$. If q is the number of sheets of the covering

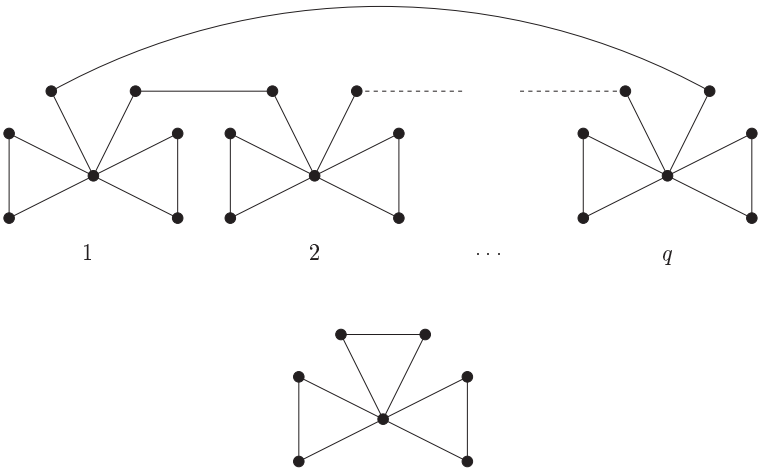


Fig. 10. S_3 and its planar coverings S^q.

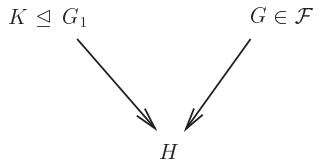


Fig. 11. Covering configuration for proof of Theorem 43.

G_1, then we get $G_1 \sim^{\mathtt{Size}} S^q$. Hence $G_1 \in \mathcal{F}$ by Corollary 38. As $\mathcal{F}$ is minor-closed, we get a contradiction with the fact that $K \notin \mathcal{F}$.

(2) The same covering configuration as above can now be applied, using Proposition 41 instead.

The covering configuration of the proof is given in Figure 11. □

Remark 44. The assumptions of the previous theorem are minimal in the following sense: if we allow H in case (2) to have at most one cycle, then $\mathcal{F}$ can be size-recognizable. Consider, e.g., the class of trees (Example 20) or the class of rings and chains (see Section 9.1). They are recognizable even not knowing the size, by the way.

Note also that there has to be a lower bound on the number of sheets of G because if it is not large enough, then the class F can be size-recognizable. The class of graphs with at most n vertices (for a given n) is minor-closed and obviously size-recognizable.

8. Enumeration and Recognition

8.1. *Recognizing $\sim^\iota$-Closed Graph Classes*

In this section we describe a distributed algorithm, more precisely, a labelled graph recognizer with structural knowledge ι, for recognizing a given recursive $\sim^\iota$-closed class of labelled graphs $\mathcal{F}$. Throughout the section we suppose that $\mathcal{F}$ is recursive and closed under $\sim^\iota$ without further mentioning it.

We remark that, in order to recognize the given graph, we have two pieces of information:

- The graph $\rho(\mathbf{G})$ computed with Mazurkiewicz' algorithm M.
- The structural knowledge $\iota(\mathbf{G})$.

We define the following set. Let

$$\mathcal{F}_\iota = \{(\mathbf{H}, \iota(\mathbf{G})) \mid \mathbf{G} \in \mathcal{F} \text{ and } \mathbf{G} \text{ is a covering of } \mathbf{H}\}.$$

From the definition of $\mathcal{F}_\iota$ and the closure of $\mathcal{F}$ under $\sim^\iota$ we obtain the fundamental lemma that links membership in F and the structural pieces of information.

Lemma 45. *Let* $\mathbf{G}$ *be a graph. Then for any run* ρ *of Mazurkiewicz' algorithm* $\mathcal{M}$,

$$\mathbf{G} \in \mathcal{F} \quad \text{if and only if} \quad (\rho(\mathbf{G}), \iota(\mathbf{G})) \in \mathcal{F}_\iota.$$

Another interesting consequence of the definition of $\mathcal{F}_\iota$ and the closure under $\sim^\iota$ is that we can decide the right-hand side without knowing graph $\mathbf{G}$.

Lemma 46. *For any* $(\mathbf{H}, \alpha)$,

(1) *if there exists* $\mathbf{K} \in \mathcal{F}$ *such that* $\mathbf{K}$ *is a covering of* $\mathbf{H}$ *and* $\iota(\mathbf{K}) = \alpha$, *then* $(\mathbf{H}, \alpha) \in \mathcal{F}_\iota$;
(2) *if there exists* $\mathbf{K} \notin \mathcal{F}$ *such that* $\mathbf{K}$ *is a covering of* $\mathbf{H}$ *and* $\iota(\mathbf{K}) = \alpha$, *then* $(\mathbf{H}, \alpha) \notin \mathcal{F}_\iota$.

We describe a semi-algorithm $\mathcal{T}_\mathcal{F}$ that runs *internally* on all vertices. Its execution is reset whenever the vertex is involved in the application of an M-rule. We fix a recursive enumeration of the set of all labelled graphs.

Semi-Algorithm 1: $\mathcal{T}_\mathcal{F}$

Data: graph $\mathbf{H} \in \mathcal{G}_L$,
label $\alpha \in L$
Result: label r /* TRUE *or* FALSE */
repeat
/* *Enumerate all the labelled graphs*: */
take the next $\mathbf{K} \in \mathcal{G}_L$
until $\iota(\mathbf{K}) = \alpha$ *and* $\mathbf{K}$ *is a covering of* $\mathbf{H}$
Output: the result of "$\mathbf{K} \in \mathcal{F}$?" /* *F is a recursive set* */

In fact, $\mathcal{T}_\mathcal{F}$ running internally means that it can be turned into a graph relabelling system in which each rule just relabels the label of one vertex, independently of the labels of its neighbours.

