Equivalence of Structural Knowledges in Distributed Algorithms

E. GODARD & Y. MÉTIVIER
LaBRI UMR 5800, Université Bordeaux I - ENSEIRB, France

Abstract

Distributed algorithms correctness usually relies upon the use of some knowledge about the underlying network (a specific topology, some metrics,...). We define equivalent structural knowledges to be such knowledges that can be computed distributively one knowing the other. We present a combinatorial characterization of this equivalence. Some applications are also given: zero knowledge, classical metrics (size, diameter,...).

This characterization is defined in terms of graphs coverings and quasi-coverings. The proofs are based upon an algorithm proposed by A. Mazurkiewicz, and on techniques of termination detection by Shy, Szymanski, and Prywes.

Keywords: Distributed Algorithms, Structural Knowledge, Reconstruction Algorithm, Coverings, Quasi-Coverings

Distributed algorithms usually assume some knowledge about the structure of the underlying graph (a specific topology: tree, chordal, ring,...; some metrics: the size, the diameter, some bounds on these values). We define structural knowledge to be such an information. Distributed algorithms can, and usually does, use a specific structural knowledge [12, 9]. A question naturally arises: given a structural knowledge, is it possible to compute another structural knowledge. It can be useful in order to, knowing a first structural knowledge, start a distributed algorithm $A$ that relies upon the explicit knowledge of a second structural property. We define equivalent structural knowledges to be knowledges that can be computed distributively one knowing the other. We present a combinatorial characterization of this equivalence.

Problems close to this one were previously investigated. A recognition algorithm is a distributed algorithm such that, knowing an initial information about the underlying network, the computation decides whether this network has or has not a given topological property. The characterization for recognizability by local computations, given a structural knowledge, was presented in [4]. The recognition was done with implicit termination. The influence of symmetry, expressed in term of the graph-theoretic concept of coverings (special kind of graphs morphism, to be precised later), is known, since the work of Angluin [1], to determine
the borderline between what is or what is not solvable by local computations. The authors give in [4] a necessary and sufficient condition for recognizable classes of graphs in terms of closure under covering-based relations.

Investigation about termination detection by local means has been done since the beginning of the study of the computational power of distributed systems, see e.g. [12] for a review. In particular, it was investigated in [7], where the tool of quasi-coverings (mappings that are partially coverings) was first introduced. A necessary condition was given. Recently, the converse was proved in [8]. The criteria for local detection of global termination given in this previous paper required that a node can detect, with local informations, the global termination of the computation. Hence this detection can be done on restricted subclasses of graphs, classes of graphs with quasi-coverings of bounded radius. For our purpose we need a weaker form of termination detection, we only require that every node detects that its own computed values will remain unchanged. When that detection is done, e.g. the node can start algorithm $A$, using the computed structural knowledge. To emphasize the difference, one shall note that with the stronger termination requirement, a corollary of the characterization of [8] is that we cannot have a universal distributed algorithm with explicit termination that, for every network, computes the degree at every node. For our weaker form, the result is obvious: every node outputs the desired value by counting its neighbours. The point is that, knowing nothing about the underlying network, the instant when every node has done so cannot be locally detected.

Hence deducing one property knowing another relates to the termination detection of recognition algorithms. Not surprisingly, our equivalence will be expressed in terms of closure under some quasi-coverings based relation between graphs. However, this equivalence cannot be derived straightly from the two previous results, because of the difference of approach in the termination requirement. The proof techniques are very similar however. It might suggest that there is a common property that could be applied to the different termination flavors. A similar approach has also been proved useful for the study of the election problem [5]. The result presented here, besides its own interest, can also be seen as a new investigation of the problem to determine what is and what is not solvable by distributed algorithms. The applications presented give also new insight about the various (quasi-)coverings based characterizations.

**The Model.** We consider networks of processors with arbitrary topology. A 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. A distributed algorithm is a set of atomic read&write rules. Labels are modified locally, that is, on a subgraph of fixed radius 1 of the given graph, according to certain rules depending on the subgraph only (local computations). The relabelling is performed until no more transformation is possible, i.e., until an irreducible form is obtained. Communication between nodes is “embedded” into the read&write scheme.
This model has several interests: it gives an abstract model to think about some problems in the field of distributed computing independently of the wide variety of models used to represent distributed systems, impossibility results remains true in weaker models, any positive solution in this model may guide the research of a solution in a weaker model or be implemented in a weaker model using randomized algorithms. This model gives nice properties and examples using classical combinatorial material.

**The Result and Applications.** Given a structural knowledge, we define a relation between graphs when they share the same value for this structural knowledge and we can partially simulate computations between them. This is formally defined with quasi-coverings Sect. 3.2. Two structural knowledges are then equivalent if and only if they define an identical relation.

Studying the equivalence class of a given knowledge help in the understanding of what knowledge is necessary to solve a distributed problem. The gap, proved here, between zero-knowledge and knowledge of a bound on the diameter is in particular very interesting. Th. 2, and consequence Th. 3 leads to interesting corollaries such as, with no knowledge, we can even not determine whether the graph has a cycle or not; or the knowledge of a bound on the diameter is equivalent to the knowledge of a bound on the size, but also to new arguments for the use of symmetry-free graphs. For those graphs, the covering-minimal graphs, the knowledge of a bound on the diameter enables to compute the entire topology of the graph, hence to deduce anything wanted.

