Deciding and verifying network properties locally with few output bits

Heger Arfaoui, Pierre Fraigniaud, David Ilcinkas, Fabien Mathieu, Andrzej Pelc

Noname manuscript No.
(will be inserted by the editor)

Deciding and verifying network properties locally with few output bits

Heger Arfaoui · Pierre Fraigniaud · David Ilcinkas · Fabien Mathieu · Andrzej Pelc

the date of receipt and acceptance should be inserted later

Abstract Given a boolean predicate on labeled networks (e.g., the network is acyclic, or the network is properly colored, etc.), deciding in a distributed manner whether a given labeled network satisfies that predicate typically consists, in the standard setting, of every node inspecting its close neighborhood, and outputting a boolean verdict, such that the network satisfies the predicate if and only if all nodes output true. In this paper, we investigate a more general notion of distributed decision in which every node is allowed to output a constant number $b \geq 1$ of bits, which are gathered by a central authority emitting a global boolean verdict based on these outputs, such that the network satisfies the predicate if and only if this global verdict equals true. We analyze the power and limitations of this extended notion of distributed decision.

Keywords Distributed decision · Distributed verification · Locality · Network

Preliminary versions of this work appeared as extended abstracts in the proceedings of the 40th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2014) [4], and of the 15th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2013) [5].

P. Fraigniaud receives additional support from the ANR project DESCARTES (ANR-16-CE40-0023), and from the INRIA project GANG.

D. Ilcinkas is partially supported by the ANR projects DESCARTES (ANR-16-CE40-0023) and ESTATE (ANR-16-CE25-0009). This study was carried out in the framework of the program "investment for the future" of Idex Bordeaux-CPU (ANR-10-IDEX-03-02).

A. Pelc is partially supported by NSERC discovery grant 2018-03899 and by the Research Chair in Distributed Computing at the Université du Québec en Outaouais. Additional support from the Foundation Sciences Mathématiques de Paris.

H. Arfaoui
South Mediterranean Univ., Tunisia

P. Fraigniaud
CNRS and Univ. Paris Diderot, France

D. Ilcinkas
CNRS and Univ. Bordeaux, France
Tel.: +33 540 006 912
E-mail: david.ilcinkas@labri.fr

F. Mathieu
Nokia Bell Labs France, France

A. Pelc
Univ.du Québec en Outaouais, Canada

1 Introduction

1.1 Context and objective

In the framework of distributed computing in large scale networks, faults of various kinds inherently occur. It is therefore of the utmost importance that the system be able to detect the presence of these faults when they occur, for taking appropriate actions aiming at fixing the errors. This paper focuses solely on the detection of faults, that is, on the design of mechanisms allowing the nodes in the network to check whether the system is in a legal state or not. For this purpose, we classically model networks as *labeled graphs*. A labeled graph is a pair (G, x) where $G = (V, E)$ is a connected simple graph, and $x : V \to \{0, 1\}^*$ is a function assigning a *label* $x(v)$ to every node v of G. For instance, these labels can be viewed as the outputs of previous computations, and we aim at checking the correctness of these outputs. A *distributed language* is a (Turing decidable) family $\mathcal{L}$ of labeled graphs. Typical examples of distributed

languages are

$$\mathsf{is\text{-}properly\text{-}colored} = \{(G,x) : \forall \{u,v\} \in E, \\ x(u) \neq x(v)\}$$

and

$$\mathsf{cycle\text{-}freeness} = \{(G,x) : G \text{ has no cycles}\}.$$

Note that in this latter case the language does not actually rely on the labels.

1.1.1 Distributed decision

The standard mechanism for distributed decision works as follows [32]. A *decision algorithm* for a distributed language $\mathcal{L}$ is a distributed algorithm performed at every node, allowing each node of any given graph G whose nodes are labeled by x to emit a verdict about whether $(G,x) \in \mathcal{L}$ or not, after having exchanged information with nodes in its vicinity (e.g., its t-neighborhood, i.e., all nodes at distance at most t in the graph, for some $t \geq 0$). This verdict is a boolean value, and, for the algorithm deciding $\mathcal{L}$ to be correct, it is required that, for every labeled graph (G,x),

$$(G,x) \in \mathcal{L} \iff \bigwedge_{v \in V} \mathsf{out}(v) = \text{true},$$

where $\mathsf{out}(v) \in \{\text{true}, \text{false}\}$ is the verdict outputted by node v of G. The logical conjunction operator applied to the individual outputs for defining the correctness condition is motivated by different aspects of distributed computing. In particular, a node outputting false in (G,x) can raise an alarm, or launch a recovery procedure aiming at moving the system back to a legal configuration, that is, aiming at computing another labeling function x' such that $(G,x') \in \mathcal{L}$.

Nevertheless, there are distributed settings in which other operators, different from the logical conjunction, may well be more appropriate. In these settings, even limiting the verdicts to be a single bit, true or false, is not mandatory. An example of such settings is *sensor networks* [25] in which the individual outputs of sensors or of motes are gathered at a gateaway node. Another example is *distributed runtime monitoring*, in which a collection of monitors observe the behavior of a system at run time, produce their individual verdicts, and transmit these verdicts to a central monitor (see [10] and the references therein). In any case, the central authority applies some boolean function f on the multi-set formed by the collection of individual verdicts, and, for the algorithm deciding $\mathcal{L}$ to be correct, it is required that, for every labeled graph (G,x),

$$(G,x) \in \mathcal{L} \iff f(\{\mathsf{out}(v), v \in V\}) = \text{true}. \quad (1)$$

Yet, in this context, it is desirable that decision algorithms produce individual verdicts on as few bits as possible, for at least two reasons. The first reason is conceptual, as the problem becomes trivial if one allows large verdicts. In particular, if every node has a unique identifier and is allowed to produce a verdict on $O(n \log n + k)$ bits in n-node graphs labeled by k-bit labels, then every node can send its identity, its label, and the list of its neighbors' identities to the central authority. This allows that central authority to reconstruct the entire instance (G,x), and to decide locally whether $(G,x) \in \mathcal{L}$ or not. (Recall that all distributed languages are supposed to be Turing decidable). More importantly, as far as practical applications might be concerned, the communication channel between the central authority and the system may be of limited capacity. This would typically be the case of a distributed system embedded onboard a satellite, or in any environment that is difficult to access, enforcing the central monitor to be remote. In this paper we investigate the power and limitations of such decision mechanisms where, for every node v, $|\mathsf{out}(v)| = O(1)$. That is, the verdict emitted by every node should be stored in a constant number of bits, independently of the network size (i.e., its number of nodes), and independently of the length of the labels (i.e., their number of bits). It follows that the number of bits transferred from the system to the central authority remains limited to $O(n)$ bits in n-node networks.

We believe that limiting the size of the output to a constant, rather than to $O(\log n)$, increases the applicability of our model for large sensor networks, in cases when transmitting output bits to the central authority is costly. This is particularly significant in the case when this constant can be made very small, e.g., 2, as we do for monitoring cycle-freeness.

1.1.2 Distributed verification

We also investigate distributed *verification* in the same framework. Distributed verification can be viewed as a non-deterministic version of distributed decision. In its abstract form, it assumes the presence of a *prover*, that is, an oracle assigning a certificate $y(v) \in \{0,1\}^*$ to every node, based on the current labeled graph (G,x). For every legal instance (G,x), these certificates should be forged to collectively form a global proof y that $(G,x) \in \mathcal{L}$. A *distributed verification* algorithm running at node v does not only take the label $x(v)$ of v as input, but also the certificate $y(v)$ assigned to v by the oracle. In the same spirit as for the class NP, for a verification algorithm to be correct, it is required that,

for every labeled graph (G, x),

$$(G,x) \in \mathcal{L} \iff \exists y : V \to \{0,1\}^* \ : \ \bigwedge_{v \in V} \mathsf{out}(v) = \text{true}. \quad (2)$$

In other words, if $(G, x) \notin \mathcal{L}$, then the verification algorithm cannot be fooled in the sense that, for every certificate assignment y, the labeled graph (G, x) will be rejected by at least one node.

Note that this form of non-deterministic decision mechanism is of practical relevance. In particular, an algorithm in charge of computing a labeling function x for every graph G such that $(G, x) \in \mathcal{L}$ may as well simultaneously compute a certificate function y enabling to easily verify $(G, x) \in \mathcal{L}$. In fact, this mechanism is frequently used for the design of self-stabilizing algorithms [1,6,12,28].

In this paper, we generalize this approach to the context in which, for a verification algorithm to be correct, it is required that, for every labeled graph (G, x),

$$(G,x) \in \mathcal{L} \iff \exists y : V \to \{0,1\}^* \ : \ f(\{\mathsf{out}(v), v \in V\}) = \text{true}. \quad (3)$$

As far as practical applications to fault-tolerance are concerned, this mechanism would simply require that the central authority inform one or few nodes of the presence of an error in case f would return false on the multiset $\{\mathsf{out}(v), v \in V\}$ of outputs.

1.1.3 Complexity classes

Two important complexity classes have been defined and analyzed in [21]. For every integer $t \geq 0$, let $\mathrm{LD}(t)$ (respectively, $\mathrm{NLD}(t)$) be the set of all distributed languages that can be decided (respectively, verified) by having each node inspecting its t-neighborhood only, and let

$$\mathrm{LD} = \cup_{t \geq 0} \mathrm{LD}(t), \text{ and } \mathrm{NLD} = \cup_{t \geq 0} \mathrm{NLD}(t).$$

For instance, $\mathsf{is\text{-}properly\text{-}colored} \in \mathrm{LD}$ as it is sufficient that every node consults the colors of its neighbors to check whether the graph is properly colored. Also, $\mathsf{cycle\text{-}freeness} \in \mathrm{NLD}$, as follows. In a cycle-free graph, the prover selects a node as the root, and assigns to every node a certificate equal to its distance to the root. Every node with given certificate $d > 0$, checks that it has exactly one neighbor with certificate $d - 1$, and all other neighbors with certificate $d + 1$. A node with certificate $d = 0$ checks that all its neighbors have certificate $d = 1$. At every node, if these tests are passed, then the node outputs true, otherwise it outputs false. One can easily check that there is no way of fooling this verification mechanism by assigning fake certificates to the nodes of a graph containing a cycle. More generally, it has been shown that NLD coincides with the set of distributed languages closed under lift (see Lemma 1), i.e., the set of languages $\mathcal{L}$ such that if $(G, x) \in \mathcal{L}$ then, for every lift (G', x') of (G, x), we have $(G', x') \in \mathcal{L}$. We refer to Section 2 for the formal definition of lifts, a.k.a., coverings, but, in essence, a configuration (G', x') is a lift of a configuration (G, x) if there is an input-preserving map $\varphi : V(G') \to V(G)$ that induces a local isomorphism between the neighborhood of every vertex $v' \in V(G')$ and the neighborhood of $\varphi(v') \in V(G)$. In other words, (G, x) and (G', x') "look the same" locally.

By relaxing the decision rule, from the logical conjunction of boolean outputs to any function applied to outputs on $O(1)$ bits, as specified in Eq. (1) and (3), we define $\mathrm{XLD}(t)$ and $\mathrm{XNLD}(t)$ similarly to $\mathrm{LD}(t)$ and $\mathrm{NLD}(t)$, respectively, and

$$\mathrm{XLD} = \cup_{t \geq 0} \mathrm{XLD}(t), \text{ and } \mathrm{XNLD} = \cup_{t \geq 0} \mathrm{XNLD}(t),$$

where "X" stands for "extended". By definition, we have $\mathrm{LD} \subseteq \mathrm{XLD}$, and $\mathrm{NLD} \subseteq \mathrm{XNLD}$. Our objective is to identify the power and limitations of our extended form of distributed decision and verification. That is, we address questions like: Is XLD much larger than LD? Is XNLD much larger than NLD? How do XLD and NLD differ? Etc.

1.2 Our results

We first show that the extended classes XLD and XNLD are significantly larger than their respective base classes LD and NLD. In particular, we prove that:

- XLD contains distributed languages that are not even in NLD, and
- XNLD contains *all* distributed languages.

In general, we prove a collection of separation results, summarized in Figure 1.