We make a fair merge of this semi-algorithm and Mazurkiewicz' rules: we suppose that the execution of the (internal) semi-algorithm $\mathcal{T}_\mathcal{F}$ is interleaved with the execution of the (external) Mazurkiewicz rules such that the **repeat until** does not loop infinitely. These assumptions can be omitted provided we make some rather technical improvements that are detailed later in Section 8.3.

We recall that a run of Mazurkiewicz' algorithm on a labelled graph $\mathbf{G}$ uses, for every vertex v, the tuple $(\lambda(v), c(v))$ where $c(v) = (n(v), N(v), M(v))$. Furthermore, $\mathbf{H}_M = (H_M, \lambda_M)$ denotes the graph that we can reconstruct with information extracted from a mailbox M.

The "fair" labelled graph recognizer $(\mathcal{R}_\mathcal{F}, \mathcal{K}, \iota)$ with structural knowledge ι is the following:

- labels: $(\iota, \lambda, n, N, M, r)$;
- rules of $\mathcal{R}_\mathcal{F}$:
  - an initial rule that converts the $(\iota(\mathbf{G}), \lambda)$ label to $(\iota(\mathbf{G}), \lambda, 0, \emptyset, \emptyset, ?)$ on all vertices;

- the (internal) $\mathcal{T}_{\mathcal{F}}$ rules run on each vertex as a *fair* concurrent process on input $\langle \mathbf{H}_M, \iota(\mathbf{G})\rangle$ and output r;
- the M rules run on $(\lambda, (n, N, M))$, resetting $\mathcal{T}_{\mathcal{F}}$ and setting r to ? each time they are applied;

– final condition $\mathcal{K}$: label r is TRUE on all vertices.

Finally we obtain:

Proposition 47. *On input* $(G, \Lambda_{\iota(\mathbf{G})})$ *the recognizer* $(\mathcal{R}_{\mathcal{F}}, \mathcal{K}, \iota)$ *terminates and* $r(v) =$ TRUE *for every vertex* v *if and only if* $\mathbf{G} \in \mathcal{F}$.

Proof. As $\mathcal{T}_{\mathcal{F}}$ runs in a fair way, M eventually terminates with $\mathbf{H}_M = \rho(\mathbf{G})$ for some run ρ. Then $\mathcal{T}_{\mathcal{F}}$ will be processed on input $\langle \rho(\mathbf{G}), \iota(\mathbf{G})\rangle$; the **repeat until** loop will terminate because at least $\mathbf{G}$ verifies the two conditions. Thus, by Lemmas 45 and 46, $r(v) =$ TRUE if and only if $\mathbf{G} \in \mathcal{F}$, for all v. □

8.2. *A Necessary and Sufficient Condition for Recognizability*

Due to the fair merge hypothesis, the recognizer of Proposition 47 is not a pure labelled graph recognizer, but with a little advance upon the course of this paper, we generalize Theorem 22 to recognizability with structural knowledge by summarizing Corollary 38 and Theorem 52:

Theorem 48. *Let* $\mathcal{F}$ *be a class of labelled graphs and let* ι *be a structural knowledge. The following statements are equivalent*:

(1) $\mathcal{F}$ *is recognizable with structural knowledge* ι.
(2) $\mathcal{F}$ *is recursive and closed under* $\sim^{\iota}$.

An interesting corollary is the closure under boolean operations.

Corollary 49. *The classes of graphs that are recognizable with structural knowledge* ι *are closed under union, intersection, and complement.*

8.3. *Improved Recognition Rules*

We present some modifications of the previous recognizer $\mathcal{R}_{\mathcal{F}}$.

Saving on Running $\mathcal{T}_{\mathcal{F}}$. First we will not run $\mathcal{T}_{\mathcal{F}}$ when termination of M is obviously not realized. We make the relative termination of M explicit. We add one more boolean variable, denoted $b(v)$, whose initial value is FALSE. When a vertex v_0 is relabelled by Mazurkiewicz' algorithm, $b(v_0)$ becomes FALSE. If no rule of Mazurkiewicz' algorithm can be applied to the vertex v_0, then $b(v_0)$ becomes TRUE: all vertices of the ball centred at v_0 can reconstruct the same graph $\mathbf{H}_M = (H_M, \lambda_M)$. If $b(v)$ is TRUE, then the vertex runs the semi-algorithm $\mathcal{T}_{\mathcal{F}}$, setting the result to $r(v)$.

Proposition 50. *If $b(v) =$ TRUE for each vertex of* $\mathbf{G}$, *then Mazurkiewicz' algorithm has reached a terminal configuration. Conversely, when Mazurkiewicz' algorithm has reached a terminal configuration, then, after a number of steps bounded by the size of* $\mathbf{G}$, *the label $b(v)$ is* TRUE *and remains* TRUE *for all vertices v of* $\mathbf{G}$.

Eliminating the Fairness Assumption on $\mathcal{T}_{\mathcal{F}}$. The basic idea of the proof is serialization: first run $\mathcal{M}$ on $(G, \Lambda_{\iota(\mathbf{G})})$, then $\mathcal{T}_{\mathcal{F}}$ on $\langle \rho(\mathbf{G}), \iota(\mathbf{G})\rangle$. The problem is that we cannot locally determine the termination of $\mathcal{M}$ [MMW]. Furthermore, the termination of $\mathcal{T}_{\mathcal{F}}$ is certain only on the final input $\langle \rho(\mathbf{G}), \iota(\mathbf{G})\rangle$, as $\mathcal{T}_{\mathcal{F}}$ will not terminate on an arbitrary input (e.g., if ι encodes the size and $|V(\mathbf{H})|$ does not divide α, then there is no solution satisfying the two constraints). Hence, it is not possible to run $\mathcal{T}_{\mathcal{F}}$ entirely each time an M-rule is applied. We cannot improve $\mathcal{T}_{\mathcal{F}}$ because, for a general knowledge ι, we cannot decide if there is a graph satisfying the two constraints on a given input $\langle \mathbf{H}, \alpha\rangle$.