**Proof techniques.** The proofs are based upon coverings and quasi-coverings as in [4, 8, 5]. However presentation and proofs of, e.g., lemmas 2 and 4 are new.

**Related Works.** Among models related to our model there are local computation systems as defined by Angluin [1], Yamashita and Kameda [13] and Boldi and Vigna [2]. In [1] an asynchronous model is considered. A basic computation step means that two adjacent vertices exchange their labels and then compute new ones. In [13] an asynchronous model is studied where a basic computation step means that a processor either changes its state and sends a message or it receives a message. In this framework, Sakamoto [10] has compared different “initial assignments” and proved they induce an infinite lattice. In [2] networks are directed graphs coloured on their arcs; each processor changes its state depending on its previous state and on the states of its in-neighbours. Activation of processors may be synchronous, asynchronous or interleaved. Equivalence of knowledge of the size and of the diameter was already proved in this framework.

## 1 Graphs, Labelled Graphs and Coverings

We only consider finite, undirected and connected graphs without multiple edges and self-loops. If $G$ is a graph, $V(G)$ denotes the set of vertices and $E(G)$ denotes the set of edges. Let $v$ be a vertex, we denote by $B_G(v,k)$, or briefly $B(v,k)$, the
centered ball of radius \( k \) with center \( v \). \( 1 \)-balls might also be noted \( B_G(v) \). The set of neighbours of a vertex \( v \) in \( G \) is denoted by \( N_G(v) \).

A homomorphism between two graphs \( G \) 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, this implies that \( \gamma(u) \neq \gamma(v) \) if \( \{u, v\} \) is an edge of \( G \). Note also that \( \gamma(N_G(u)) \subseteq N_H(\gamma(u)) \). We say that \( \gamma \) is an isomorphism if \( \gamma \) is bijective and \( \gamma^{-1} \) is also a homomorphism. By \( G \simeq G' \) we mean that \( G \) and \( G' \) are isomorphic. A class of graphs will be any class of graphs in the set-theoretical sense containing all graphs isomorphic to some of its members. The class of all graphs will be denoted \( G \).

Throughout the paper we will consider only connected graphs where vertices and edges are labelled with labels from a possibly infinite alphabet \( L \). A graph labelled over \( L \) will be denoted by \( (G, \lambda) \), 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 \( \lambda \) is a labelling of \( G \). The class of labelled graphs over some fixed alphabet \( L \) will be denoted by \( G_L \). Let \( \alpha \) be a label, \( (G, \Lambda_\alpha) \) denotes the labelled graph with underlying graph \( G \), uniformly labelled by \( \alpha \).

Let \( (G, \lambda) \) and \( (G', \lambda') \) be two labelled graphs. Then \( (G, \lambda) \) is a subgraph of \( (G', \lambda') \), denoted by \( (G, \lambda) \subseteq (G', \lambda') \), if \( G \) is a subgraph of \( G' \) and \( \lambda \) is the restriction of the labelling \( \lambda' \) to \( V(G) \cup E(G) \). A mapping \( \varphi \) is a homomorphism from \( (G, \lambda) \) to \( (G', \lambda') \) if \( \varphi \) is a graph homomorphism from \( G \) to \( G' \) which preserves the labelling, i.e. such that \( \lambda'(\varphi(x)) = \lambda(x) \) holds for every \( v \in V(G) \) and if \( \{u, v\} \) is an edge of \( G \) then \( \lambda(\{u, v\}) = \lambda(\{\varphi(u), \varphi(v)\}) \). A labelled graph will often be designed by a bold letter like \( G, H \) etc ...

An occurrence of \( (G, \lambda) \) in \( (G', \lambda') \) is an isomorphism \( \varphi \) between \( (G, \lambda) \) and a subgraph \( (H, \eta) \) of \( (G', \lambda') \). We say that a graph \( G \) is a covering of a graph \( H \) if there exists a surjective homomorphism \( \gamma \) from \( G \) onto \( H \) such that for every vertex \( v \) of \( V(G) \) the restriction of \( \gamma \) to \( B_G(v, 1) \) is a bijection onto \( B_H(\gamma(v), 1) \). The covering is proper if \( G \) and \( H \) are not isomorphic. It is called connected if \( G \) (and thus also \( H \)) is connected. We extend the notion of covering to labelled graphs in an obvious way. The labelled graph \( (H, \lambda') \) is covered by \( (G, \lambda) \) via \( \gamma \), if \( \gamma \) is a homomorphism from \( (G, \lambda) \) to \( (H, \lambda') \) whose restriction to \( B_G(v, 1) \) is an isomorphism from \( (B_G(v, 1), \lambda) \) to \( (B_H(\gamma(v), 1), \lambda') \).

A graph \( G \) is called covering-minimal if every covering from \( G \) to some \( H \) is a bijection. Graphs with relatively prime number of vertices and number of edges, trees, labelled graphs with a distinguished vertex, graphs with nodes having different identification numbers are examples of covering-minimal graphs.

2 Local Computations in Graphs

Graph relabelling systems and more generally local computations satisfy the following constraints which seem to be natural when describing distributed compu-
notations with a decentralized control: 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; they are local, that is, each relabelling step changes only a connected subgraph of a fixed size in the underlying graph; they are locally generated, that is, the application condition of the relabelling only depends on the local context of the relabelled subgraph.