In the second part of the paper, we turn our attention to the extensively studied problem of checking whether a given graph is cycle-free. (Recall the promise that all considered graphs are connected). It is known [12,24,28] that

$$\mathsf{cycle\text{-}freeness} \in \mathrm{NLD} \setminus \mathrm{LD}$$

with certificates of non-constant size, $\Theta(\log n)$ bits in n-node networks. This result holds even in graphs of bounded degree. Regarding deciding cycle-freeness, it still holds that

$$\mathsf{cycle\text{-}freeness} \notin \mathrm{XLD}.$$

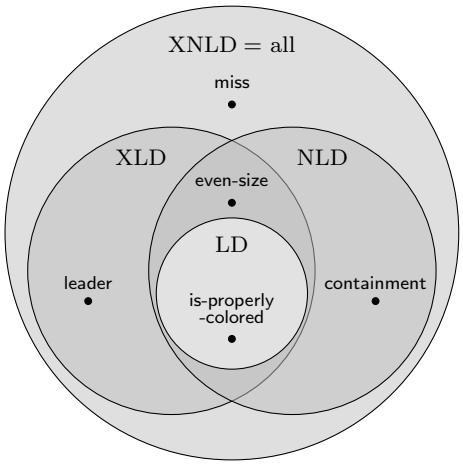


Fig. 1: *Four distributed decision and verification classes, with representatives*

However, as opposed to classical distributed decision with the logical conjunction operator, cycle-freeness $\in$ XLD whenever restricted to graphs of bounded degree. More precisely, we prove:

– cycle-freeness $\in$ XLD in (connected) graphs with maximum degree d requires to output at least $\log d - 1$ bits at some node.

That is, the trivial zero-round algorithm consisting of (1) asking every node to output its degree, and (2) checking whether the sum of these degrees is equal to twice the number of nodes minus one is essentially optimal. Also, we show that

– cycle-freeness $\in$ XNLD with certificates of size $O(1)$ bits, and 2-bit output per node.

This is in contrast with the $\Omega(\log n)$-bit certificates required to place cycle-freeness in NLD. These results are summarized in Table 1.

The paper is structured as follows. In Section 2, we present the model and some already known results that we will use in the paper. Section 3 proves all the inclusion and separation results summarized in Table 1. Section 4 shows that the size of the certificates to prove that a language is in XNLD may be large. The next two sections are devoted to the cycle-freeness problem, with Section 5 focusing on verification while Section 6 concerns deciding cycle-freeness.

1.3 Used techniques

In terms of techniques, several of our separation results regarding the classes XLD, NLD, and XNLD use reduction to communication-complexity problems. Establishing that deciding cycle-freeness implies to output $\lceil \log d \rceil - 1$ bits at some node requires to combine several techniques. First, we show that one can reduce our concern to *order-invariant* algorithms, that is, roughly, to algorithms whose output at a node does not depend on the actual *value* of the identities of the nodes in its vicinity, but solely on the *relative order* of these values. The celebrated result by Naor and Stockmeyer [32] enabling to reduce the study of certain kinds of algorithms to order-invariant algorithms cannot be applied in our context because our instances are not necessarily in the class LCL of so-called *locally checkable languages*. Nevertheless, we were able to provide a novel reduction, that does not require LCL membership, by using the *infinite version* of Ramsey's Theorem. Our second main technique is the construction, for every order-invariant algorithm supposed to decide cycle-freeness with too small output at each node, of two explicit instances, one with a cycle, and one without, that cannot be distinguished by the algorithm. This construction is difficult because cycle-freeness can be distributedly decided with just 2-bit output per node in subdivided graphs[1]. Nevertheless, by an appropriate use of the known fact that every free group can be linearly ordered, we were able to construct legal and illegal instances that cannot be distinguished locally by the assumed order-invariant distributed algorithm.

1.4 Related work

There has been an enormous amount of work on distributed decision and verification during the last few years. In this section, we shall mention the most relevant results, and we refer to the survey [16] for a more complete account.

Distributed decision was first mentioned in [32] where the class of locally checkable labelings (LCL) is introduced, as a crucial tool for analyzing local algorithms, including the ability to derandomize algorithms for LCL tasks running in a constant number of rounds. Local decision was later revisited in [21], where the classes LD and NLD are defined and analyzed. The same paper also defines the class BPLD, the probabilistic version of LD. Interestingly, [15] proved that the same derandomization result as in [32] also holds for the class BPLD (which strictly includes LCL), whenever we

[1] A subdivided graph [2] is a graph in which no two nodes of degree different from 2 are adjacent. In such graphs with $n \geq 3$, the following algorithm works, with only four different kinds of outputs (i.e., 2-bit outputs): a node with degree $\neq 2$ outputs 0, and a node with degree 2 outputs 2, 3 or 4 depending on whether it is adjacent to 0, 1 or 2 nodes with degree $\neq 2$, respectively. The central authority then accepts if and only if the sum of the outputs equals $2(n-1)$.

	output size (#bits)	comments
distributed decision [folklore]	1	impossible with conjunction
extended distributed decision [this paper]	$\log d \pm \Theta(1)$	sum of degrees is optimal
distributed verification [12,24,28]	1	$\Theta(\log n)$-bit certificates
extended distributed verification [this paper]	2	$O(1)$-bit certificates

Table 1: *Monitoring cycle-freeness in n-node max-degree-d (connected) graphs*

restrict ourselves to graphs with degrees upper bounded by a constant, and labels of constant size. The impact of identifiers in local decision was analyzed in [17,19], which demonstrate that, perhaps surprisingly, identifiers help local decision even if such tasks do not require symmetry breaking. Randomized distributed decision was extensively studied in [20], where it is shown that, as opposed to the sequential setting, the probability of success for the randomized decision algorithm cannot be boosted. Note another important difference between sequential and distributed decision: checking the optimality of a given solution may be more difficult than constructing an optimal solution (see., e.g., [11]). Distributed property testing is a recent variant of randomized distributed decision (see [14] and the references therein). In this framework, the algorithm must decide if the current instance is legal or "far from" being legal, which enables to decide more properties in constant time.

Different mechanisms for distributed verification, where certificates are provided to the nodes by an oracle, were presented in the context of fault-tolerant computing, including self-stabilization [1,12,24,26]. Distributed verification per se was explicitly studied in [28], which introduced proof-labeling schemes (PLS). These schemes were generalized in [23] by considering locally checkable proofs (LCP). The difference between PLS and LCP is twofold: first, in proof-labeling schemes, the verification algorithms should run in a single round, while locally checkable proofs allow verification algorithms running in an arbitrary (constant) number of rounds; second, proof-labeling schemes exchange only certificates between nodes, while locally checkable proofs allow the nodes to exchange their whole states, including their certificates. In both cases, the certificates can be assigned by the prover, after the nodes have been provided with their identities. That is, the certificate function y in Eq. (2) can depend on the identities assigned to the nodes. This property has an important consequence: all distributed languages belong to both PLS and LCP, with certificates on $O(n^2+kn)$ bits in n-node graphs with k-bit labels [28]. This certificate size is optimal, in the sense that there are distributed languages requiring proofs with certificates on $\Omega(n^2+kn)$ bits [23]. In contrast, the aforementioned class NLD defined in [21] requires that the certificate function y do not depend on the identity assignment to the nodes, but only on the given labeled graph (G,x). It follows that there are distributed languages outside NLD. In fact, NLD has been characterized in [18] as the set of distributed languages closed under lift (see Lemma 1 in the next section). On the other hand, NLD can be viewed as Σ_1^{LD} in a "local hierarchy" mimicking the polynomial hierarchy in the local distributed decision setting[2]. In fact, [7] recently showed that this hierarchy collapses at the second level, as $\Pi_2^{\text{LD}} = \text{all}$. To sum up, it is known that

$$\text{PLS} = \text{LCP} = \Pi_2^{\text{LD}} = \text{all},$$

while

$$\text{NLD} \stackrel{\text{def}}{=} \Sigma_1^{\text{LD}} = \{\mathcal{L} \text{ closed under lift}\},$$

and, in particular, we shall show in this paper that

$$\text{XNLD} \stackrel{\text{def}}{=} \Sigma_1^{\text{XLD}} = \text{all}.$$

It is worth noticing that our model bears similarities with the shared whiteboard model, as studied in [8,9]. In this model, all nodes share a whiteboard with read/write accesses to it, and each outputs according to its state, and to the content of the whiteboard. However, the network is *not* the communication medium, and all communications occur via the shared whiteboard. As a consequence, even if the communications between the nodes and the whiteboard can potentially be somehow limited (e.g., to $O(\log n)$ bits at each round), every node become aware of information from nodes far away in the network. In contrast, we consider here the LOCAL model (see below), in which nodes have only access to information from their vicinity.

Finally, we are stressing the fact that a new direction of research related to the theory of distributed decision has been investigated recently, namely *distributed interactive proofs* [27]. In this context, a centralized

[2] We have adopted the notation Σ_1^{LD}, Π_2^{LD}, etc., to mimic the standard notation for the existential/universal definition of the polynomial hierarchy, where $P = \Sigma_0^{\text{P}} = \Pi_0^{\text{P}}$, $NP = \Sigma_1^{\text{P}}$, etc. The P in exponent in the polynomial hierarchy refers to *polynomial time*, while LD in exponent refers to *local decision*, and XLD to *extended local decision*.

prover (Merlin) is interacting with a distributed randomized verifier (Arthur) for a finite number of times, before a local verification phase is performed among the nodes. In the paper introducing the model [27], as well as in the follow up contributions [22,31], the power of such interactive protocols is investigated regarding deciding classical problems (symmetry, isomorphism, diameter, etc.), as well as comparing the complexity of interactive proofs with the one of locally checkable proofs.

2 The model

We consider networks modeled as simple *connected* graphs, whose vertices model the computing nodes, and edges model the communication links. The computational model considered in this paper is the classic LOCAL model [34], which is a standard distributed computing model capturing the essence of locality. In this model, each node has an identity that is unique in the network, i.e., distinct from all other identities in the network. The identity of a node v is a non negative integer, denoted by $\mathsf{id}(v) \in \mathbb{N}$. All nodes are woken up simultaneously, and execute the same algorithm. Computation proceeds in fault-free synchronous *rounds* during which every node

1. sends a message of unlimited size to each of its neighbors in the underlying network,
2. receives the messages sent by its neighbors, and
3. performs arbitrary individual computations on its data.

The running time of an algorithm is defined as the maximum number of rounds it takes to terminate at all nodes, over all possible networks, and all possible identity assignments to the nodes in these networks. Similarly to [32], we are interested in algorithms whose running times are independent of the size of the network, and independent of the size of the identities. That is, we are focusing on algorithms that run in a constant number of rounds.

Besides its identity $\mathsf{id}(v)$, every node v has a label $x(v) \in \{0,1\}^*$, and we are interested in deciding whether a given pair (G,x) satisfies some given boolean predicate, that is, whether (G,x) belongs to some given distributed language $\mathcal{L}$. Here, the pair (G,x) denotes the graph G whose every node v is labeled by the binary string $x(v)$. Let $\mathsf{out}_{\mathcal{A}}(G,x,\mathsf{id},v)$ denote the output of node $v \in V(G)$ running Algorithm $\mathcal{A}$ in the labeled graph (G,x) with identity assignment id. We denote by $\mathsf{out}_{\mathcal{A}}(G,x,\mathsf{id})$ the global output, that is,

$$\mathsf{out}_{\mathcal{A}}(G,x,\mathsf{id}) = \{\mathsf{out}_{\mathcal{A}}(G,x,\mathsf{id},v), v \in V(G)\}$$

is the *multiset* of all individual outputs (the same individual output may appear more than once in $\mathsf{out}_{\mathcal{A}}(G,x,\mathsf{id})$).

Definition 1 Let f be a boolean function taking as input any multi-set with elements taken from $\{0,1\}^b$, for $b \geq 1$. Algorithm $\mathcal{A}$ *decides* the distributed language $\mathcal{L}$, with b-bit outputs, and using interpretation f if, for every n-node labeled graph (G,x), and for every identity assignment id to the nodes of G, we have $\mathsf{out}_{\mathcal{A}}(G,x,\mathsf{id},v) \in \{0,1\}^b$ for every node $v \in V(G)$, and

$$(G,x) \in \mathcal{L} \iff f(\mathsf{out}_{\mathcal{A}}(G,x,\mathsf{id})) = \text{true}.$$

We denote by XLD (for extended local decision) the class of distributed languages for which there exists a constant time decision algorithm, that is, for which there exist $t \geq 0$, a boolean function f taking as input any multi-set with elements taken from $\{0,1\}^b$ for some $b \geq 1$, and a t-round algorithm $\mathcal{A}$ that decides $\mathcal{L}$, with b-bit outputs, and using interpretation f.