The first solution to these problems is to make some reasonable fairness assumptions on the runs of $\mathcal{T}_{\mathcal{F}}$. The second solution is to introduce a "pseudo-synchronization" of $\mathcal{M}$ and $\mathcal{T}_{\mathcal{F}}$ executed step by step.

More formally, we add some rules and some labels to the enumeration algorithm $\mathcal{M}$ in order to obtain a pseudo-synchronization with $\mathcal{T}_{\mathcal{F}}$. We add a (counter) label $co \in \mathbb{N}$ to every vertex and the following pseudo-synchronization rule, which ensures that the enumeration of vertices progresses in a uniform way throughout the graph:

$\mathcal{M}'$-0: **Pseudo-synchronization rule**

Precondition:

- there exists a vertex $v \in B(v_0)$ such that $co(v_0) < co(v)$ and $co(v_0) \leq co(v')$ for all $v' \in B(v_0)$.

Relabelling:

- $co'(v_0) := co(v_0) + 1$.

We modify the rules of $\mathcal{M}$ as follows. First we add the precondition $co(v_0) \leq c(v)$, for all $v \in B(v_0)$, to rules $\mathcal{M}$-1 and $\mathcal{M}$-2 of $\mathcal{M}$. Then we add to the relabelling performed by the *diffusion rule* $\mathcal{M}$-1:

$$co'(v) := co(v) + 1 \qquad \text{for all} \quad v \in B(v_0).$$

Let $\mathcal{M}'$ denote the above modification of the enumeration algorithm $\mathcal{M}$. We have:

Proposition 51. *For every graph* $\mathbf{G}$, $\mathcal{M}'$ *has the same behaviour as* $\mathcal{M}$ *on* $\mathbf{G}$.

Next, we add a rule depending on the set $\mathcal{F}$ that we want to recognize. As $\mathcal{T}_{\mathcal{F}}$ is a semi-algorithm we have to perform it step by step. We fix an enumeration of the class of labelled graphs $\mathcal{G}_L$. We add two more labels $p \in \mathbb{N}$ and $r \in \{$TRUE, FALSE, ?$\}$. The label p will contain the actual step of $\mathcal{T}_{\mathcal{F}}$ and the flag r its answer, as previously. Every time the diffusion rule is applied the flag will be reset to ?. Applying one step of $\mathcal{T}_{\mathcal{F}}$ can yield TRUE, FALSE, or ?. Let pure$\mathcal{T}_{\mathcal{F}}$ now be the following algorithm.

Algorithm 2: pure$\mathcal{T}_{\mathcal{F}}$

Data: graph **H**,
label α,
integer p
Result: label r /* ? *or* TRUE *or* FALSE */
$\mathbf{K} = \mathbf{G}_p$ /* *the* pth *graph in the enumeration of* $\mathcal{G}_L$ */
if $\iota(\mathbf{K}) \neq \alpha$ or **K** is not a covering of **H**
  Output: ?
else
  Output: the result of "$\mathbf{K} \in \mathcal{F}$" /* TRUE *or* FALSE */

We add $p'(v) := 1$ and $r'(v) := ?$ in the relabelling part of the *diffusion rule*, for all $v \in B(v_0)$. The last rule needed for synchronizing $\mathcal{M}$ and pure $\mathcal{T}_{\mathcal{F}}$ is then the following one:

$\mathcal{M}'$-3: **Testing rule**
Precondition:
- $co(v_0) \leq co(v)$ for all $v \in B(v_0)$.
- $r(v_0) = ?$.

Relabelling:
- $r'(v_0) :=$ pure$\mathcal{T}_{\mathcal{F}}(\mathbf{H}_{M(v_0)}, \iota(\mathbf{G}), p(v_0))$.
- $p'(v_0) := p(v_0) + 1$.
- $co'(v_0) := co(v_0) + 1$.

By $\mathcal{R}_{\mathcal{F}}$ we denote the graph relabelling system consisting of rules $\mathcal{M}'$-0, $\mathcal{M}'$-3, and the modified rules $\mathcal{M}'$-1, $\mathcal{M}'$-2, with priorities $\mathcal{M}'$-0, $\mathcal{M}'$-3 $<$ $\mathcal{M}'$-2, M'–2. Using different priorities means that if a vertex v_0 satisfies the conditions of two rules, then the rule with the higher priority will be applied. It is a way of simplifying the expression of the preconditions.

At the end of a run, the final condition $\mathcal{K}$ is that all flags r are TRUE.

Theorem 52. *For any structural knowledge* ι, $(\mathcal{R}_{\mathcal{F}}, \mathcal{K}, \iota)$ *is a labelled graph recognizer of* $\mathcal{F}$.

Proof. The proof is based on the fact that our system is close enough to a serial composition of $\mathcal{M}$ and pure $\mathcal{T}_{\mathcal{F}}$. To emphasize this, we use the following notation. Suppose that w is a (possibly infinite) $\mathcal{R}_{\mathcal{F}}$ relabelling sequence, $w = \alpha_0, \beta_1^1, \ldots, \beta_{d_1}^1, \alpha_1, \beta_1^2, \ldots, \beta_{d_2}^2, \ldots, \alpha_p, \ldots$, where the α_i's and β_j^i's are labelled graphs, α_0 being the initial labelling, α_i are labellings obtained using rules $\mathcal{M}'$-1 and $\mathcal{M}'$-2, and β_j^i are labellings obtained by applying rule $\mathcal{M}'$-0 or $\mathcal{M}'$-3. For a labelled graph γ, let $\bar{\gamma}$ denote the projection on the first four components of the labels. We extend the notation $\bar{w}$ to every relabelling sequence w. Then we have:

Lemma 53. *If w is an $\mathcal{R}_{\mathcal{F}}$ relabelling sequence, then $\bar{w}$ is a valid $\mathcal{M}$ relabelling sequence.*

We denote by $i_{\max}$ the greatest index for α.