2.1 Local Computations

Local computations as considered here can be described in the following general framework. Let \( \mathcal{R} \subseteq \mathcal{G}_L \times \mathcal{G}_L \) be a binary relation on \( \mathcal{G}_L \). Then \( \mathcal{R} \) will denote a graph rewriting relation. We assume that \( \mathcal{R} \) is closed by isomorphism, i.e., whenever \( G \mathcal{R} G' \) if \( H \simeq G \) then \( H \mathcal{R} H' \) for some labelled graph \( H' \simeq G' \). In the remainder of this paper \( \mathcal{R}^* \) stands for the reflexive and transitive closure of \( \mathcal{R} \). The labelled graph \( G \) is \( \mathcal{R} - \text{irreducible} \) if there is no \( G' \) such that \( G \mathcal{R} G' \). Let \( G \in \mathcal{G}_L \), then \( \text{Irred}_{\mathcal{R}}(G) \) denotes the set of \( \mathcal{R} - \text{irreducible} \) graphs (or just irreducible if \( \mathcal{R} \) is fixed) which can be obtained from \( G \) using \( \mathcal{R} \). The relation \( \mathcal{R} \) is noetherian if there is no infinite relabelling chain \( G_1 \mathcal{R} G_2 \mathcal{R} \ldots \)

Definition 1 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 i.e.: \( G \mathcal{R} H \implies G = H \).

2. \( \mathcal{R} \) is local if only labels of a ball of radius 1 may be changed by \( \mathcal{R} \), i.e., \( (G, \lambda) \mathcal{R} (G, \lambda') \) implies that there exists a vertex \( v \in V(G) \) such that \( \lambda(x) = \lambda'(x) \) for every \( x \not\in V(B_G(v, 1)) \cup E(B_G(v, 1)) \).

The next definition states that a local relabelling relation \( \mathcal{R} \) is locally generated if its restriction on centered balls of radius 1 determines its computation on any graph.

Definition 2 Let \( \mathcal{R} \) be a local relabelling relation. Then \( \mathcal{R} \) is locally generated if the following is satisfied: For any labelled graphs \( (G, \lambda), (G, \lambda'), (H, \eta), (H, \eta') \) and any vertices \( v \in V(G), w \in V(H) \) such that the balls \( B_G(v, 1) \) and \( B_H(w, 1) \) are isomorphic via \( \varphi: V(B_G(v, 1)) \rightarrow V(B_H(w, 1)) \) and \( \varphi(v) = w \), the following three conditions

1. \( \lambda(x) = \eta(\varphi(x)) \) and \( \lambda'(x) = \eta'(\varphi(x)) \) for all \( x \in V(B_G(v, 1)) \cup E(B_G(v, 1)) \)

2. \( \lambda(x) = \lambda'(x) \), for all \( x \not\in V(B_G(v, 1)) \cup E(B_G(v, 1)) \)

3. \( \eta(x) = \eta'(x) \), for all \( x \not\in V(B_H(w, 1)) \cup E(B_H(w, 1)) \)

imply that \( (G, \lambda) \mathcal{R} (G, \lambda') \) if and only if \( (H, \eta) \mathcal{R} (H, \eta') \).
2.2 Local Computations, Coverings and Quasi-Coverings

The fundamental lemma which connects coverings and locally generated relabelling relations states that, because the preimage of a 1-ball through a covering is a collection of disjoint isomorphic balls, whenever \( G \) is a covering of \( H \), every local computation in \( H \) can be lifted to a local computation in \( G \) which is compatible with the covering relation.

**Lemma 1 (Lifting Lemma)** Let \( \mathcal{R} \) be a locally generated relabelling relation and let \( G \) be a covering of \( H \) via \( \gamma \). Moreover, let \( H \stackrel{\mathcal{R}}{\rightarrow} H' \). Then there exists \( G' \) such that \( G \stackrel{\mathcal{R}}{\rightarrow} G' \) and \( G' \) is a covering of \( H' \).

![Figure 1: \( G \) is a quasi-covering of \( H \) of radius \( r \)](image)

**Definition 3** Let \( G, H \) be two labelled graphs and let \( \gamma \) be a partial function on \( V(G) \) that assigns to each element of a subset of \( V(G) \) exactly one element of \( V(H) \). Then \( G \) is a quasi-covering of \( H \) via \( \gamma \) of radius \( r \) if there exists a finite or infinite covering \( G_0 \) of \( H \) via \( \delta \), vertices \( z_0 \in V(G_0) \), \( z \in V(G) \) such that:

1. \( B_G(z, r) \) is isomorphic via \( \varphi \) to \( B_{G_0}(z_0, r) \),
2. the domain of definition of \( \gamma \) contains \( B_G(z, r) \), and
3. \( \gamma = \delta \circ \varphi \) when restricted to \( V(B_G(z, r)) \).

\( \text{card}(\varphi(B_G(z, r))) \) is called the size of the quasi-covering, and \( z \) the center. The graph \( G_0 \) is called the associated covering of the quasi-covering. See Fig. 1.