Definition 2 Let f be a boolean function taking as input any multi-set with elements taken from $\{0,1\}^b$, for some fixed $b \geq 1$. Algorithm $\mathcal{A}$ *verifies* the distributed language $\mathcal{L}$, with b-bit outputs, and using interpretation f if, for every n-node labeled graph (G,x), we have

$$\begin{cases} (G,x) \in \mathcal{L} \Longrightarrow \exists y : V(G) \to \{0,1\}^*, \forall \mathsf{id} : V(G) \to \mathbb{N}, \\ \qquad\qquad f(\mathsf{out}_{\mathcal{A}}(G,x,y,\mathsf{id})) = \text{true} \\ (G,x) \notin \mathcal{L} \Longrightarrow \forall y : V(G) \to \{0,1\}^*, \forall \mathsf{id} : V(G) \to \mathbb{N}, \\ \qquad\qquad f(\mathsf{out}_{\mathcal{A}}(G,x,y,\mathsf{id})) = \text{false} \end{cases}$$

where $\mathsf{out}_{\mathcal{A}}(G,x,y,\mathsf{id},v) \in \{0,1\}^b$ for every node $v \in V(G)$.

There are significant differences between distributed decision and distributed verification. The function $y : V(G) \to \{0,1\}^*$ in Definition 2 is called a *certificate function*. It assigns to every node a *certificate* $y(v)$. Of course, the certificates depend on the labeled graph (G,x), and on the considered distributed language $\mathcal{L}$. Alternatively, one can view the certificates as assigned by a powerful *prover*, which aims at helping the algorithm (which can be viewed as a verifier) to check whether $(G,x) \in \mathcal{L}$. Note that the algorithm in Definition 1 takes only the identity $\mathsf{id}(v)$ and the label $x(v)$ as input at every node v. Instead, the algorithm in Definition 2 takes also the certificate $y(v)$ as input at node v. Finally, note that the certificate function y depends on the given labeled graph (G,x), but *not* on the identity assignment to the nodes.

We denote by XNLD (for extended nondeterministic local decision) the class of distributed languages for which there exists a constant time verification algorithm, that is, for which there exist $t \geq 0$,

a boolean function f taking as input any multi-set with elements taken from $\{0,1\}^b$ for some $b \geq 1$, and a t-round algorithm $\mathcal{A}$ that verifies $\mathcal{L}$, with b-bit outputs, and using interpretation f.

Finally, we use the notion of t-lifts, sometimes also called t-coverings (or simply coverings when $t = 1$), see [3,30]. For $t \geq 1$, a t-lift of a labeled graph (G, x) is a labeled graph (G', x') such that there exists a graph homomorphism $\varphi : V(G') \to V(G)$ with $x'(v) = x(\varphi(v))$ for every $v \in V(G')$, and φ induces a graph isomorphism between t-neighborhoods, i.e., between $B_{G'}(v,t)$ and $B_G(\varphi(v),t)$ for every node v of G', where $B_{G'}(v,t)$ is the open ball of radius t around v in G', that is, the graph defined as the union of all paths of length at most t starting from v in G'. The t-lift is called strict if (G', x') is not isomorphic to (G, x).

Example: Let us consider the n-node cycles $C_n = (v_0, \ldots, v_{n-1})$, for different values of n. The labeled graph (C_8, x) with $x(v_i) = i \bmod 4$ is a 1-lift of the labeled graph (C_4, x). The lift is $\varphi : V(C_8) \to V(C_4)$ defined by $\phi(v_i) = v_{i \bmod 4}$. Similarly, (C_{16}, x) is a 2-lift of (C_8, x). By contrast, (C_8, x) is not a 2-lift of (C_4, x) because any ball of radius 2 in (C_8, x) includes five different nodes, while any ball of radius 2 in (C_4, x) includes four nodes.

The main property of t-lifts that we will use in the paper is the following folklore observation.

Fact 1 *For any t-lift (G', x') of (G, x) by $\varphi : V(G') \to V(G)$, the size $|\varphi^{-1}(v)|$ of the pre-image $\varphi^{-1}(v)$ of v is the same for any node $v \in V(G)$ (and this size is larger than 1 in the case of a strict t-lift).*

Proof If two nodes v and w of G satisfy $|\varphi^{-1}(v)| > |\varphi^{-1}(w)|$, then this also holds for two adjacent nodes v and w. Thus, since φ induces an isomorphism between balls of radius t, it induces an isomorphism between $B_{G'}(v', t)$ and $B_G(v, t)$, for every $v' \in \varphi^{-1}(v)$. Let $\varphi^{-1}(v) = \{v'_1, \ldots, v'_k\}$. To the edge $\{v, w\}$ in G corresponds a collection $\{v'_i, w'_i\}$, $i = 1, \ldots, k$, of edges in G' where w'_i is the image of w by the isomorphism φ^{-1} between $B_G(v, t)$ and $B_{G'}(v'_i, t)$. Since $|\varphi^{-1}(w)| < k$, it must be the case that $w'_i = w'_j$ for some $i \neq j$. This contradicts the fact that φ induces an isomorphism between $B_{G'}(w'_i, t)$ and $B_G(w, t)$, as the two neighbors v'_i and v'_j of w'_j are mapped to the same node v. This contradiction establishes Fact 1. □

The following result will also be used throughout the paper.

Lemma 1 (Lemmas 1 and 2 in [18])
Let $\mathcal{L}$ be a distributed language. $\mathcal{L} \in$ NLD if and only if there exists $t \geq 1$ such that $\mathcal{L}$ is closed under t-lift.

3 Classification and separation

Recall that the classes LD and NLD, defined in [21], are the respective restrictions of XLD and XNLD to the setting in which each node can output a single bit, and the interpretation is the result of the conjunction operator on these outputs. Hence, by definition, LD $\subseteq$ XLD, and NLD $\subseteq$ XNLD. Also, by definition, XLD $\subseteq$ XNLD. The purpose of this section is to show that these inclusions are strict (the strict inclusion LD $\subset$ NLD is established in [21]), and to study the relationship between XLD and NLD. The following result is illustrated in Figure 1.

Theorem 1 $\mathrm{XLD} \setminus \mathrm{NLD} \neq \emptyset$, $\mathrm{NLD} \setminus \mathrm{XLD} \neq \emptyset$, *and*

$$\mathrm{LD} \subset (\mathrm{XLD} \cap \mathrm{NLD}) \subset (\mathrm{XLD} \cup \mathrm{NLD}) \subset \mathrm{XNLD} = \mathrm{All}.$$

The proof of the above theorem is direct by combining the following five lemmas, including Lemma 4 which, in addition, provides an upper bound on the size of the certificates enabling to place every language in XNLD. As it will become clear from the proof of these lemmas, the extra power of XNLD compared to NLD allows us to distinguish whether a configuration (G', x') is isomorphic to a configuration (G, x), or is a strict 1-lift of that configuration.

Lemma 2 $\mathrm{XLD} \setminus \mathrm{NLD} \neq \emptyset$.

Proof To establish $\mathrm{XLD} \setminus \mathrm{NLD} \neq \emptyset$, let

$$\mathsf{leader} = \{(G, x) : \forall u \in V(G), x(u) \in \{0,1\}, \text{ and } \sum_{u \in V(G)} x(u) = 1\},$$

i.e., the set of labeled graphs (G, x) such that $x(u) \in \{0, 1\}$ equals 0 for all nodes except exactly one (the leader). We have $\mathsf{leader} \notin$ NLD because this language is not closed under lift (cf. Lemma 1). Indeed, no strict lift of an element of leader is in leader itself (by Fact 1, a legal configuration with one leader is lifted to a configuration with at least two leaders, which is illegal). To establish that $\mathsf{leader} \in$ XLD, we describe a local distributed algorithm enabling each node to output a constant number of bits, with the associated interpretation. The algorithm runs in zero rounds: every node u simply returns the single bit $b_u = x(u)$. The decision is then made according to the collection $\{b_i \in \{0,1\}, i \in [n]\}$ of outputs[3], by applying the logical operator

$$f(\{b_i \in \{0,1\}, i \in [n]\}) = \bigvee_{i=1}^{n} \left(b_i \wedge \bigwedge_{j \neq i} \overline{b_j} \right)$$

[3] The indices $i = 1, \ldots, n$ are only for the purpose of notation. The decision is taken based on an unordered multiset of outputs.

which is true if and only if there is a unique b_i equal to 1. Hence, the input configuration is accepted if and only if there is a unique node u with $x(u) = 1$, as desired. This proves that XLD $\setminus$ NLD $\neq \emptyset$. □

Lemma 3 LD $\subset$ XLD $\cap$ NLD.

Proof We have LD $\subseteq$ XLD $\cap$ NLD by definition. To establish the strict inclusion LD $\subset$ XLD $\cap$ NLD, let

$$\mathsf{even\text{-}size} = \{(G, x) : G \text{ has an even number of nodes}\}.$$

This language is in NLD because it is closed under lift (cf. Lemma 1). Indeed, by Fact 1, the number of nodes of a lift is a multiple of the number of nodes of the lifted graph. To show that it also belongs to XLD, consider the algorithm running in zero rounds consisting, for every node u, of outputting the single bit $b_u = 1$. The decision is then made by applying the operator

$$f(\{b_i \in \{0,1\}, i \in [n]\}) = 1 - \bigoplus_{i=1}^{n} b_i$$

to the multi-set $\{b_i \in \{0,1\}, i \in [n]\}$ of output bits, where $\oplus$ denotes the exclusive-disjunctive (XOR) operator. The value of f is equal to 1 if and only if the graph has an even number of nodes. Finally, we have $\mathsf{even\text{-}size} \notin$ LD. This is because if some node u outputs 0 in an odd cycle C with some identity assignment (there must be such a node so that C be rejected by the conjunction operator), then it also outputs 0 in some even cycle, causing this latter legal instance to be wrongly rejected. (Take the same cycle C with the same identity assignment, and insert one node between the two nodes at distance $\lfloor n/2 \rfloor$ from u, with some arbitrary identity distinct from the existing ones: node u still outputs 0 in this cycle).

This completes the proof of LD $\subset$ XLD $\cap$ NLD. □

Lemma 4 *Every (Turing decidable) distributed language is in* XNLD. *Moreover, the verification of languages on n-node networks with k-bit labels per node can be achieved using $O(n^2 + kn)$-bit certificates, by having each node inspecting its* 1*-hop neighborhood with 2-bit outputs, or in* 2 *rounds with* 1*-bit outputs.*

Proof Let $\mathcal{L}$ be a language. We describe a 1-round verification protocol for $\mathcal{L}$. The certificate y of an instance $(G, x) \in \mathcal{L}$ is an $n \times n$ adjacency matrix M of G, with nodes indexed arbitrarily by distinct integers in $[1, n]$, plus an n-dimensional array L where L_i is the label of node $i \in [1, n]$. In addition, every node v receives the index $\lambda(v) \in [1, n]$ corresponding to v in M and L. More formally, the certificate at node v is

$$y(v) = ((G', x'), i),$$

where G' is an isomorphic copy of G with nodes labeled by λ from 1 to n, x' is an n-dimensional vector such that $x'(\lambda(u)) = x(u)$ for every node u, and $i = \lambda(v)$. In n-node networks with k-bit input per node, such a certificate is on $O(n^2 + kn)$ bits.