Proposition 54. *There exists some constant κ depending only on $\mathbf{G}$, such that $d_i \leq \kappa$, for all $1 \leq i \leq i_{\max}$.*

Proof. For a given i, let v_i be the vertex where the rule leading from $\beta^i_{d_i}$ to α_i is applied. That is, the vertex v_i satisfies all conditions for rules $\mathcal{M}'$-1 or $\mathcal{M}'$-2 except perhaps the part related to co. That means that if the conditions of rules $\mathcal{M}'$-0 or $\mathcal{M}'$-3 are fulfilled for v_i, then so they are for rules $\mathcal{M}'$-1 or $\mathcal{M}'$-2. In other words, since rules $\mathcal{M}'$-1 or $\mathcal{M}'$-2 have higher priority than rules $\mathcal{M}'$-0 or $\mathcal{M}'$-3, the label of v_i does not change in $\beta^i_1, \ldots, \beta^i_{d_i}$.

It is easy to verify that for every graph $\mathbf{G}$ in w and all adjacent vertices $\{u, v\} \in E(\mathbf{G})$, we have $|co(u) - co(v)| \leq 1$. Hence, the co-label of a vertex v in $\mathbf{G}$ belongs to the interval $[co(v_i) - \Delta(\mathbf{G}), co(v_i) + \Delta(\mathbf{G})]$, where $co(v_i)$ is the co-label of v_i in α_i, and $\Delta(\mathbf{G})$ is the diameter of $\mathbf{G}$. Since co-values increase at each application of rules $\mathcal{M}'$-0 or $\mathcal{M}'$-3, we finally obtain $d_i \leq \kappa = |V(\mathbf{G}) - 1|^{2\Delta(\mathbf{G})+1}$. □

As an immediate corollary, we obtain that w is finite. Let p denote the maximal size of an execution of $\mathcal{M}$ on $(G, \Lambda_{\iota(\mathbf{G})})$. If $|w| \geq p(\kappa + 1)$, then w is of the form $w_0\beta$ where $\bar{w} = \bar{w_0}$. As we cannot apply the *pseudo-synchronization rule* any more, $co(v) = co(u)$ holds for all vertices u and v (consider the vertex with the minimal co). This means that the underlying $\mathcal{M}$ relabelling is finished, and that β corresponds to the computation of pure$\mathcal{T}_{\mathcal{F}}$ on $\rho(\mathbf{G})$ for a certain execution ρ of $\mathcal{M}$. This computation is finite because at least $(G, \Lambda_{\iota(\mathbf{G})})$ verifies the two constraints and decides whether $(G, \Lambda_{\iota(\mathbf{G})})$ belongs to $\mathcal{F}$.

Therefore, $(\mathcal{R}_{\mathcal{F}}, \mathcal{K}, \iota)$ is a recognizer of $\mathcal{F}$. □

Remark 55. In [G] a different version for the "pure" recognizer is proposed, based upon the notion of quasi-covering and the "quasi-cartography" algorithm of [GM3].

8.4. *Extension to k-Local Computations*

The notions of coverings and graph relabelling systems can be easily extended to labelled graphs or systems where the relabelling occurs in a ball of fixed radius $k \in \mathbb{N}$. The lifting lemma is straightforwardly extended to this case and the necessary condition (Corollary 38) holds. The difficulty is to extend the Mazurkiewicz algorithm. In the following we present the solution for (unlabelled) graphs for the sake of simplicity. The proofs can be extended to labelled graphs as previously.

k-Coverings. Let v be a vertex of a graph $\mathbf{G}$, and let k be a positive integer. We denote by $B_{\mathbf{G}}(v, k)$ the (open) ball of radius k with centre v. Moreover, for any $k < k'$ a k'-covering is also a k-covering. Note that a 1-covering is exactly a covering in the classical

sense [M1]. Hence, all k-coverings are coverings. Furthermore, the following lemma links 1-coverings and k-coverings.

Lemma 56 [B]. *Let $k > 0$ and consider two graphs $\mathbf{G}, \mathbf{G}'$ such that $\mathbf{G}$ covers $\mathbf{G}'$ via μ. Then μ is a k-covering (resp. $\mathbf{G}$ is a k-covering of $\mathbf{G}'$) if and only if, for every cycle C' in $\mathbf{G}'$ of length at most $2k+1$, the inverse image $\mu^{-1}(C')$ is a disjoint union of cycles isomorphic to C'.*

As a consequence of Theorem 3 we deduce that all k-coverings can be obtained from a given spanning tree: we compute coverings and for each covering we test whether it is a k-covering.

k-Local Relabelling Rules. The crucial property of the algorithm is based on a total order on local views such that the local view of any vertex cannot decrease during the computation. Let $\mathcal{N}$ be the set of k-balls labelled by positive integers (up to isomorphism), this is the set of local views. Let $\blacktriangleleft$ be any total order on the set of k-balls.

We denote by $W((B,\lambda)) = \sum_{v\in V(B)} \lambda(v)$ the *weight* of the labelled ball (B,λ). For any ball B and any integer w we choose an arbitrary total order $\blacktriangleleft_w^B$ on the finite set of labelled balls with underlying graph B and weight w.

We define a total order $\prec$ on $\mathcal{N}$ by letting $(B_1,\lambda_1) \prec (B_2,\lambda_2)$ if one of the following conditions is satisfied:

1. $B_1 \blacktriangleleft B_2$.
2. $B_1 = B_2$ and $W(B_1,\lambda_1) < W(B_2,\lambda_2)$.
3. $B_1 = B_2$, $W(B_1,\lambda_1) = W(B_2,\lambda_2)$, and $(B_1,\lambda_1) \blacktriangleleft_w^B (B_2,\lambda_2)$, where $B = B_1 = B_2$ and $w = W(B_1,\lambda_1) = W(B_2,\lambda_2)$.

Remark 57. Note that we make no particular assumption on the orderings $\blacktriangleleft$ and $\blacktriangleleft_w^B$, except that they have to be computable.

We need a new rule, that we call the *updating rule*, and an additional flag f now. The reason is that a neighbourhood of radius 1, as in the previous case, is just a set, hence the local views of neighbours can be updated within the *renaming rule*. However, that is no longer the case with neighbourhoods of radius k, as can be seen in the simple example of Figure 12.