Quasi-coverings have been introduced to study the problem of the detection of the termination in [7]. The idea behind them is to enable the partial simulation
of local computations on a given graph in a restricted area of a larger graph. The restricted area where we can perform the simulation will shrink while the number of simulated steps increases. The following lemma precisely how much the radius shrinks when one step of simulation is performed:

**Lemma 2 (Quasi-Lifting Lemma)** Let \( \mathcal{R} \) be a locally generated relabelling relation and let \( G \) be a quasi-covering of \( H \) of radius \( r \) via \( \gamma \). Moreover, let \( H \mathcal{R} H' \). Then there exists \( G' \) such that \( G \mathcal{R}^+ G' \) and \( G' \) is a quasi-covering of radius \( r - 2 \) of \( H' \).

**Proof.** Let \( G_0 \) be the associated covering and \( z \) be the center of the ball of radius \( r \). Suppose now the relabelling step \( H \mathcal{R} H' \) apply rule \( R_0 \) and modify labels in \( B_H(v) \), for a given \( v \in V(H) \). \( R_0 \) can also be applied to all the balls \( \delta^{-1}(B_H(v)) \) yielding \( G_0' \) and \( \delta' \). It applied also to the balls \( \gamma^{-1}(B_H(v)) \) that are included in \( B_G(z, r) \), since they are also isomorphic to \( B_H(v) \). We get \( G' \) and \( \gamma' \) satisfying the quasi-covering properties with radius \( r - 2 \): consider \( w \) in \( B_{G'}(z, r - 2) \): since any ball containing \( w \) is included in \( B_G(z, r) \), \( w \) and \( \gamma(w) \) have the same label. \( \square \)

# 3 Structural Knowledge

## 3.1 Definition and Examples

**Definition 4** A structural knowledge is a computable mapping \( \kappa: G_L \rightarrow L \).

Given two structural knowledge \( \iota \) and \( \kappa \), we say that \( \kappa \) is deducible from \( \iota \), noted \( \iota \vdash \kappa \), if there exists a relabelling system \( \mathcal{R} \) such that, for all graph \( G \in G_L \),

- any relabelling chain \((G, \Lambda_1(G)) \mathcal{R} \ldots \mathcal{R} (G, \Lambda_{\ell}(G)) \mathcal{R} \ldots \) is finite.
- \((G, \Lambda_{\kappa(G)})\) is the only \( \mathcal{R} \)-normal form for \((G, \Lambda_{\iota(G)})\).
- for every labelling \( \Lambda \), if there exists a vertex \( u \) such that \( \Lambda(u) = \kappa(G) \) then for every \( \Lambda' \) such that \((G, \Lambda) \mathcal{R}^+ (G, \Lambda') \), \( \Lambda'(u) = \kappa(G) \). We say that \( \iota \) and \( \kappa \) are equivalent if \( \iota \vdash \kappa \) and \( \kappa \vdash \iota \). We note \( \kappa \equiv \iota \) in this case.

Some examples of structural knowledge are given below:

- **“Metrics”** knowledges
  - An upper bound \( b \) on the size of the graph: this is a function \( b \) such that \( b(G) \geq |V(G)| \) for all \( G \).
  - The size, i.e. the number of vertices \( |V(G)| \).
  - The diameter \( \Delta(G) \) of the graph. A bound on the diameter.
  - If no knowledge at all is available, we say that we are in the zero-knowledge case and we denote it by \( \varepsilon \).

- **“Characteristic”** or boolean knowledge: whether the graph is or is not in a given class of graphs \( \mathcal{C} \). We note \( \chi_\mathcal{C} \) the structural knowledge mapping graphs of \( \mathcal{C} \) to label \( \text{TRUE} \) and graphs of \( G \backslash \mathcal{C} \) to label \( \text{FALSE} \). Using characteristic knowledge, one can address, in our settings, the problem of, e.g., computing the size in a particular class of graphs.
• Topology knowledge (e.g. the adjacency matrix of the underlying graph is given).

Remark: Formally, the structural knowledge we deal with are scalar, as opposed to knowledges such as the presence of unique identifiers or of one vertex labelled initiator. The reader should keep in mind the distinction between structural knowledge (a property of the underlying graph that is uniformly encoded) and structural information (distributively encoded informations such as distinct identities, sense of direction, . . .).

3.2 Relations for simulations

We will define some relations between graphs that will translate the fact that some computations on a graph can be partially simulated on another.

Definition 5 Let κ be a structural knowledge, \( r : G \to \mathbb{N} \) a function. Given two graphs \( G \) and \( G' \), we say that \( G \sigma^\kappa \underline{r} G' \) if \( \kappa(G) = \kappa(G') \) there exists \( H \) such that both \( G \) and \( G' \) are quasi-coverings of \( H \) of radius \( r(G) \).

We note \( \sim^\kappa \) the equivalence relation defined by the symmetric and transitive closure of \( \sigma^\kappa \).

By extension, for \( r_0 \in \mathbb{N} \), \( \sim_0^\kappa \) denotes the equivalence relation for constant function \( G \mapsto r_0 \). Given an equivalence relation \( \sim \), for any graph \( G \), we note \([G]_\sim\) the equivalence class of \( G \). If \( \sim_1 \) and \( \sim_2 \) are two equivalence relations such that, for all \( G \), \([G]_\sim_1 \subset [G]_\sim_2 \), we note \( \sim_1 \subset \sim_2 \).