Let us first describe an algorithm using two bits a_u and b_u of output at every node u, and then we will show how to reduce these two bits to just one bit, at the cost of an extra round of communication. Every node u with index $\lambda(u) = 1$ sets $a_u = 1$. All other nodes set $a_u = 0$. For computing b_u, every node u performs a single round of communication, sending its input $x(u)$ and its certificate $y(u)$ to all its neighbors. Each node then uses its input, its certificate, and the received information from its neighbors to check that (1) they all got the same graph G' and the same label x' in their certificates, and (2) its actual neighborhood corresponds to its neighborhood in the certificate (isomorphism of the neighborhoods preserving the input x and the label λ). If some inconsistency is detected by a node u, then this node sets $b_u = 0$. At this point, each node u that has not yet set the variable b_u sets it to 1 if $(G', x') \in \mathcal{L}$, and to 0 otherwise. (Recall that a distributed language is, by definition, Turing decidable). Every node u outputs the pair (a_u, b_u). The decision is then made according to the multiset $\{(a_i, b_i) \in \{0,1\}^2, i \in [n]\}$ of outputs, by applying

$$f(\{(a_i, b_i) \in \{0,1\}^2, i \in [n]\}) = \left(\bigvee_{i=1}^{n} \left(a_i \wedge \bigwedge_{j \neq i} \overline{a_j}\right)\right) \wedge \left(\bigwedge_{i=1}^{n} b_i\right).$$

To see why $f = 1$ if and only if $(G, x) \in \mathcal{L}$, observe that if every node u passes the tests regarding the certificates, without setting b_u to 0, then all nodes agree on the graph G' and on the input vector x', and $(G', x') \in \mathcal{L}$. Moreover, they know that their respective neighborhood in G fits with the corresponding one in G'. Therefore, if every node u passes the tests regarding the certificates without setting b_u to 0, then (G, x) is either identical (up to isomorphism) to (G', x') or to a strict 1-lift of it[4]. It follows that, if all bits b_u are 1, then $(G', x') = (G, x)$ if and only if there exists exactly one node $v \in G$, with index $\lambda(v) = 1$. This is precisely the leader problem, which is decided using the outputs in $\{a_u, u \in V(G)\}$.

Now, we reduce the two bits a_u and b_u to just one bit c_u. This is done using an extra round of communication, in which each node u sends its value b_u to all its neighbors. If any received bit is 0, then node u sets b_u

[4] Recall that a graph H is a 1-lift of a graph G if there exists a homomorphism from H to G preserving the neighborhood of each node.

to 0. Otherwise, the bit b_u is kept unchanged. Observe that now, thanks to this extra round of communication, if any node u detects some inconsistencies in the first round, then all its neighbors are also aware of the issue. As a consequence, if some node "raises an alarm" (i.e., sets $b_u = 0$), then at least another node does the same. Thus, every node u sets

$$c_u = a_u \vee \overline{b_u},$$

and outputs c_u. The decision is then made according to the collection $\{c_i \in \{0,1\}, i \in [n]\}$ of outputs, by applying the operator

$$f(\{c_i \in \{0,1\}, i \in [n]\}) = \bigvee_{i=1}^{n} \left(c_i \wedge \bigwedge_{j \neq i} \overline{c_j}\right)$$

which is 1 if and only if $(G,x) \in \mathcal{L}$. Indeed, $c_u = 1$ if and only if u detects some issue (i.e., $b_u = 0$) or $\lambda(u) = 1$ (i.e., $a_u = 1$). However, if u has detected some issue, then one of its neighbors u' has also detected some issue, which guarantees $c_{u'} = 1$ for u' as well. Thus, $f = 0$ if $(G,x) \notin \mathcal{L}$. (The case where G is reduced to a single node is an exception: in this case, the unique node u sets $c_u = a_u \wedge b_u$). This completes the proof that XNLD = All. □

In the following, we use a classical result from communication complexity. In this field introduced by Yao [36], two players called Alice and Bob each receives an n-bit string, respectively α and β. They must compute the value $f(\alpha,\beta)$, where f is some given boolean function, by communicating one bit at a time to each other. The communication complexity of a function f is defined as the minimum number of bits exchanged by Alice and Bob in the worst case in order to compute the function f. The next two lemmas use the following classical lower bound.

Theorem 2 ([36], see also [29]) *The communication complexity of the identity function* EQUALITY *on n-bit strings is at least n.*

Lemma 5 $\mathrm{XLD} \cup \mathrm{NLD} \subset \mathrm{XNLD}$.

Proof We use a language, called miss, introduced in [7], and we show that $\mathsf{miss} \notin \mathrm{XLD} \cup \mathrm{NLD}$. In miss, every node u of a labeled graph (G,x) is given a family $\mathcal{F}(u)$ of labeled graphs, each described by an adjacency matrix representing the graph, and a 1-dimensional array representing the labeling of the nodes of that graph. In addition, every node u of (G,x) is given a binary string $x'(u) \in \{0,1\}^*$. Hence, (G,x') is also a labeled graph. The actual configuration (G,x) is legal if and only if (G,x') is missing in all families $\mathcal{F}(u)$ for every $u \in V(G)$, i.e., $(G,x') \notin \mathcal{F}$ where $\mathcal{F} = \cup_{u \in V(G)} \mathcal{F}(u)$. In short, we consider the language

$$\mathsf{miss} = \{(G,x) : \forall u \in V(G), x(u) = (\mathcal{F}(u), x'(u)) \text{ and } (G,x') \notin \mathcal{F} = \cup_{u \in V(G)} \mathcal{F}(u)\}.$$

Note that $\mathsf{miss} \notin \mathrm{NLD}$ because it is not closed under lift (as it may be the case that $(G,x') \notin \mathcal{F}$ but a lift of (G,x') is in $\mathcal{F}$), cf. Lemma 1.

We prove that $\mathsf{miss} \notin \mathrm{XLD}$ by contradiction, using arguments from communication complexity. Assume that there exists a t-round algorithm $\mathcal{A}$ and an interpretation f of individual b-bit outputs produced by $\mathcal{A}$ enabling to decide miss. In particular, the combination of $\mathcal{A}$ with f must decide the restricted version of miss, defined on paths $P = (v_1, \ldots, v_n)$, defined as follows. Let $k > (t+1)b$. We set $x'(v_1) = \alpha \in \{0,1\}^k$, $x'(v_n) = \beta \in \{0,1\}^k$, and, for every node v_i, $1 < i < n$, we set $x'(v_i)$ as k consecutive zeros. For every node v_i, $1 \leq i \leq n$, we set

$$\mathcal{F}(v_i) = \{(G,x) : G = P \text{ and } x(v_j) \in \{0,1\}^k \text{ with } x(v_1) \neq x(v_n)\}.$$

From this setting, it follows that

$$(P, (\mathcal{F}, x')) \in \mathsf{miss} \iff \alpha = \beta.$$

We show that, using $\mathcal{A}$ and f, one could solve the communication complexity problem EQUALITY between Alice and Bob, by exchanging fewer than k bits, which contradicts the well known lower bound k from Theorem 2. Given α as input, Alice simulates the algorithm $\mathcal{A}$ applied at the $n-t-1$ nodes $v_1, \ldots, v_{n-t-1}$, while, given β as input, Bob simulates $\mathcal{A}$ applied to the $t+1$ nodes $v_{n-t}, \ldots, v_n$. Since $\mathcal{A}$ produces b bits of output at each node, the simulation of $\mathcal{A}$ allows Alice to compute $(n-t-1)b$ bits, i.e., the $n-t-1$ outputs of the nodes $v_1, \ldots, v_{n-t-1}$. Similarly, Bob computes $(t+1)b$ bits. It is thus sufficient for Bob to send these $(t+1)b$ bits to Alice so that she can apply f on these bits together with her own $(n-t+1)b$ bits to determine whether $\alpha = \beta$ or not. This holds for any $\alpha, \beta \in \{0,1\}^k$. This is a contradiction, whenever $k > (t+1)b$. Hence $\mathsf{miss} \notin \mathrm{XLD}$, which completes the proof. □

Lemma 6 $\mathrm{NLD} \setminus \mathrm{XLD} \neq \emptyset$.

Proof Let us consider the following language defined in [21]. Every node v is given an element $x(v)$ of a ground set U, and a collection $\mathcal{S}(v)$ of subsets of U.

$$\mathsf{containment} = \{(G,(x,\mathcal{S})) : \exists v \in V(G),\ \exists S \in \mathcal{S}(v) \text{ with } \{x(u) : u \in V(G)\} \subseteq S\}.$$

We have containment $\in$ NLD because it is closed under lift (the set of node labels is preserved under lift). Now, by the same arguments as for proving miss $\notin$ XLD in Lemma 5, one can show containment $\notin$ XLD as well.

More precisely, we also focus on the paths $P = (v_1, \ldots, v_n)$ where the only differences between the instances are restricted to the input labels of the two extremities. The internal nodes of the paths receive the same $x = x_0$, where x_0 is some fixed element of the ground set U. For the extremities, we set $x(v_1) = \alpha$ and $x(v_n) = \beta$, where α and β are not necessarily distinct elements of U that are nevertheless both different from x_0. Finally, for each node v of the path (including its extremities), we set $\mathcal{S}(v)$ as the set of all unordered pairs of elements of U. Similarly as in the proof of Lemma 5, we have that $(P, (x, \mathcal{S})) \in$ containment $\iff \alpha = \beta$. By choosing a set U of size $2^k + 1$, with $k > (t+1)b$, the same communication complexity arguments lead to a contradiction, proving the lemma. □

Remark. Lemma 4 states that all distributed languages are verifiable using certificates of $O(n^2 + kn)$ bits, which is the same upper bound as for proof-labeling schemes [28]. However, while proof-labeling schemes allow certificates to depend on the identity assignment, our verification algorithm uses certificates that are independent of the identity assignment. On the other hand though, our verification algorithm is not restricted to use the conjunction operator for interpreting the multiset of local outputs.

4 Minimum certificate size for universal verification

By Lemma 4, we know that every TM-decidable distributed language with k-bit inputs is locally verifiable by providing nodes with certificates of $O(n^2 + kn)$ bits in n-node networks. Moreover, the verification is performed in one round with 2-bit outputs, or in two rounds with 1-bit outputs. The following theorem proves that this bound is tight, in the sense that, for every integers $k, t, b \geq 1$, there exist languages with k-bit inputs which require certificates of size $\Omega(n^2 + nk)$ bits to be verified in t rounds with b-bit outputs at each node.

Theorem 3 *For any integers $k \geq 0$, $t \geq 0$, and $b \geq 1$, there exist languages with k-bit labels that require certificates of size $\Omega(n^2 + nk)$ bits in n-node networks to be verified locally in t rounds using b-bit outputs (i.e., to be placed in* XNLD(t)*).*

Proof We define the language symmetry as follows. Given a graph G with k-bit label $x(u)$ at every node u, a *label-preserving* automorphism ϕ of G is an automorphism satisfying $x(u) = x(\phi(u))$ for every node u. Let

$$\text{symmetry} = \{(G, x) : \text{ there is a non-trivial label-preserving automorphism for } G\},$$

where, by non-trivial, we mean distinct from the identity mapping ($\phi(v) = v$, for every node v). Note that most (unlabeled) graphs are asymmetric. More precisely, Erdős and Rényi [13] showed that there are $(1 - o(1))2^{\binom{n}{2}}$ asymmetric graphs with n nodes.

The proof that symmetry requires $\Omega(n^2 + nk)$ bits to be verified in n-node networks with k-bit labels is based on a construction used in [23] to prove a lower bound on the size of the certificates when using the conjunction operator. We show that the arguments from [23] can be extended to apply to languages on labeled graphs (i.e., not only to graph properties), and extended to apply to all possible operators f for interpreting b-bit outputs (i.e., not only to the conjunction operator for 1-bit outputs). So, let $k \geq 0$, $t \geq 0$, and $b \geq 1$ be three integers.