A vertex v is then labelled by a tuple of the form (n, N, M, f) representing the following information during the computation:

– n, N, M represents the same information as previously.

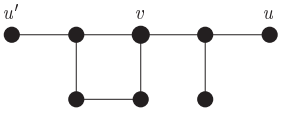


Fig. 12. According to its 2-view, u cannot conclude about its *real* situation in the 2-view of v. For example, u cannot update the 2-view of v after its first renaming.

– $f(v) \in \{\text{UPTODATE}, \text{DATED}\}$ is a flag that will be set to DATED when the vertex has to update its neighbourhood information.

The new rule that will be triggered by the flag f is

$\mathcal{M}_k$-2bis: **Updating rule**
Precondition:
• $f(v_0) = \text{DATED}$.
Relabelling:
• v_0 updates its local view by letting $N'(v_0) := (B(v_0, k), \lambda)$, where λ is given by $\lambda(v) = n(v)$.
• The new value $f'(v_0)$ of the flag is UPTODATE.

The diffusion and renaming rules are adapted in order to manage the flag f: add "$f(v_0) = \text{UPTODATE}$" to the preconditions of both rules and add "$f(v_0) := \text{DATED}$" to the relabelling of the renaming rule.

Taking into account the new rule and the fact that, before any of its neighbour can apply a rule, to any relabelling of a vertex will correspond an updating of the local view,[2] it is possible to adapt the previous proofs and obtain a characterization of k-recognizability of the same kind: k-recognizable recursive classes of graphs are $\sim_k^\iota$-closed recursive classes of graphs, where $\sim_k^\iota$ is the counterpart of $\sim^\iota$ by using k-coverings instead of coverings.

An interesting corollary for this extension is that it is possible to compare the expressiveness of rules of different radii. It is a way to show that there exists a strict radius hierarchy since, for any $k \neq k'$, some equivalence classes are not equal. For example, for $k \in \mathbb{N}$, the equivalence class for $\sim_k^\varepsilon$ of the ring of size $2k$ contains all the rings of size greater than $2k$, but its equivalence class for $\sim_{k+1}^\varepsilon$ is a singleton.

9. Applications: Particular Cases of Structural Knowledge

The results of the previous section state that the $\sim^\iota$-equivalence classes are the "atoms" of recognizable classes of graphs. In this section we investigate the properties of equivalence classes in some particularly interesting cases. We will see the influence of no knowledge, of an upper bound on the diameter, of an upper bound on the size, and of a tight bound on the size, of the diameter, of the size, and of the topology. These kinds of knowledge present a strict hierarchy that we describe precisely.

9.1. *No Structural Knowledge*

(*Unlabelled*) *Graphs*. In [L], Leighton gives a decidable criterion for two graphs admitting a common covering. He uses the so-called *degree refinement* of a graph G, i.e., the partition of the vertices of G into the minimal number of blocks $B_0, B_1, \ldots, B_{t-1}$ for which there are constants $r_{i,j}, 0 \leq i, j < t$, such that every vertex v in B_i is incident

[2] That is why the Updating Rule is numbered 2bis.

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 the two matrices are conjugated.

Theorem 58 [L]. *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.*

Concerning our relation $\sim^{\varepsilon} = \overset{*}{\leftrightarrow}$, we obtain:

Proposition 59. *Let G and G' be two connected graphs. Then $G \sim^{\varepsilon} G'$ if and only if G and G' have the same degree refinement.*

Remark 60. We do not have such a characterization for k-coverings with $k > 1$, and general k-coverings seem to be a more difficult subject, see [DZ].

In particular, graph classes that are recognizable with no structural knowledge can be seen as recursive sets of degree refinement.

Noting that the $\sim^{\varepsilon}$-class of a tree is a singleton, we get

Corollary 61. *Any recursive class of trees is ε-recognizable.*

Corollary 62. *Let $d \in \mathbb{N}$. The class of d-regular graphs is ε-recognizable.*

As a consequence, the class of rings is recognizable.

Labelled Graphs. An interesting problem of distributed computations, the majority problem (see [LMS1]), can be expressed in terms of the recognition of labelled graphs. The problem can be stated as follows. On a given graph whose vertices are labelled by label $\mathtt{a}$or $\mathtt{b}$, decide whether there are as many vertices labelled by $\mathtt{a}$as by $\mathtt{b}$, or more. A positive answer is given in [LMS1] where a recognition system with 15 rules is given. We can now see this result as a corollary of Theorem 22 since an easy computation, relying upon the number of sheets (Lemma 2), shows that the ration $|\mathbf{G}|_{\mathtt{a}}/|\mathbf{G}|_{\mathtt{b}}$ is invariant by $\sim^{\varepsilon}$, where $|\mathbf{G}|_{\alpha}$ is the number of times the label α appears in $\mathbf{G}$. An open question of [LMS1] was to know if it was possible to decide if $|\mathbf{G}|_{\mathtt{a}} \leq m|\mathbf{G}|_{\mathtt{b}}$ for a given m. The previous remark also gives a positive answer in this case.

These questions, included in the study of statistical problems (problems, like the Majority Problem, whose answers depend upon the multiplicity of the labels that are present on the network), are reviewed in detail in Chapter 8 of [G].

9.2. *Size Upper Bounds, Tight Upper Bounds, Size, and Diameter*

We define an upper bound recognizable class to be a class that is recognizable by a labelled graph recognizer whatever the chosen upper bound function, see Section 6.1. It is then equivalent to say that an upper bound recognizable class is a class that is closed under all $\sim^{b}$ relations, where b is an upper bound function. So, if we define

$\sim^{\text{Bound}} = \bigcup_b \sim^b$, where the union is taken over all upper bounds b on the size of G, we have

Proposition 63. *Let $\mathcal{F}$ be a class of graphs. Then $\mathcal{F}$ is upper bound recognizable if and only if $\mathcal{F}$ is closed under $\sim^{\text{Bound}}$.*

Lemma 64. $\sim^{\varepsilon} = \sim^{\text{Bound}}$.