4 Two Fundamental Algorithms

4.1 The Mazurkiewicz Enumeration Algorithm

A distributed enumeration algorithm on a graph \( G \) 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 [6] Mazurkiewicz presents a distributed enumeration algorithm for the class of graphs minimal for the covering relation. In the following we give a description of the Mazurkiewicz algorithm including its extension to labelled graphs. This algorithm will be denoted by \( \mathcal{M} \). We first give a general description of the algorithm \( \mathcal{M} \) applied to the graph \( G \). Let \( G = (G, \lambda) \), let \( v_0 \) be a vertex of \( G \), let \( \{v_1, \ldots, v_d\} \) be the set of neighbours of \( v_0 \).

The label of the vertex \( v_0 \) used by \( \mathcal{M} \) is the tuple \( (\lambda(v_0), (n(v_0), N(v_0), M(v_0)) \) representing the following information 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 $v_0$, it is the set of triples defined by :
$$\{(n(v_i), \lambda(v_i), \lambda(\{v_0,v_i\})) | 1 \leq i \leq d\}$$

• $M(v_0) \subset \mathbb{N} \times L \times \mathcal{N}$ is the mailbox of $v_0$ and contains all the information received by $v_0$ at this step of the computation.

Every vertex $v$ attempts to get its own number $n(v)$, which shall 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” in some sense to be precised later, then $u$ chooses another number. Each new number, with its local view, is broadcasted again over the network. At the end of the computation it is not guaranteed that every node has a unique number, unless the graph is covering-minimal. However, all nodes 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 that the set of labels $L$ is equipped with a total order denoted $>_L$. Let $v_0$ be a vertex and let $\{v_1, ..., v_d\}$ the neighbours of $v_0$ we assume that: $n(v_1) \geq n(v_2) \geq ... \geq n(v_d)$, if $n(v_i) = n(v_{i+1})$ then $\lambda(v_i) \geq \lambda(v_{i+1})$, if $n(v_i) = n(v_{i+1})$ and $\lambda(v_i) = \lambda(v_{i+1})$ then $\lambda(\{v_0,v_{i+1}\}) \geq_L \lambda(\{v_0,v_i\})$. Then to $N(v)$, the local view, we associate the $d$-tuple $((n(v_1), \lambda(v_1), \lambda(\{v_0,v_1\})), ..., (n(v_d), \lambda(v_d), \lambda(\{v_0,v_d\})))$. Let $\mathcal{N}_{>L}$ be the set of such ordered tuples. We define a total order, $\prec$, on $\mathcal{N}_{>L}$ by comparing the numbers, then the vertex labels and finally the edge labels. Formally, let $((n_1, l_1, e_1), ..., (n_d, l_d, e_d))$ and $((n'_1, l'_1, e'_1), ..., (n'_d, l'_d, e'_d))$ be two elements of $\mathcal{N}_{>L}$ then $((n'_1, l'_1, e'_1), ..., (n'_d, l'_d, e'_d)) \prec ((n_1, l_1, e_1), ..., (n_d, l_d, e_d))$ if one of the following holds

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

The initial labelling of the vertex $v_0$ is $(\lambda(v_0), (0, 0, 0))$. The rules are described below for a given centered ball $B = B(v_0, 1)$ with center $v_0$. The vertices $v$ of $B$ have labels $(\lambda(v), (n(v), N(v), M(v)))$. The labels obtained after applying a rule are $(\lambda(v), (n'(v), N'(v), M'(v)))$. To make the rules easier to be read, we omit labels that are left unchanged.
$M-1$ : Diffusion rule

**Precondition :**
- $\exists v \in B(v_0, 1), M(v) \neq M(v_0)$

**Relabelling :**
- For all $v \in B(v_0, 1), M'(v) := \bigcup_{w \in B} M(w)$

$M-2$ : Renaming rule

**Precondition :**
- $\forall v, M(v) = M(v_0)$
- 1. $n(v_0) = 0$
- or
- 2. $(n(v_0) > 0$ and $\exists (n(v_0), l, N) \in M(v_0)$) such that
  $((\lambda(v_0) < l) \text{ or } ((\lambda(v_0) = l) \text{ and } (N(v_0) \prec N))$

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

Let $G$ be a labelled graph, if $v$ is a vertex of $G$, the label of $v$ after a run $\rho$ of
the Mazurkiewicz algorithm is denoted $(\lambda(v), n_\rho(v), N_\rho(v), M_\rho(v))$.

**Theorem 1** [6] Any run $\rho$ of Mazurkiewicz' enumeration algorithm on a connected
labelled graph $G = (G, \lambda)$ terminates and yields a final labelling $(\lambda(v), n_\rho(v), N_\rho(v), M_\rho(v))$ verifying the following conditions for all vertices $v, v'$ of $G$:

1.i $M_\rho(v) = M_\rho(v')$.

1.ii $(n_\rho(v), \lambda(v), N_\rho(v)) \in M_\rho(v')$.

1.iii $n_\rho(v) = n_\rho(v')$ implies $(\lambda_\rho(v) = \lambda_\rho(v')$ and $N_\rho(v) = N_\rho(v'))$

As in [4], we interpret the final labelling as a graph that each vertex can compute. For a mailbox $M$, we define for each integer $n \in \{1, \ldots, |V(G)|\}$ the maximal mail box element of the form $(n, l, N) : F(M) = \{(n, l, N) \in M \mid$ there is no $(n, l', N') \in M$ verifying $l' > l$ or $(l' = l$ and $N \preceq N')\}$.