Let $\mathcal{F}_{n,k}$ be the family of labeled graphs (G, x) where G is a non-symmetric graph with n-nodes, and $|x(u)| = k$ for every node u of G. More precisely, by enumerating the nodes of G from 1 to n in an arbitrary manner, we select a unique (labeled) instance of each non-symmetric graph with n nodes, to be placed in $\mathcal{F}_{n,k}$. It results from the same analysis as in [23] that

$$|\mathcal{F}_{n,k}| = 2^{kn} \, \frac{(1 - o(1))2^{\binom{n}{2}}}{n!}$$

and thus $\log |\mathcal{F}_{n,k}| = \Theta(n^2 + nk)$. Now, for every two labeled graphs (F_1, x_1) and (F_2, x_2) in $\mathcal{F}_{n,k}$, let $(G, x) = (F_1, x_1) + (F_2, x_2)$ be the labeled graph formed by a copy of F_1 together with its labels x_1, a copy of F_2 together with its labels x_2, and a path P of $4t+1$ nodes (all with label 0^k), connecting node 1 in F_1 to node 1 in F_2. The number of nodes in G is therefore $2n + 4t + 1 = \Theta(n)$. Let

$$\mathcal{C} = \{(G, x) = (F_1, x_1) + (F_2, x_2) \ : \ (F_1, x_1) \in \mathcal{F}_{n,k} \text{ and } (F_2, x_2) \in \mathcal{F}_{n,k}\}.$$

We show that even verifying symmetry-membership for labeled graphs in $\mathcal{C}$ requires $\Omega(n^2 + nk)$-bit certificates. Since all graphs in $\mathcal{F}_{n,k}$ are non-symmetric, we get that, for any $(G, x) \in \mathcal{C}$, we have $(G, x) \in$ symmetry if and only if $(F_1, x_1) = (F_2, x_2)$. (Recall that the graphs in $\mathcal{F}_{n,k}$ are labeled, and thus equality here means the existence of a label-preserving isomorphism between F_1 and

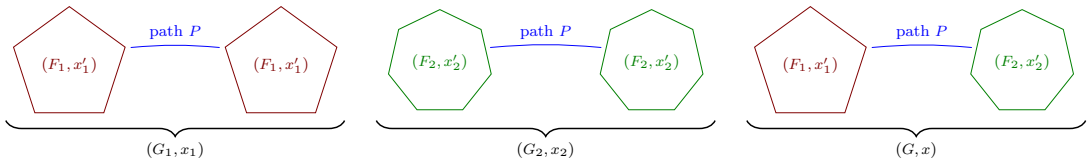


Fig. 2: Illustration of the "cutting and gluing" construction of (G, x) from (G_1, x_1) and (G_2, x_2).

F_2). Let $\mathcal{C}_{\text{sym}}$ be the subset of $\mathcal{C}$ consisting of symmetric graphs in $\mathcal{C}$, i.e.,

$$\mathcal{C}_{\text{sym}} = \mathcal{C} \cap \mathsf{symmetry}.$$

We have:

$$\mathcal{C}_{\text{sym}} = \{(G, x) = (F, x') + (F, x') \ : \ (F, x') \in \mathcal{F}_{n,k}\}.$$

Note that $|\mathcal{C}_{\text{sym}}| = |\mathcal{F}_{n,k}|$. Also note that $|\mathcal{F}_{n,k}| \geq 2^{c(n^2+nk)}$ for some constant $c > 0$, and for large enough values of n. Assume now, for the sake of contradiction, that one can verify $\mathsf{symmetry}$ in t rounds with certificates of size $s \leq \frac{c}{8t+2}(n^2 + nk)$ bits per node, using algorithm $\mathcal{A}$ with global interpretation f on b-bit outputs. Then, for every configuration in $\mathcal{C}$, the path P includes $4t + 1$ certificates, for a total of $(4t + 1)s$ bits, that is at most $\frac{c}{2}(n^2 + nk)$ bits. Therefore, there are at least

$$R = 2^{\frac{c}{2}(n^2+nk)}$$

graphs in $\mathcal{C}_{\text{sym}}$ that have the same collection of certificates on their respective paths P. On the other hand, for an $(n+t)$-node graph with b bits of output per node, the total number of possible multi-sets the verification algorithm $\mathcal{A}$ can produce on this graph is

$$N = \left(\!\!\binom{2^b}{n+t}\!\!\right) = \binom{2^b + n + t - 1}{n+t} = \binom{2^b + n + t - 1}{2^b - 1}$$

where $\left(\!\binom{x}{y}\!\right)$ denotes the multiset coefficient "x multi-choose y". Therefore,

$$N = \frac{(n + t + 1)(n + t + 2) \cdots (n + t + 2^b - 1)}{(2^b - 1)!} = O(n^{2^b}).$$

So, let us assign identities to every graph $(G, x) = (F, x') + (F, x')$ in $\mathcal{C}_{\text{sym}}$ as follows. One copy of (F, x') is given identities from 1 to n, while the other copy of (F, x') is given identities from $n + 1$ to $2n$. In both copies, the identity assignment is set with respect to the enumeration of the nodes in F, i.e., the i-th node receives identity i in one copy, and identity $n + i$ in the other copy. Nodes in the path P connecting the two graphs are given identities from $2n + 1$ to $2n + 4t + 1$.

We now use the family of all these graphs $(G, x) = (F, x') + (F, x')$ similarly as a fooling set is used in communication complexity.

Since R is very large compared to N^2, there exist two configurations

$$(G_1, x_1) = (F_1, x'_1) + (F_1, x'_1)$$

and

$$(G_2, x_2) = (F_2, x'_2) + (F_2, x'_2)$$

in $\mathcal{C}_{\text{sym}}$ that receive the same collection of certificates on their respective path P, and for which $\mathcal{A}$ produces the same multi-set M_1 of outputs in the copies of (F_1, x'_1) and (F_2, x'_2) connected to the nodes with identities $2n+1, \ldots, 2n+t$ on P, and the same multi-set M_2 of outputs in the copies of (F_1, x'_1) and (F_2, x'_2) connected to the nodes with identities $2n + 3t + 2, \ldots, 2n + 4t + 1$ on P. Let us denote by M_0 the multi-set of outputs produced by $\mathcal{A}$ on the $2t + 1$ nodes at the middle of P in both configuration (G_1, x_1) and (G_2, x_2).

Now, consider the following configuration (G, x) formed by "cutting and gluing" (G_1, x_1) and (G_2, x_2) as shown in Figure 2. More precisely, (G, x) is formed by connecting (F_1, x'_1), $(P, 0^k)$, and (F_2, x'_2), with identities in $[1, n]$ for F_1, in $[n + 1, 2n]$ for F_2, and in $[2n+1, 2n+4t+1]$ for P. Let us provide these nodes with the certificates inherited from these respective copies of (F_1, x'_1) in (G_1, x_1), and (F_2, x'_2) in (G_2, x_2). Each node with identity in $\{1, \ldots, n\} \cup \{2n + 1, \ldots, 2n + t\}$ (resp., with identity in $\{n + 1, \ldots, 2n\} \cup \{2n + 3t + 2, \ldots, 2n + 4t + 1\}$) has the same local view of radius t in (G, x) as in (G_1, x_1) (resp., (G_2, x_2)). Moreover, nodes in the middle part of the path, with identities in $[2n + t + 1, 2n + 3t + 1]$ have the same view in (G, x) as in (G_1, x_1) and (G_2, x_2). Therefore, the verification algorithm $\mathcal{A}$ outputs the same multi-set $M_0 \cup M_1 \cup M_2$ for the illegal configuration (G, x), as it does for the legal configurations (G_1, x_1) and (G_2, x_2), yielding the desired contradiction. □

Remark. By inspecting R and N in the proof of Theorem 3, we can notice that the theorem holds even if the number of output bits per node is up to $c' \log(n^2 + nk)$, for $c' < 1$.

5 Verifying cycle-freeness with constant-size certificates

In this section, we show that, for languages in NLD, restricting the interpretation to the use of the conjunctive operator may have a significant cost in terms of certificate size. For instance, it is known [28] that verifying cycle-freeness using the conjunction operator requires $\Omega(\log n)$-bit certificates for n-node graphs. (Recall that the considered graphs are always assumed to be connected). This holds even if the certificates can depend on the identity assignment, and even if the verification can take an arbitrarily large (but constant) number of rounds. In contrast, we show that using both the conjunction and the disjunction operators simultaneously, on 2-bit outputs, enables to verify cycle-freeness in a single round, using certificates of only $O(1)$ bits. Moreover, as we can see in the proof of this result, the decision is made by applying a 2-bit logical operator that is idempotent, commutative, and associative, and thus with all the desirable properties to be used in environments supporting gossip protocols, as well as in computing models restricting the bandwidth of the links.

Theorem 4 *Cycle-freeness can be distributedly verified in one round, with certificates of constant size, and two output bits per node.*

Proof To establish the theorem, we first describe the collection of $O(1)$-bit certificates assigned to the nodes in the case of a valid instance, i.e., for a tree T. The certificate assigned to node v is a pair $y(v) = (r(v), d(v))$, where $r(v)$ is on one bit, and $d(v)$ is on two bits. Every certificate is thus encoded using three bits. To define the assignment of these bits at node v, let us pick an arbitrary node u of T, and set u as the root of T. Set $r(u) = 1$, and $r(v) = 0$ for every node $v \neq u$. For every $v \in V(T)$, let $d(v) = \text{dist}_T(v, u) \bmod 3$, where $\text{dist}_T(x, y)$ denotes the distance in T between nodes x and y, i.e., the minimum number of edges of a path from x to y in T.

We now describe the verification algorithm. It runs in a single round, during which every node v sends its certificate $y(v)$ to all its neighbors, and receives all the certificates of its neighbors. Given its own certificate, and the certificates of its neighbors, every node v then computes a pair of bits (a_v, b_v) as follows. First, every node v checks whether it has at most one neighbor w with $d(w) = d(v) - 1 \pmod 3$. Node w is called the parent of v. More precisely, if $r(v) = 1$ then there must be no parent for v, and, if $r(v) = 0$ then there must be exactly one parent for v. Similarly, v checks whether all its neighbors w different from its parent satisfy $d(w) = d(v)+1 \pmod 3$. All such nodes are called the children of v. If any of these tests is not passed, then v aborts, and outputs $(0, 0)$. If node v has not aborted, then it has identified its parent, and its children (apart from the root which has no parent), and it outputs $(1, r(v))$. This completes the description of the verification algorithm.

We now describe the interpretation of the collection of 2-bit outputs $\{(a_i, b_i), i = 1, \ldots, n\}$. It is the result of the following operator:

$$f(\{(a_i, b_i), i = 1, \ldots, n\}) = \Big(\bigwedge_{i=1}^{n} a_i\Big) \wedge \Big(\bigvee_{i=1}^{n} b_i\Big).$$

By construction, if T is a tree, then $f = 1$. Indeed, all tests are passed successfully, and thus the (unique) node u with $r(u) = 1$ returns $(1, 1)$ while all the other nodes return $(1, 0)$. Establishing that $f = 0$ whenever T is not a tree, independently of the certificates given to the nodes, is based on the fact that, if all tests are passed (i.e., if $\bigwedge_{i=1}^{n} a_i = 1$) then there cannot be a node v with $r(v) = 1$, and therefore $\bigvee_{i=1}^{n} b_i = 0$, yielding $f = 0$. To see why this is indeed the case, assume that the given (connected) graph G is not a tree. Assume moreover that the verification algorithm returns a set $\{(a_i, b_i), i = 1, \ldots, n\}$ such that $\bigwedge_{i=1}^{n} a_i = 1$. (Note that if this is not the case, then $f = 0$, and we are done). Since $\bigwedge_{i=1}^{n} a_i = 1$, every edge of G is given an orientation, from child to parent, and this orientation is locally consistent. That is, every node has exactly one outgoing edge, and a (potentially empty) set of incoming edges, apart from nodes marked $r(v) = 1$, if any, which may have no outgoing edges. Therefore, G has at most n edges. Since G is connected and not a tree, every node must have exactly one outgoing edge, and thus there cannot be a root node in G, i.e., a node v with $r(v) = 1$. Thus, $b_i = 0$ for all i, yielding $f = 0$, which completes the proof of the theorem.

Note nevertheless that certificates could be encoded with only two bits, because only four certificates could actually be used: $(1, 0)$, $(0, 0)$, $(0, 1)$, and $(0, 2)$. Similarly, note that although the output is on two bits, only three different outputs are actually used: $(0, 0)$, $(1, 0)$, and $(1, 1)$. □

6 Deciding cycle-freeness requires logarithmic-size outputs

In this section, we show that, for any positive even integer d, every distributed decision algorithm for cycle-freeness in connected graphs with maximum degree at most d requires outputs of size at least $\lceil \log d \rceil - 1$ bits. To establish this result, we revisit the classical notion of order-invariance introduced by Naor and Stockmeyer [32], and extend their reduction result to distributed languages that are not locally checkable.