Proof. $\sim^{\varepsilon} \supset \sim^{\text{Bound}}$ is obvious. For the other inclusion, suppose we have a graph H that is a covering of G and G'. Then for an upper bound function b such that $b(G) = b(G') = \max(|G|, |G'|)$, we have $G \sim^b G'$. □

As an immediate corollary of the previous section, we obtain:

Corollary 65. *Let $\mathcal{F}$ be a class of graphs that is upper bound recognizable. Then $\mathcal{F}$ is also recognizable with no structural knowledge.*

Remark 66. Knowing a bound on the number of vertices is the same as having no information in the recognition context. This quite surprising result is similar to one obtained in Proposition 18 of [YK1]. This is quite different in the termination detection context (see [MT] or [GM3]).

We recall from Section 6.1 that a tight upper bound is an upper bound b such that for all graphs G, $|V(G)| \leq b(G) < 2|V(G)|$. We define a tight upper bound recognizable class to be a class that is recognizable by a labelled graph recognizer whatever the chosen tight upper bound function. It is then equivalent to say that a tight upper bound recognizable class is a class that is closed under the $\sim^b$ relation, for all tight upper bound functions b. Then, since a strict covering has at least two sheets, the equivalence class of any covering-minimal graph G is a singleton, thus:

Proposition 67. *The class of covering-minimal graphs is recognizable knowing a tight upper bound.*

It is easy to see that if $\mathcal{F}$ is size-recognizable, then $\mathcal{F}$ is recognizable with a tight upper bound, hence $\mathcal{F}$ is recognizable knowing a bound on the diameter, hence $\mathcal{F}$ is ε-recognizable. Figure 13 proves that diameter and size knowledge are incomparable.

We can also note using Lemma 1 that if G and H are both coverings of a 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. That is to say that knowing the number of vertices is the same as knowing the number of edges.

For more applications when the size is known, please refer to Sections 6.3 and 7.

We summarize the results in Table 1 where YES and NO answer the question whether the listed class is recognizable or not with the corresponding structural knowledge.

This table shows that even for simple structural knowledge, there is a strict hierarchy of recognizable graph classes based on such knowledge.

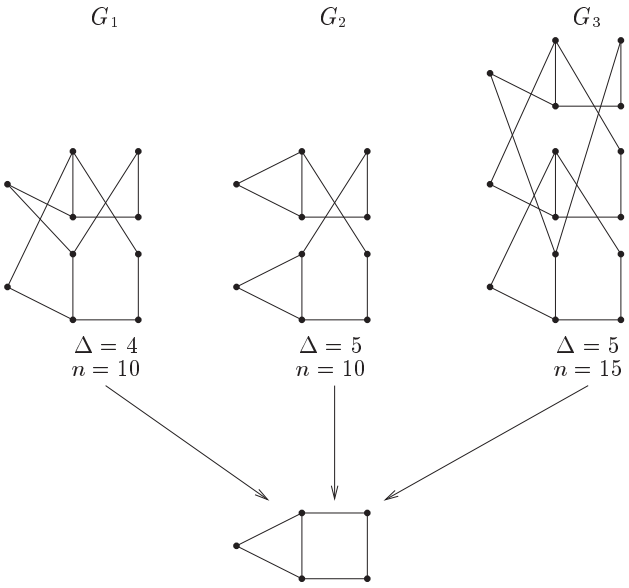


Fig. 13. Diameter and size knowledge are incomparable.

10. Conclusion

10.1. *Nondeterministically Recognizable Classes of Labelled Graphs*

In this subsection we present some ideas on the nondeterministic case.

Given a nondeterministic labelled graph recognizer $(\mathcal{R}, \mathcal{K}, \iota)$, a labelled graph $\mathbf{G}$ is recognized with structural knowledge ι if there exists *at least* one relabelling sequence which leads from $(G, \Lambda_{\iota(\mathbf{G})})$ to a labelled graph in $\mathcal{K}$.

Table 1. Summary of results.

	No information = size bound = diameter bound	Tight bound on the size	Number of vertices	Topology
Trees	YES[1]	–	–	–
Covering-minimal graphs	NO[2]	YES[3]	–	–
Graphs with even size	–	NO[4]	YES[5]	–
Planar graphs	–	–	NO[6]	YES

[1]By Corollary 61.
[2]The class of covering-minimal graphs is not closed under coverings. Let us consider rings. The covering-minimal rings are the ones that have a prime number of vertices.
[3]By Proposition 67.
[4]If p is an odd integer, then consider the equivalence on the rings $R_{2p}\sigma^{\text{TightBound}}R_{3p}$ via R_p and apply Corollary 38.
[5]Trivial.
[6]By Theorem 43.

Definition 68. A class $\mathcal{F}$ of labelled graphs is nondeterministically recognizable with structural knowledge ι if there exists a nondeterministic labelled graph recognizer with structural knowledge $(\mathcal{R}, \mathcal{K}, \iota)$ such that the set of recognized labelled graphs is $\mathcal{F}$.

The relation $\leftarrow$ denotes the inverse covering relation. We denote by $\leftarrow^{\iota}$ the following relation: $\mathbf{H} \leftarrow^{\iota} \mathbf{G}$ if $\mathbf{G}$ is a covering of $\mathbf{H}$, and $\iota(\mathbf{G}) = \iota(\mathbf{H})$. We remark that, as the composition of two coverings is a covering, the relation $\leftarrow^{\iota}$ is transitive. It is also reflexive, since the identity is a covering. It is worth noting that $\leftarrow^{\iota}$ is *not* symmetric.

We say that a graph class Fis *closed* under $\leftarrow^{\iota}$ if whenever $\mathbf{H}$is in F, then any graph $\mathbf{G}$ such that $\mathbf{H} \leftarrow^{\iota} \mathbf{G}$ is also in F.