For a given mailbox $M$, we define the graph $G(M)$ to be the following graph:

$V(G(M)) = \{n \mid \exists N, l, (n, l, N) \in F(M)\}$

$E(G(M)) = \{(n, n') \mid (n, l, N) \in F(M)$, and $N = (\ldots, (n', l', l''), \ldots)\}$
We also define a labelling on $G(M)$ as follows, $\lambda_M(n) = (l, n, N, M)$, with $(n, l, N) \in F(M)$. Note that uniqueness of $l$ and $N$ comes from the definition of $F(M)$. Let $\rho$ be a run of $M$, then $(G(M_\rho(u)), \lambda_{M_\rho(u)})$ does not depend on $u$ by Th. 1. We then define $G_\rho = (G(M_\rho(u)), \lambda_{M_\rho(u)})$. From Th. 1.iii, we can see a run of $M$ as computing a covering:

**Proposition 1** Let $G$ be a graph, and $\rho$ a run of $M$, $G$ is a covering of $G_\rho$.

If the underlying graph is covering-minimal, then $G_\rho$ is an isomorphic reconstruction of $G$. The algorithm of Mazurkiewicz can actually be seen as a reconstruction algorithm.

### 4.2 An Algorithm to Detect Stable Properties

In this section we describe in our framework the algorithm by Szymanski, Shy and Prywes (the SSP algorithm for short) [11]. We consider a distributed algorithm which terminates when all processes reach their local termination conditions. Each process is able to determine only its own termination condition. The SSP algorithm detects an instant in which the entire computation is achieved. Let $\rho$ be a vertex and $\rho'$ be a run of $\rho$. We consider a distributed algorithm $\rho(t)$ which terminates when all processes reach their local termination conditions.

In [11] the following assumption is considered: for each node $v$ the value of $P(v)$ eventually becomes true and remains true for ever. In this paper we present a generalization of the hypothesis under which the SSP algorithm is run. For each node $v$ the value of $P(v)$ eventually becomes true and either remains true for ever or becomes false and remains false for ever. We say that, in this case, the predicate has the **trueness connectivity property**. We denote this generalization by GSSP algorithm. We will use the following notation. Let $(G_i)_{0 \leq i}$ be a relabelling chain associated to the GSSP algorithm. We denote by $a_i(v)$ (resp. $p_i(v)$) the integer (resp. the boolean) associated to the vertex $v$ of $G_i$.

According to the SSP rule, we remark that for a vertex $v$, if $a(v)$ has increased from $h - 1$ to $h$, that means that, at the previous step, all its neighbours $w$ were such that $a(w) \geq h - 1$. The following lemma is the iterated version of this remark.

**Lemma 3** For all $j$, for all $v$, for all $w \in B(v, a_j(v))$, there exists $i \leq j$ such that $a_i(w) \geq a_j(v) - d(v, w)$.

In particular that proves that for every $j$, for every $v$, for every $w \in B(v, a_j(v))$, there exists $i_w < j$ such that $P_{i_w}(w)$ is TRUE. We show now that, for all $w$ in the ball of radius $\left\lfloor \frac{a_j(v)}{2} \right\rfloor$, we can choose the same $i_w$. This is the fundamental property of the GSSP algorithm.
Lemma 4 For all \( j \), for all \( v \), such that \( P_j(v) \) is TRUE, there exists \( i \leq j \) such that for all \( w \in B(v, [\frac{a_j(v)}{3}]) \), \( P_i(w) \) is TRUE.

Proof. Let \( i \) be the first step such that \( a_i(v) = [\frac{a_j(v)}{3}] \). Let \( w \in B(v, [\frac{a_j(v)}{3}]) \), note \( i_+ \) a step, which existence is given by lemma 3, such that \( a_{i_+}(w) \geq a_j(v) - d(v,w) \geq 2[\frac{a_j(v)}{3}] \). Now, we apply lemma 3 centered on \( w \) at step \( i_+ \). We have then \( i' \leq i_+ \) such that \( a_{i'}(v) \geq a_{i_+}(w) - d(w,v) \geq [\frac{a_j(v)}{3}] = a_i(v) \). Hence by minimality of \( i \), \( i \leq i' \) and finally \( i \leq i_+ \).

Now, we once again apply lemma 3, centered in \( v \), at step \( i \). We get \( i_- \leq i \), such that \( a_{i_-}(w) \geq a_i(v) - d(v,w) \geq 0 \).

To conclude, we have steps \( i_- \) and \( i_+ \) such that \( P_{i_-}(w) = P_{i_+}(w) = \text{TRUE} \), and \( i_- \leq i \leq i_+ \). By trueness connectivity of the predicate, \( P_i(w) = \text{TRUE} \). \( \square \)

4.3 Mazurkiewicz Algorithm + GSSP Algorithm = Graph Quasi-Reconstruction Algorithm

Seeing Mazurkiewicz algorithm as a reconstruction algorithm, one can wonder how do determine when the reconstruction is completed or, alternatively, what useful partial reconstruction information shall we get in the course of the computation. The main idea developed in this section is to use the GSSP algorithm for computing the radius of stability of \( \mathcal{M} \) for any vertex. Any vertex will know until which distance all vertices agree with its reconstruction of the underlying graph topology. Let \( G = (G, \lambda) \) be a labelled graph, let \( (G_i)_{0 \leq i} \) be a relabelling chain associated to a run of the Mazurkiewicz algorithm on the graph \( G \). To the node \( v \) of \( G_i \) is associated the label \( (\lambda(v), (n_i(v), \lambda_i(v), M_i(v))) \). Using the interpretation of Subsection 4.1, it is possible to reconstruct a graph.