6.1 Order-invariance revisited

In this section, we show that, without loss of generality, one can consider only *order-invariant* distributed decision algorithms when considering graphs with constant maximum degree. Recall that an order-invariant distributed algorithm is a distributed algorithm for which the output at any given node does not depend on the actual values of the identities of the nodes in its vicinity, but only on the relative order of these identities. More precisely, let $B_G(v,t)$ be the open ball of radius t around node v in graph G, that is, $B_G(v,t)$ is the subgraph of G induced by all nodes at distance at most t from v, excluding the edges between the nodes at distance exactly t from v.

Definition 3 A t-round algorithm $\mathcal{A}$ is order-invariant if the following holds: for every graph G, for every node v of G, and for every two identity assignments id and id' of the nodes in G, if the ordering of the nodes in $B_G(v,t)$ induced by id is identical to the one induced by id', then the output of $\mathcal{A}$ at node v is the same in both (G,id) and (G,id').

The *construction* task defined by a distributed language $\mathcal{L}$ consists, for every node v of every graph G, of computing $x(v) = \mathsf{out}(v)$ such that $(G,x) \in \mathcal{L}$. More generally, every node v may be given some input $\mathsf{in}(v)$, and the objective of every node v is then to compute $\mathsf{out}(v)$ so that $x = (\mathsf{in},\mathsf{out})$ satisfies $(G,x) \in \mathcal{L}$. In their seminal paper, Naor and Stockmeyer [32] consider the subclass LCL of *locally checkable labelings*, where LCL is a class of distributed languages that are defined on graph families with constant maximum degree, and with constant label size. Let us rephrase this definition using the terminology of this paper.

Definition 4 Let k be a non-negative integer. Let

$$\mathrm{LCL}(k) = \Big\{\mathcal{L} \in \mathrm{LD} : \forall (G,x) \in \mathcal{L}, \max_{v \in V(G)} \deg(v) \leq k \text{ and } x : V(G) \to \{0,1\}^k\Big\}.$$

That is, LCL(k) is LD restricted to distributed languages on labeled graphs with maximum degree at most k, and with labels on at most k bits. By definition:

$$\mathrm{LCL} = \bigcup_{k \geq 0} \mathrm{LCL}(k).$$

Note that it follows from the definition that, for every distributed language $\mathcal{L}$ whose labels are composed of input-output pairs $x = (\mathsf{in},\mathsf{out})$, if $\mathcal{L} \in \mathrm{LCL}(k)$ then both the inputs and the outputs are on at most k bits. Theorem 3.3 in [32] establishes that, for every language $\mathcal{L} \in \mathrm{LCL}$, if there exists a *construction* algorithm for $\mathcal{L}$ running in $t = O(1)$ rounds, then there exists an *order-invariant construction* algorithm for $\mathcal{L}$ running in $O(1)$ rounds. In the context of this paper, the distributed language corresponding to deciding a given distributed language $\mathcal{L} = \{(G,x)\}$ using a function f to interpret the outputs of the nodes is

$$\widehat{\mathcal{L}}_f = \{(G,(x,\mathsf{out})) : (G,x) \in \mathcal{L} \iff f(\mathsf{out}) = \text{true}\}.$$

That is, given a labeled graph (G,x), the algorithm produces individual outputs out that are globally interpreted by f as true if $(G,x) \in \mathcal{L}$, and as false otherwise. In particular, the language corresponding to deciding cycle-freeness using the logical conjunction operator is

$$\widehat{\mathcal{L}}_\wedge = \{(G,\mathsf{out}) : G \text{ is cycle-free} \iff \bigwedge_{v \in V(G)} \mathsf{out}(v) = \text{true}\}.$$

Similarly, the language corresponding to deciding cycle-freeness by summing up the nodes' degrees is

$$\widehat{\mathcal{L}}_\Sigma = \{(G,\mathsf{out}) : G \text{ is cycle-free} \iff \sum_{v \in V(G)} \mathsf{out}(v) = 2(n-1)\}.$$

In both cases, the input x plays no role, and is omitted. However, the input x may play a role for some other tasks, like computing a MIS of maximum weight, with $x(v)$ equal to the weight of node v. Observe that such languages $\widehat{\mathcal{L}}_f$ are not necessarily locally checkable, even for input graphs of bounded degrees. For instance, neither $\widehat{\mathcal{L}}_\wedge$ nor $\widehat{\mathcal{L}}_\Sigma$ belong to LCL. Hence, Theorem 3.3 in [32] does not apply to our setting. We extend this latter theorem to non locally checkable languages, i.e., the prerequisite of our theorem does not assume $\mathcal{L} \in \mathrm{LCL}$. Our proof uses the infinite version of Ramsey's Theorem, that we recall here.

Theorem 5 ([35]) *For any (finite or infinite) set X, and any positive integer r, let us denote by $X^{(r)}$ the set of all subsets of X with size exactly r. Let X be a countably infinite set, let r and s be two positive integers, and let $c : X^{(r)} \to \{0,\ldots,s-1\}$ be a coloring of each set in $X^{(r)}$ by an integer between 0 and $s-1$. Then there exists an infinite set $Y \subseteq X$ such that the image by c of $Y^{(r)}$ is a singleton (that is, all sets in $Y^{(r)}$ are colored the same by c).*

Both the proof of Theorem 3.3 in [32] and our proof of Theorem 6 below show, using some version of Ramsey's Theorem, that there exists a sufficiently large set U of identifiers such that a given t-round construction algorithm $\mathcal{A}$ is order-invariant when the identities

are restricted to the set U. Then, again in both proofs, the sought t-round construction algorithm $\mathcal{A}'$ that is order-invariant for any identities is defined as $\mathcal{A}$ executed on the gathered local information where nodes are virtually assigned new identities from U, typically the smallest ones. Therefore, different nodes may use the same virtual identities from the set U, which means that the correctness of $\mathcal{A}'$ is not a direct consequence of the correctness of $\mathcal{A}$. In the proof of Theorem 3.3 in [32], the set U is finite but sufficiently large to provide identities to constant-radius but rather large balls around any node of the graph. Thanks to the restriction to LCL languages, this is sufficient to prove the overall correctness of $\mathcal{A}'$. In our case however, when languages are not restricted to LCL languages, such a finite set U is not sufficient because graphs in the language can be arbitrarily large. This is why we use the infinite version of Ramsey's Theorem, which allows us to exhibit an infinite set U from which enough identities for the whole graph can be taken.

Theorem 6 *For every non-negative integers k, t, d, and every distributed language $\mathcal{L}$ defined on connected labeled graphs with maximum degree d, and k-bit labels, if there exists a t-round* construction *algorithm $\mathcal{A}$ for $\mathcal{L}$, then there exists a t-round* order-invariant construction *algorithm $\mathcal{A}'$ for $\mathcal{L}$.*

Proof Let us consider the collection of all graphs isomorphic to some ball $B_G(v,t)$ of radius t, centered at some node v in some graph G with maximum degree d. Let $\mathcal{B}$ be the collection of all such balls $B_G(v,t)$ with nodes labeled by k-bit labels. There is a finite number β of distinct labeled balls in $\mathcal{B}$, i.e., either non-isomorphic balls, or isomorphic balls but labeled differently.

We enumerate these labeled balls from 1 to β arbitrarily, and we let n_i be the number of nodes in the i-th ball, for $i = 1, \ldots, \beta$. For every i, the nodes of the i-th labeled ball can be ordered in $n_i!$ different manners, corresponding to the $n_i!$ permutations in Σ_{n_i}, the set of permutations of n_i elements. We consider the $N = \sum_{i=1}^{\beta} n_i!$ ordered labeled balls $B_{i,\sigma}$, for $i = 1, \ldots, \beta$, and $\sigma \in \Sigma_{n_i}$, and we enumerate these ordered labeled balls as $\mathbf{B}_1, \ldots, \mathbf{B}_N$ in an arbitrary order. Note that each labeled ordered ball $\mathbf{B}_i$ has an implicit ordering σ_i associated to it. Using these ordered labeled balls, we define an infinite set $\mathcal{I}$ of identities as follows.

Let $X_0 = \mathbb{N}$, and assume that we have already secured the existence of a sequence of infinite sets $X_0 \supseteq X_1 \supseteq \cdots \supseteq X_j$, $0 \leq j < N$, such that, for every i, $1 \leq i \leq j$ the output of the construction algorithm $\mathcal{A}$ at the center of $\mathbf{B}_i$ is the same for all possible identity assignments to the nodes in $\mathbf{B}_i$ with values in X_i respecting the ordering σ_i of the nodes in $\mathbf{B}_i$.

Let r be the number of nodes in $\mathbf{B}_{j+1}$. We define the coloring

$$c : X_j^{(r)} \to \{0, \ldots, 2^k - 1\}$$

as follows. For each r-element set $I \in X_j^{(r)}$, assign r pairwise distinct identities to the nodes of $\mathbf{B}_{j+1}$ using the r values in I, and respecting the order σ_{j+1} of the nodes in $\mathbf{B}_{j+1}$. Then, define $c(I)$ as the output of Algorithm $\mathcal{A}$ at the center of $\mathbf{B}_{j+1}$ under this identity assignment to the nodes of $\mathbf{B}_{j+1}$. By Theorem 5, there exists an infinite set $Y_j \subseteq X_j$ such that all r-element sets $I \in Y_j^{(r)}$ are given the same color. We set $X_{j+1} = Y_j$. We proceed that way until we exhaust all balls $\mathbf{B}_i$, $i = 1, \ldots, N$, and we set $\mathcal{I} = X_N$.

By construction, the set $\mathcal{I}$ satisfies that, for every ball $B_{i,\sigma}$, for $i = 1, \ldots, \beta$, and $\sigma \in \Sigma_{n_i}$, the output of $\mathcal{A}$ at the center of $B_{i,\sigma}$ is the same for all identity assignments to the nodes of $B_{i,\sigma}$ with identities taken from $\mathcal{I}$ and assigned to the nodes in the order σ.

We now define the order-invariant algorithm $\mathcal{A}'$ as follows. Every node v inspects its radius-t ball $B_G(v,t)$ around it in the actual graph G. In particular, it collects the inputs to the nodes, if any, and the identities of the nodes in that ball. Let σ be the ordering of the nodes in $B_G(v,t)$ induced by their identities. Node v simulates $\mathcal{A}$ by reassigning identities to the nodes of $B_G(v,t)$ using the $r = |B_G(v,t)|$ smallest values in $\mathcal{I}$, in the order specified by σ, and outputs what would have outputted $\mathcal{A}$ if nodes were given these identities.

$\mathcal{A}'$ is well defined, as nodes can be provided with the $\nu = \sum_{i=0}^{t} d^i$ smallest integers in the set $\mathcal{I}$. (That is, nodes do not need to know the entire set $\mathcal{I}$, but only a finite number of values in $\mathcal{I}$). Also, by construction, $\mathcal{A}'$ is order-invariant. To establish that $\mathcal{A}'$ is correct, let us consider some n-node input labeled graph (G, x'), with nodes provided with pairwise distinct identities. Now rename the nodes of G by taking pairwise distinct identities only in $\mathcal{I}$, conserving the same order of the nodes induced by the original identifiers, and let $x'' = \{x''(v), v \in V(G)\}$ be the output of $\mathcal{A}$ in this context. This output is precisely the multi-set outputted by $\mathcal{A}'$ in G with the original identities. Indeed, every node v relabels its radius-t ball with the smallest identities in $\mathcal{I}$, respecting the order induced by the global identities in $\mathcal{I}$, and $\mathcal{I}$ is precisely defined so that the output of v will be the same in both cases. In other words, the output of $\mathcal{A}'$ is precisely the output of $\mathcal{A}$ if nodes were assigned identities restricted to be in $\mathcal{I}$. Hence, since $\mathcal{A}$ is correct, it follows that $\mathcal{A}'$ is correct as well. □

In the next subsection, we show that deciding cycle-freeness in bounded degree graphs requires outputs on

lots of bits, namely outputs on a number of bits logarithmic in the (maximum) degree. The proof uses order-invariance.

6.2 Deciding cycle-freeness

In this section, we prove that cycle-freeness $\notin$ XLD, as cycle-freeness requires outputs on $\Omega(\log n)$ bits to be decided. To establish this result, we show that, on graphs with maximum degree d, outputs on $\lceil \log d \rceil - 1$ bits are required.