We have not obtained a characterization, nevertheless using exactly the same tools as previously, namely coverings and Mazurkiewicz' enumeration algorithm, we can prove:

Proposition 69. *Let $\mathcal{F}$ be a class of labelled graphs. If $\mathcal{F}$ is recursive and closed under $\leftarrow^{\iota}$, then $\mathcal{F}$ is nondeterministically recognizable with structural knowledge ι.*

Proof. We assume that Fis a recursive set closed under $\leftarrow^{\iota}$. It turns out that the following nondeterministic labelled graph recognizer, very close to our first one, in Section 5.6, recognizes F: As done in that section, we add a label $r \in \{?, \text{TRUE}, \text{FALSE}\}$ to the labels used by the system M. This label will contain the result of the computation given by the following new testing rule:

TEST2:
Precondition:
• $r(v_0) = ?$
Relabelling:
• Compute $\mathbf{H} = (\mathbf{H}_{\mathbf{M}(\mathbf{v_0})}, \lambda_{\mathbf{M}(\mathbf{v_0})})$.
• Set $r(v_0)$ to the result of the test "$\mathbf{H} \in \mathcal{F}$ and $\iota(\mathbf{H}) = \iota(\mathbf{G})$?"

The system is the following:

- labels: (λ, n, N, M, r);
- rules:
  - an initial relabelling rule that converts, for every vertex v, the $\lambda(v)$ label to $(\lambda(v), 0, \emptyset, \emptyset, ?)$ on all vertices v;
  - the M rules run on $(\lambda, (n, N, M))$, setting $r(v)$ to ? each time a rule is applied to v;
  - the TEST2 rule;
- final condition: label r is TRUE on all vertices.

As previously, the set of rules expressed by the generic rule TEST2 can be used in a graph relabelling system because $\mathcal{F}$ is recursive.

This nondeterministic labelled graph recognizer is again noetherian and we prove it recognizes exactly F.

Suppose that $\mathbf{G}$ is in F, then, by Proposition 30, there exists a run ρ of M such that $\mathbf{H} = \mathbf{G}$. For this run, the rule TEST2 returns TRUE. Hence $\mathbf{G}$ is recognized.

Suppose now that $\mathbf{G}$ is recognized. Then by definition, this means that there is a run ρ such that $\rho(\mathbf{G}) \in \mathcal{F}$ and $\iota(\rho(\mathbf{G})) = \iota(\mathbf{G})$. Since $\mathbf{G}$ is a covering of $\rho(\mathbf{G})$ and Fis closed under $\leftarrow^{\iota}$, it follows that $\mathbf{G} \in \mathcal{F}$. □

Nondeterministic labelled graphs are strictly more powerful than deterministic ones, since, contrary to deterministic ones, it turns out it is always possible to distinguish between two distinct graphs. That means that, given $\mathbf{G}$ and $\mathbf{G}'$, there exists a nondeterministic labelled graph recognizer such that either $\mathbf{G}$ is recognized and $\mathbf{G}'$ is not recognized, or $\mathbf{G}'$ is recognized and $\mathbf{G}$ is not recognized. This follows obviously from the fact that if $\mathbf{G} \neq \mathbf{G}'$, then it is not possible to obtain ($\mathbf{G}' \leftarrow^{\varepsilon} \mathbf{G}$, and $\mathbf{G} \leftarrow^{\varepsilon} \mathbf{G}'$).

10.2. *General Comments*

One motivation for defining recognizable classes of graphs through distributed computations was to try to find a notion of recognizability for graphs, with properties similar to the ones enjoyed by simpler objects like words or trees. One difficulty in the quest for a good notion of graph recognizability is that, contrary to words or trees, there is no natural unique computation on a graph. Hence, we can try to solve this by a distributed approach, starting and computing everywhere at the same time *in a distributed way*. Since what is computed and recognized by the graph recognizers are the $\sim^{\iota}$ classes, we saw, in the case of recognition with no additional knowledge, that two graphs with the same degree refinement are undistinguishable by local computations. Assuming global knowledge ι does not help either, since recognizable classes are arbitrary (recursive) unions of $\sim^{\iota}$-equivalence classes. That is, if we want each equivalence class to be a singleton, then the recognizability notion becomes too powerful.

However, from the distributed point of view, this study brings a lot of insight about the relation between what is computable and what is not computable in a network according to the structural knowledge that is available to a distributed algorithm. One main difficulty in the conception of distributed algorithm is that there is a priori no centralized behaviour, hence distributed algorithms are often designed with the assumption that the network meets some special specifications (a given topology, unique numbers available for each node, sense of direction) or that some structural properties of the network are known (the diameter or the size). These specifications can be difficult to maintain, especially in a distributed way, and, from a reliability point of view, it is important to reduce them as much as possible. This study can then be seen as a step towards understanding what network specifications are *really necessary* in order to achieve a given distributed task.

10.3. *Related Works and Open Problems*

This paper investigates the recognition problem for labelled graphs by local computations. Angluin [A] and Yamashita and Kameda [YK1], [YK3] addressed the same problem for different models. Note that in these models a recognition problem is solved when vertices recognize this fact. The characterization of classes of labelled graphs recognized in this way by local computations remains open. In particular, we have studied in which cases it is possible to detect global termination locally [GMT].

We consider labelled graphs, nevertheless we assume that vertex and edge labels are not necessarily prepared by a distributed algorithm.

We consider our model asynchronous: not in the sense that all vertices may rewrite their labels in parallel but in the sense that the computations on disjoint balls may be done in parallel.

We do not need some kind of fairness because we only consider noetherian computations, thus steps are executed until the whole computation is terminated.

Some similarities exist between this paper and [YK1] and [YK3], in particular the following are some parallel concepts/results:

- mailbox of a vertex / view of a vertex,
- some computations of Mazurkiewicz' algorithm / Lemma 4 [YK1],
- quotient graph / quotient graph,
- covering-minimal graph / minimal realization,
- some graphs having the same quotient are indistinguishable / Lemma 14, Theorem 23 of [YK1],
- Corollary 62 / Theorem 18 of [YK1].