We now associate an infinite family of predicate stating which is the current mailbox (and indirectly which graph is reconstructed). We introduce on the node \( v \) of the graph \( G_i \) the predicate \( p_M(v) \), that will be true if label of \( v \) in \( G_i \) is such that \( M_i(v) = M \) and \( (n_i(v), \lambda_i(v), N_i(v)) \in F(M_i(v)) \). The associated value \( a_M(v) \) is computed as previously defined by the GSSP algorithm. The trueness connectivity property of the \( p_M(v) \) predicates is a consequence of the fact that, by construction, mailboxes \( M_i(v) \) are increasing during the computation.

We note QR algorithm (for quasi-reconstruction) the merging of the Mazurkiewicz algorithm and the potentially infinite instances of GSSP algorithm. Now we formalize the output \( <\mathbf{H}(v), a(v)> \) of QR for a given vertex \( v \).

\[
\begin{align*}
\mathbf{H}(v) & = \begin{cases} 
G(M_i(v)) & \text{if } (n_i(v), \lambda_i(v), N_i(v)) \in F(M_i(v)) \\
\bot & \text{otherwise.}
\end{cases} \\
a(v) & = \begin{cases} 
-1 & \text{if } \mathbf{H}(v) = \bot \\
a_M(v) & \text{otherwise.}
\end{cases}
\end{align*}
\]
Note that, while \( H(v) = \perp \), the node knows that it is in a non final state. The main property of \( QR \) is a direct consequence of lemma 4:

**Proposition 2 (quasi-covering reconstruction)** At all step \( j \), for all vertex \( v \), the output of \( QR \) on \( v \) is a couple \( < H, a > \) such that whenever \( H \neq \perp \), there exists a previous step \( i < j \), such that \( G_i \) is a quasi-covering of \( H \) of center \( v \) and of radius \( \left\lfloor \frac{a_i}{3} \right\rfloor \).

**Sketch of proof.** \( i \) is given by Lem. 4. \( M_i \) is then constant on the ball \( B_{G}(v, \left\lfloor \frac{a_i}{3} \right\rfloor) \). As associated covering, we will append some ad hoc infinite trees to a copy of \( B_{G_i}(v, \left\lfloor \frac{a_i}{3} \right\rfloor) \). For all vertex \( w_0 \) in \( H \), we define \( \Pi(w_0, w_1, \ldots, w_n) \) to be the subtree of the universal covering of \( H \) induced by the set of paths originating from \( w_0 \) and starting by any of the the vertex \( w_k \), \( k \in [1, n] \). For all \( w \) such that \( d(w, w') = \left\lfloor \frac{a_i}{3} \right\rfloor \), we define \( H_w \) to be an isomorphic copy of \( \Pi(n_i(w), w_1, \ldots, w_n) \), where \( \{w_1, \ldots, w_n\} = \{s \mid n_i(w), s \in V(H), \forall u \in N(w) n_i(u) \neq s\} \). We complete a copy \( \Pi \) of \( B_{G}(v, \left\lfloor \frac{a_i}{3} \right\rfloor) \) by the \( H_w \), for all \( w \) such that \( d(v, w) = \left\lfloor \frac{a_i}{3} \right\rfloor \), identifying \( w \) and the root of \( H_w \). Verifying that we have all quasi-covering properties is then quite straightforward.

We can now state the deductibility and equivalence theorems

### 5 Equivalence of structural knowledges

**Theorem 2 (deductibility)** Let \( \kappa \) and \( \tau \) two structural knowledges. Then \( \tau \vdash \kappa \) if and only if there exist a computable function \( r : G_L \rightarrow \mathbb{N} \), such that \( \sim_{\kappa} \subseteq \sim_{\tau} \).

**Proof.** **Necessary condition** Suppose the left hand side holds and note \( \mathcal{R} \) the relabelling system computing \( \kappa \) from \( \tau \).

Let \( C(G) = \{H | G \text{ is a covering of } H\} \). Since \( (G, \Lambda_{\kappa(G)}) \mathcal{R} (G, \Lambda_{\tau(G)}) \), from the Lifting Lemma we get that for any \( H \in C(G) \), \( (H, \Lambda_{\kappa(G)}) \mathcal{R} (H, \Lambda_{\tau(G)}) \). For any \( H \), we note \( n_H \) the length of such a relabelling chain. Now we define \( r(G) = 1 + \max_{H \in C(G)} \{ n_H, |V(G)| \} \). As \( C(G) \) is finite, this maximum is always defined.