Theorem 7 *For any positive even integer d, every distributed decision algorithm for cycle-freeness in connected graphs with maximum degree d has to generate outputs of size at least $\lceil \log d \rceil - 1$ bits.*

The rest of the section is entirely dedicated to prove this result. The proof is by contradiction. Let t and d be two positive integers, and assume that d is even. We assume the existence of a t-round distributed decision algorithm $\mathcal{A}$ for cycle-freeness in connected graphs with maximum degree d, which outputs at most $\lceil \log d \rceil - 2$ bits at each node. We first start by shrinking the set of candidate algorithms $\mathcal{A}$. Indeed, as a direct consequence of Theorem 6, we get the following:

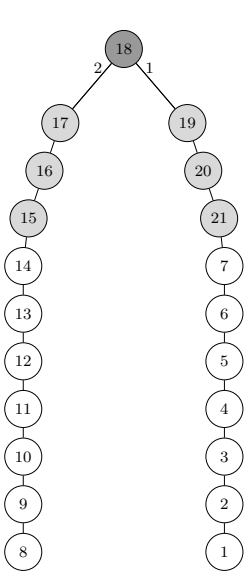
Corollary 1 *If there exists a t-round distributed decision algorithm $\mathcal{A}$ for cycle-freeness with b-bit outputs in connected graphs with maximum degree d, then there is an* order-invariant *t-round distributed decision algorithm $\mathcal{A}$ for cycle-freeness with b-bit outputs.*

Based on this latter result, we now show that every order-invariant decision algorithm $\mathcal{A}$ for cycle-freeness in connected graphs with maximum degree at most d has output size at least $\lceil \log d \rceil - 1$ bits. The intuition is as follows. We will focus our attention on so-called type-i nodes, with i being an even integer between 2 and d. Intuitively, such nodes are defined as nodes of degree i that only "see" a tree of nodes with degree i in their t-neighborhood, with a particular ordering of their identities. We will construct two (connected) graphs, with their corresponding identity assignments, such that only one of these two graphs is a tree, while the multi-set of the local views gathered by the nodes in the two graphs will only differ by their numbers of type-i and type-j nodes, for some $i \neq j$. Any decision algorithm for cycle-freeness has to distinguish the two graphs, and has thus to return different output values to type-i and type-j nodes. This will prove that any distributed tester for cycle-freeness must have at least $d/2$ different output values, establishing the theorem.

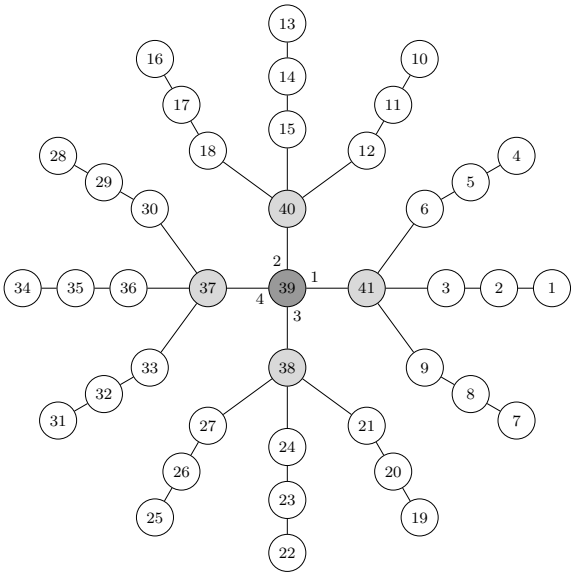
We now define formally two families of trees, which will be used as building blocks in our constructions. Examples of such trees are given in Figure 3.

Let i, $2 \leq i \leq d$, be an even integer. We define the trees T_i and T'_i as follows. For T_i (resp. T'_i), we first build a rooted tree of height $t+1$ (resp. $t+2$) where all internal nodes have degree i and all leaves are at the same distance from the root. In T_i, the root is called a *downtown* node, while in T'_i, the $i+1$ most central nodes (i.e., the root and its i neighbors) are the *downtown* nodes. In both cases, all the other internal nodes are called *suburb* nodes. Note that the downtown node closest to a leaf is at distance exactly $t+1$ from that leaf. For ease of description of our constructions, and for simplifying our arguments, we assign port numbers to the edges incident to the internal nodes of these two trees. More specifically, for each internal node v, a distinct port number between 1 and $\deg(v)$ is assigned to every edge e incident to v. The port numbers are assigned in such a way that, for every edge e whose extremities are two nodes with the same degree i, if $p \in [1, i]$ is one of the two port numbers assigned to e, then $i - p + 1$ is the other port number assigned to e. Then, we apply another transformation, which consists of replacing every leaf of these two trees by an extremity of a path of length $2t+1$. The trees T_i and T'_i are the trees resulting from this second transformation. In these two trees, all the nodes that are neither downtown nodes, nor suburb nodes, are called *countryside* nodes. Note that all countryside nodes are of degree 2 or 1.

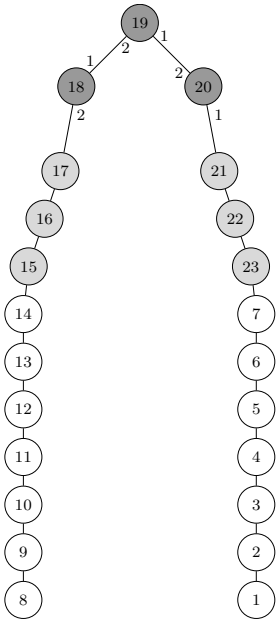
Before describing the identity assignments for these trees, let us make the following observations. An infinite regular tree of degree i can be viewed as the Cayley graph of the free group of rank $i/2$. More precisely, let $\{a_1, a_2, \ldots, a_{i/2}\}$ be the set of the $i/2$ generators of the free group $(F, \star)$ of rank $i/2$. The Cayley graph associated to this group is a directed arc-labeled graph with the following properties: the set of nodes is the set F, and there is an arc with label a_p, $1 \leq p \leq i/2$, from node g to node g' if and only if $g' = g \star a_p$. By replacing each arc (g, g') with label a_p by an (undirected) edge $\{u, v\}$ with port number p at u and port number $i-p+1$ at v, we get the infinite regular tree of degree i with a port-numbering similar to the one we have described for T_i and T'_i. More precisely, for each node in this infinite tree, the edges incident to it are assigned a port number from 1 to i such that if p is one of the port numbers assigned to some edge, then $i - p + 1$ is the other port number assigned to that same edge. Besides, it is known since at least the 1940's (see [33]) that any finitely generated free group is bi-ordered, i.e., it admits a total order $\preceq$ such that, for any three elements $a, b,$



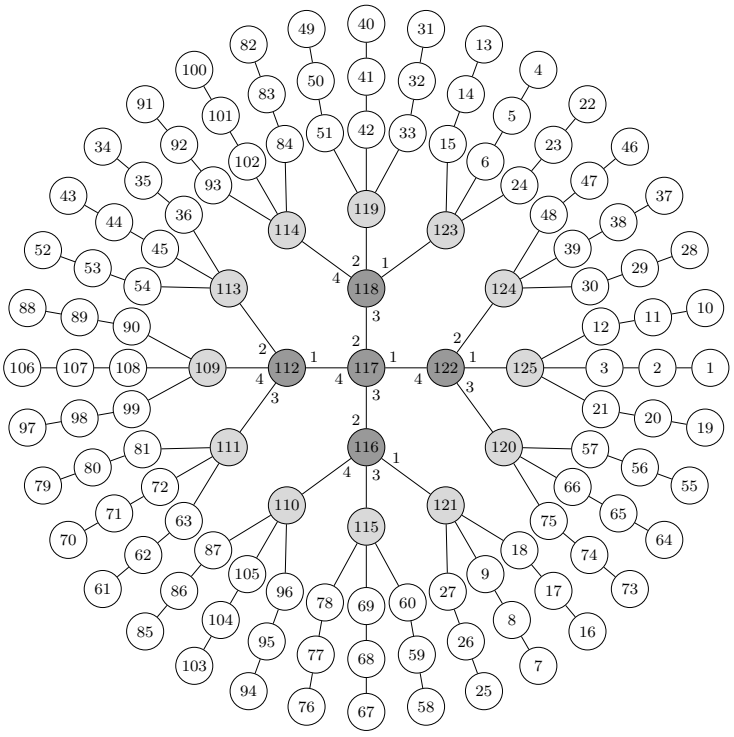
(a) T_2 $(t = 3)$



(b) T_4 $(t = 1)$



(c) T'_2 $(t = 3)$



(d) T'_4 $(t = 1)$

Fig. 3: *Examples of T_i and T'_i. The downtown, suburb, and countryside nodes are depicted as black, grey, and white nodes respectively. Port numbers are only indicated for the downtown nodes. The other port numbers can be inferred using the same cyclic ordering of port numbers around the nodes.*

and c of the group,

$a \preceq b \Longrightarrow a \star c \preceq b \star c$ and $c \star a \preceq c \star b$.

In other words, the total order $\preceq$ is stable by translation. In the infinite regular tree of degree i, it means that if, for some node u and some sequence of port numbers s, the node v accessible from u by following the sequence s satisfies $u \preceq v$, then for any node u', the node v' accessible from u' by following the sequence s

satisfies $u' \preceq v'$. This implies that, for any positive integer t, the relative order of the nodes in a t-neighborhood around some node u does not depend on the choice of u but only on the relative positions of the nodes, that is on the sequences of port numbers on the paths between these nodes.

Let us now describe the identity assignment for each of the trees T_i and T'_i. The construction is similar for both cases. The countryside nodes receive the smallest identities, while the suburb and downtown nodes receive the largest identities. More specifically, to every countryside node u, we associate its distance j to its closest leaf, and the sequence s of the first $t+1$ port numbers describing the path going from the closest downtown node to u. The countryside nodes are assigned identities respecting the lexicographic order of their pair (s, j), with ties broken arbitrarily. The suburb and downtown nodes are assigned identities that are compatible with the total order $\preceq$ of the corresponding free group. See Figure 3 for examples of these identity assignments.

Notation. A node having the same t-neighborhood as the t-neighborhood of the unique downtown node of T_i, except for actual identity values, but respecting the order of these identities, is called a *type-i node*.

By construction and due to the properties of the total order $\preceq$, all downtown nodes in T_i and T'_i are type-i nodes. For the same reasons, countryside nodes sharing the same pair (s, j) have the same relative order of identities in their t-neighborhood. The situation is similar for suburb nodes, except that the sequence s in the pair (s, j) is defined in this case as the sequence of port numbers leading to the closest downtown node.

For the purpose of contradiction, assume that there exists an order-invariant distributed decision algorithm for cycle-freeness using fewer than $d/2$ output values. Under this assumption, by the pigeonhole principle, there must exist two even integers i and j, with $2 \leq i < j \leq d$, such that the algorithm outputs the same value for type-i and type-j nodes. Hence, let G'_1 be the graph formed by the disjoint union of one copy of T'_i, with $j-1$ copies of T_j, using disjoint ranges of identity values assigned to the nodes in these copies. Connect $j-1$ disjoint pairs of leaves by an edge to make the graph connected. We denote by G_1 the resulting graph. Note that G_1 is a tree. Similarly, let G'_2 be the graph formed by the disjoint union of $i-1$ copies of T_i with one copy of T'_j, using disjoint ranges of identities assigned to the nodes in these copies. Connect $i-1$ disjoint pairs of leaves by as many edges to make the graph connected. Further, connect $j-i$ other pairs of leaves to create cycles. We denote by G_2 the resulting graph. Note that G_2 is connected but is not a tree. Figure 4 shows the shape of G_1 and G_2 for $t = 1, i = 4, j = 6$.

It follows from the construction that, in G_1, there are exactly $i+1$ downtown nodes of degree i, and $j-1$ downtown nodes of degree j. On the other hand, G_2 contains exactly $i-1$ downtown nodes of degree i, and $j+1$ downtown nodes of degree j. However, concerning the suburb nodes adjacent to a downtown node, both graphs contain $i(i-1)$ such nodes of degree i and $j(j-1)$ such nodes of degree j.