Acknowledgments

The authors are grateful to the anonymous referees for the valuable comments. They thank Joost Engelfriet for corrections, very fruitful comments, and suggestions.

References

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

[B] A. Bottreau. Réécritures de graphe et calculs distribués. Ph.D. thesis, Université Bordeaux I, 1997.

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

[BV] P. Boldi and S. Vigna. Computing anonymously with arbitrary knowledge. In *Proceedings of the* 18*th ACM Symposium on Principles of Distributed Computing*, pages 181–188. ACM Press, New York, 1999.

[CM] B. Courcelle and Y. Métivier. Coverings and minors: applications to local computations in graphs. *European Journal of Combinatorics*, 15:127–138, 1994.

[DFPP] H. De Fraysseix, J. Pach, and R. Pollack. Small sets supporting Fáry embeddings of planar graphs. In *Proceedings of the* 20*th Annual ACM Symposium on Theory of Computing*, *Chicago*, 1988, pages 426–433. ACM Press, New York, 1988.

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

[F] M. R. Fellows. The Robertson–Seymour theorems: a survey of applications. *Contemporary Mathematics*, 89:1–18, 1989.

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

[G] E. Godard. Réécritures de Graphes et Algorithmique Distribuée. Ph.D. thesis, Université Bordeaux I, 2002.

[GM1] E. Godard and Y. Métivier. Characterization of classes of graphs recognizable by local computations with initial knowledge (extended abstract). In *SIROCCO* 01—8*th International Colloquium on Structural Information & Communication Complexity*, number 11 in Proceedings in Informatics, pages 179–194. Carleton Scientific, Ottawa, 2001.

[GM2] E. Godard and Y. Métivier. A characterization of families of graphs in which election is possible (extended abstract). In M. Nielsen and U. Engberg, editors, *Proceedings of Foundations of Software Science and Computation Structures*, *FOSSACS* '02, number 2303 in LNCS, pages 159–171. Springer-Verlag, Berlin, 2002.

[GM3] E. Godard and Y. Métivier. Equivalence of structural knowledges in distributed algorithms. In *SIROCCO* 02—9*th International Colloquium on Structural Information & Communication Complexity*, number 13 in Proceedings in Informatics, pages 149–164. Carleton Scientific, Ottawa, 2002.

[GMM] E. Godard, Y. Métivier, and A. Muscholl. The power of local computations in graphs with initial knowledge. In *Proceedings of Theory and Applications of Graph Transformations* '98, number 1764 in LNCS, pages 71–84. Springer, Berlin, 2000.

[GMT] E. Godard, Y. Métivier, and G. Tel. Election, termination and graph cartography. In preparation.

[L] F. T. Leighton. Finite common coverings of graphs. *Journal of Combinatorial Theory*, *Series B*, 33:231–238, 1982.

[LMS1] I. Litovsky, Y. Métivier, and E. Sopena. Checking global graph properties by means of local computations: the majority problem. *Electronic Notes in Theoretical Computer Science*, 2, 1995.

[LMS2] I. Litovsky, Y. Métivier, and E. Sopena. Graph relabelling systems and distributed algorithms. In H. Ehrig, H.J. Kreowski, U. Montanari, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 3, pages 1–56. World Scientific, Singapore, 1999.

[LMZ] 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.

[M1] W. S. Massey. *A Basic Course in Algebraic Topology*. Springer-Verlag, New York, 1991.

[M2] A. Mazurkiewicz. Trace theory. In W. Brauer et al., editor, *Petri Nets*, *Applications and Relationship to Other Models of Concurrency*, volume 255 of LNCS, pages 279–324. Springer-Verlag, Berlin, 1987.

[M3] A. Mazurkiewicz. Distributed enumeration. *Information Processing Letters*, 61:233–239, 1997.

[MMW] Y. Métivier, A. Muscholl, and P.-A. Wacrenier. About the local detection of termination of local computations in graphs. In D. Krizanc and P. Widmayer, editors, *SIROCCO* 97—4*th International Colloquium on Structural Information & Communication Complexity*, Proceedings in Informatics, pages 188–200. Carleton Scientific, Ottawa, 1997.

[MT] Y. Métivier and G. Tel. Termination detection and universal graph reconstruction. In *SIROCCO* 00—7*th International Colloquium on Structural Information & Communication Complexity*, pages 237–251, 2000.

[NS] M. Naor and L. Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.

[R1] K. Reidemeister. *Einführung in die Kombinatorische Topologie*. Vieweg, Brunswick, 1932.

[R2] K. H. Rosen, editor. *Handbook of Discrete and Combinatorial Mathematics*. CRC Press, Boca Raton, FL, 2000.

[RFH] P. Rosenstiehl, J.-R. Fiksel, and A. Holliger. Intelligent graphs. In R. Read, editor, *Graph Theory and Computing*, pages 219–265. Academic Press, New York, 1972.

[RS] N. Robertson and P. Seymour. Graph minors, VII: disjoint paths on a surface. *Journal of Combinatorial Theory*, *Series B*, 45:212–254, 1988.

[T] G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, Cambridge, 2000.

[vL] Jan van Leeuwen, editor. *Handbook of Theoretical Computer Science*, volumes A and B. Elsevier Science, Amsterdam, 1990.

[YK1] M. Yamashita and T. Kameda. Computing on anonymous networks: Part I—characterizing the solvable cases. *IEEE Transactions on Parallel and Distributed Systems*, 7(1):69–89, 1996.

[YK2] M. Yamashita and T. Kameda. Computing on anonymous networks: Part II—decision and membership problems. *IEEE Transactions on Parallel and Distributed Systems*, 7(1):90–96, 1996.

[YK3] M. Yamashita and T. Kameda. Leader election problem on networks in which processor identity numbers are not distinct. *IEEE Transactions on Parallel and Distributed Systems*, 10(9):878–887, 1999.

Received December 18, 2001, *and in revised form March* 5, 2003, *and August* 4, 2003, *and in final form August* 29, 2003. *Online publication January* 23, 2004.