Now, we consider a graph \( G' \) such that \( G \sigma^i \sigma^r G' \). There exists \( H \) such that \( G \) and \( G' \) are both quasi-coverings of \( H \) of radius \( r(G) \). As \( r(G) > |V(G)| \), \( G \) is in fact a covering of \( H \). Or, in other words, \( H \in C(G) \). By \( n_H \) iterations of the Quasi-Lifting Lemma for \( G \) and \( H \), for the relabelling chain associated to \( n_H \), we get that there exist \( \Lambda \) such that \( (G', \Lambda_{\kappa(G')}) \mathcal{R} (G', \Lambda) \), with \( (G', \Lambda) \) being a quasi-covering of \( (H, \Lambda_{\kappa(G)}) \) of radius at least 1. Hence the label \( \kappa(G) \) appears in at least one vertex of \( G', \Lambda \). As \( \mathcal{R} \) computes \( \kappa \), we deduce that \( \kappa(G') = \kappa(G) \).

Hence \( G \sigma^i G' \Rightarrow G \sigma^r G' \) for all \( G \) and \( G' \). By symmetric and transitive closure, we get \( \sim_{\kappa} \subseteq \sim_{\tau} \).

**Sufficient condition** We choose an (arbitrary) enumeration of the set of all labelled graphs \( G_L \) and on each node, we use the following algorithm based upon \( QR \).
repeat
  get $H,a$ from QR algorithm
repeat
  get $K$ from the enumeration of $G_L$
until $\begin{cases} 1(K) = 1(G) \\ K \text{ is a quasi-covering of } H \text{ of radius } \left\lfloor \frac{a}{3} \right\rfloor. \end{cases}$ \hspace{1cm} (*)
until $r(K) \leq \left\lfloor \frac{a}{3} \right\rfloor$
Output : $\kappa(K)$

By Lem. 2, $G_i$, in the notations of the lemma, satisfies the constraint for the termination of the $G_L$ enumeration repeat-until loop. Hence this inner loop always terminates during the computation.

As $M$ terminates, $H$ will eventually become constant and $a$ will go to the infinite. Hence, if we note $G_f$ the final graph, we have by Prop. 1 that $G_f$ is a solution for (*)). Furthermore, if we note $r_0$ the maximum of $r$ over the finite set of the first enumerated graphs until $G_f$ is reached, we have that the inequality will be satisfied for $a = 3r_0$. Hence the outer loop will also finish.

Verifying the correctness is then straightforward. We have, always by Lem. 2 that $K \sigma^1_r G_f$, hence by hypothesis, applied for $K$, $\kappa(G) = \kappa(K)$. □

**Theorem 3 (equivalence)** Let $\kappa$ and $\iota$ be two structural knowledges. Then $\kappa \equiv \iota$ if and only if there exists a computable function $r : G_L \rightarrow \mathbb{N}$, such that $\sim^1_r \equiv \sim^\kappa_r$.

### 6 Applications

#### 6.1 The zero-knowledge case

**Proposition 3** Let $r : G_L \rightarrow \mathbb{N}$. For all graph $G$, there exists a tree $T$ such that $G \sigma^F_r T$.

This yields some negative results: $\varepsilon \not\subseteq \mathcal{L}_f$. That means that there is no way, without any knowledge to deduce whether the underlying graph is a tree or not. Moreover, if $G$ and $G'$ are not trees we can choose the same $T$. In other words, for all non-tree graphs $G$ and $G'$, $G \sim^F_r G'$.

For further description, we use a definition coming from trees automata theory, that describes trees with some regularity[3]. A regular family of trees is a family such that there exists a tree-automaton that recognizes this family. A strongly irregular family $S$ of trees is a family of trees such that no infinite subset of balls of graphs in $S$ are regular. The family of “$n$—stars” (a vertex with exactly $n$ neighbours) is an example of such a family. A consequence of the pumping lemma [3], expressing underlying regularity, for regular classes of trees is the following:
Proposition 4 Let $\kappa$ be a structural knowledge. Then $\kappa \equiv \varepsilon$ if and only if there exists a strongly irregular family of graphs $S$ such that the $\approx^\kappa$-equivalence class of any non-tree graph $G$ is $[G]_{\approx^\kappa} = G_L \setminus S$.

6.2 The bounded diameter case

If we know a bound on the diameter, we can detect the termination of $M$. The problem is that the reconstructed graph $G_\rho$ is not necessarily isomorphic to the underlying graph. These restriction come from the distributed aspect of the computation. But we shall notice that restricted to minimal graphs (that is a very large family of graphs) this knowledge enables the computation of the topology. Hence, for minimal graphs the hierarchy of knowledges collapses very quickly.

More generally, we remark that given a graph $H$ there is a finite number of coverings $G$ with a given diameter. So, taking the maximum size as $r(H)$, we see that the knowledge of a bound on the size is deducible, hence equivalent, from the knowledge of a bound on the diameter. However, size and diameter are not deducible one from the other, consider Fig. 2, where the 3 graphs on the top are coverings of the one on the bottom. We have, for any function $r$, $G_1 \approx^\text{size} G_2$ and not $G_1 \approx^\Delta G_2$. We have also $G_2 \approx^\Delta G_3$ and not $G_2 \approx^\text{size} G_3$.

![Figure 2: Diameter and size knowledge are not comparable](image)

References


Emmanuel Godard and Yves Métivier are with the LaBRI, Université Bordeaux I - ENSEIRB, 351, cours de la libération, 33405 Talence France. E-mail: {godard | metivier}@labri.fr