More generally, ignoring the identities assigned to the nodes, but taking into account their relative order, the t-neighborhoods are identical in both graphs G_1 and G_2, with the only exception that G_1 has two type-i nodes more compared to G_2, and two type-j nodes less than G_2. The distributed algorithm $\mathcal{A}$ will thus output the same, resulting in the same decision taken by the interpretation f for both graphs, which contradicts the fact that the algorithm is correct. This contradiction completes the proof of the theorem. □

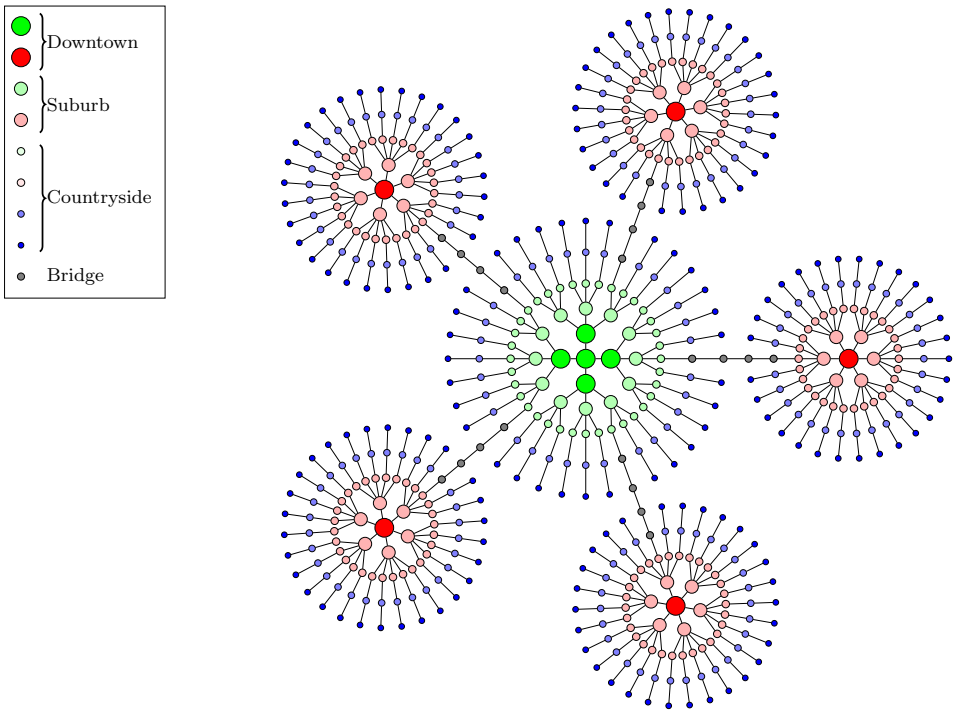
7 Conclusion

In this paper, we have defined and analyzed a generalized version of distributed decision and verification, where the nodes are allowed, after $O(1)$ rounds, to output $O(1)$ bits collected by a central authority which emits a global verdict (true or false) as a function of the collection of individual outputs. We have defined the related classes XLD and XNLD, and established separation results summarized in Figure 1 between these classes, and between them and the "standard" classes LD and NLD.

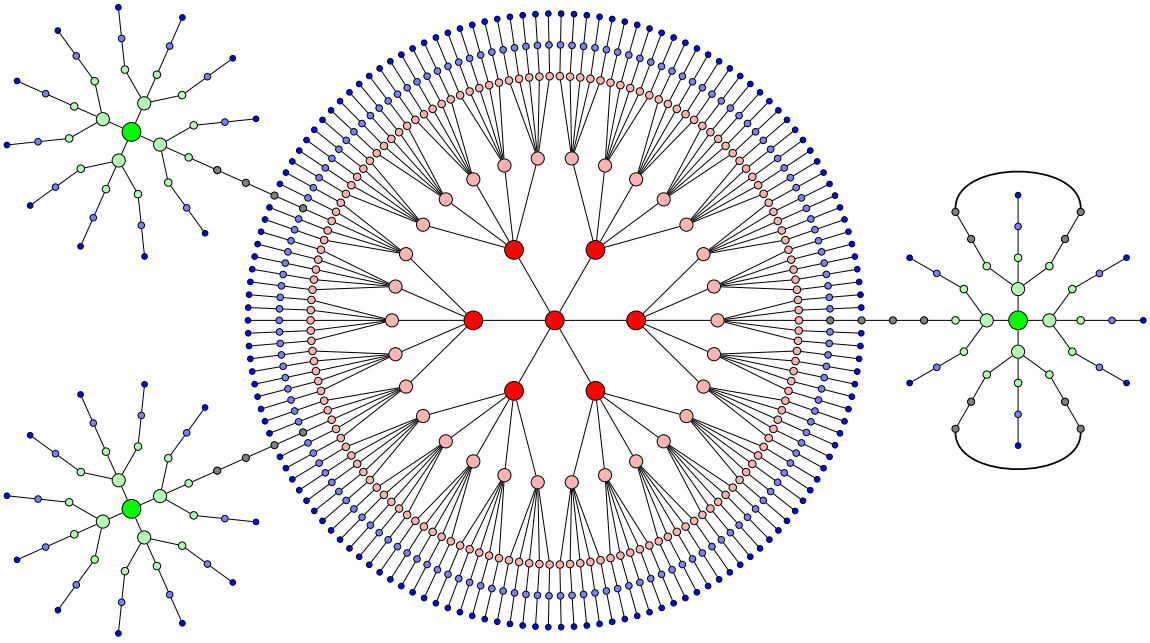
At this stage of our current understanding of distributed decision and verification, it would be of interest to determine whether there exist languages that are complete for XLD and XNLD with respect to local reduction, in the same way miss and its variant $\mathsf{miss}^{\uparrow}$ were shown to be complete for $\mathrm{NLD}^{\#\mathsf{nodes}}$ and NLD, respectively, in [7].

References

1. Y. Afek, S. Kutten, and M. Yung. The local detection paradigm and its applications to self stabilization. *Theoretical Computer Science*, **186**(1–2): 199–230, (1997)
2. N. Alon. Subdivided graphs have linear ramsey numbers. *Journal of Graph Theory* **18**(4): 343–347 (1994)
3. D. Angluin. Local and Global Properties in Networks of Processors (Extended Abstract). In Proc. *12th Annual ACM Symposium on Theory of Computing* (STOC), pages 82–93, 1980.
4. H. Arfaoui, P. Fraigniaud, D. Ilcinkas, F. Mathieu. Distributedly Testing Cycle-Freeness. In Proc. *40th Int.*



(a) G_1



(b) G_2

Fig. 4: *Illustration of the proof of Theorem 7 for $t = 1$. G_1 and G_2 share the same types of nodes (the algorithm always outputs the same value for a given type) and only differ by the number of type-4 and type-6 downtown nodes. Hence, if type-4 and type-6 nodes have the same output value, the same decision will be taken for G_1 and G_2. Note that these two graphs have quite a lot of different types of nodes when considering (the relative order of) the identities of the nodes. For the sake of clarity, in this figure, we do not consider the identities, but we take into account, on the other hand, that nodes are aware of the degree of their neighbors.*

Workshop on Graph-Theoretic Concepts in Computer Science (WG), pages 15–28, 2014.
5. H. Arfaoui, P. Fraigniaud, and A. Pelc. Local Decision and Verification with Bounded-Size Outputs. In Proc. *15th Symp. on Stabilization, Safety, and Security of Distributed Systems* (SSS), pages 133–147, 2013.
6. B. Awerbuch, B. Patt-Shamir, G. Varghese, and S. Dolev. Self-Stabilization by Local Checking and Global Reset. In Proc. *Workshop on Distributed Algorithms and Graphs* (WDAG), Springer, LNCS 857, pages 326–339, 1994.
7. A. Balliu, G. D'Angelo, P. Fraigniaud, D. Olivetti: What can be verified locally? *Journal of Computer and System Sciences* **97**: 106–120 (2018)
8. F. Becker, A. Kosowski, N. Nisse, I. Rapaport, and K. Suchan. Allowing each node to communicate only once in a distributed system: shared whiteboard models. In Proc. *24th ACM Symp. on Parallelism in Alg. and Architectures* (SPAA), pages 11–17, 2012.
9. F. Becker, M. Matamala, N. Nisse, I. Rapaport, K. Suchan, and I. Todinca. Adding a referee to an interconnection network: What can(not) be computed in one round. In Proc. *25th IEEE Int. Symp. on Parallel and Dist. Proc.* (IPDPS), pages 508–514, 2011.
10. B. Bonakdarpour, P. Fraigniaud, S. Rajsbaum, C. Travers. Challenges in Fault-Tolerant Distributed Runtime Verification. In proc. *7th Int. Symp. on Leveraging Applications of Formal Methods, Verification and Validation* (ISoLA), pages 363-370, 2016.
11. A. Das Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg and R. Wattenhofer. Distributed Verification and Hardness of Distributed Approximation. In Proc. *43rd ACM Symp. on Theory of Computing* (STOC), 2011.
12. S. Dolev, M. G. Gouda, and M. Schneider. Memory Requirements for Silent Stabilization. *Acta Informatica* **36**(6): 447–462 (1999)
13. P. Erdős and A. Rényi. Asymmetric graphs. *Acta Mathematica Academiae Scientiarum Hungaricae* **14**(3–4): 295–315 (1963)
14. G. Even, O. Fischer, P. Fraigniaud, T. Gonen, R. Levi, M. Medina, P. Montealegre, D. Olivetti, R. Oshman, I. Rapaport, I. Todinca: Three Notes on Distributed Property Testing. In proc. *31st Int. Symp. on Distributed Computing* (DISC), pages 15:1-15:30, 2017.
15. L. Feuilloley, P. Fraigniaud: Randomized Local Network Computing. In Proc. *27th ACM Symposium on Parallelism in Algorithms and Architectures* (SPAA), pages 340–349, 2015.
16. L. Feuilloley, P. Fraigniaud: Survey of Distributed Decision. *Bulletin of the EATCS* **119** (2016)
17. P. Fraigniaud, M. Göös, A. Korman, J. Suomela. What can be decided locally without identifiers? In Proc. *32nd ACM Symp. on Principles of Distributed Computing* (PODC), pages 157–165, 2013.
18. P. Fraigniaud, M. M. Halldórsson, A. Korman. On the Impact of Identifiers on Local Decision. In Proc. *16th Int. Conference on Principles of Distributed Systems* (OPODIS), pages 224–238, 2012.
19. P. Fraigniaud, J. Hirvonen, J. Suomela: Node Labels in Local Decision. In Proc. *22nd Int. Colloquium on Structural Information and Communication Complexity* (SIROCCO), pages 31–45, 2015.
20. P. Fraigniaud, A. Korman, M. Parter, D. Peleg. Randomized Distributed Decision. In Proc. *26th Int. Symp. on Distributed Computing* (DISC), pages 371–385, 2012.
21. P. Fraigniaud, A. Korman, and D. Peleg. Towards a complexity theory for local distributed computing. *J. ACM* **60**(5): 35:1–35:26 (2013)
22. P. Fraigniaud, P. Montealegre, R. Oshman, I. Rapaport, and I. Todinca. Lower Bounds for Single-Interaction Distributed Arthur-Merlin and Merlin-Arthur Protocols. In proc. *26th Int. Colloquium on Structural Information and Communication Complexity* (SIROCCO), 2019.
23. M. Göös, J. Suomela. Locally checkable proofs. In Proc. *30th ACM Symp. on Principles of Distributed Computing* (PODC), pages 159–168, 2011.
24. G. Itkis, L. A. Levin. Fast and Lean Self-Stabilizing Asynchronous Protocols. In Proc. *35th IEEE Symp. on Foundations of Computer Science* (FOCS), pages 226–239, 1994.
25. H. Karl, A. Willig. Protocols and Architectures for Wireless Sensor Networks. Wiley, 2007.
26. S. Katz, and K. Perry. Self-stabilizing extensions to for message-passing systems. *Distributed Computing* **7**: 17–26, (1993)
27. Gillat Kol, Rotem Oshman, Raghuvansh R. Saxena: Interactive Distributed Proofs. In 37th ACM Symposium on Principles of Distributed Computing (PODC), pp255-264, 2018
28. A. Korman, S. Kutten, and D. Peleg. Proof labeling schemes. *Distributed Computing* **22**(4): 215–233 (2010)
29. E. Kushilevitz and N. Nisan. *Communication complexity.* Cambridge University Press, 1997.
30. I. Litovsky, Y. Metivier, and W. Zielonka. On the Recognition of Families of Graphs with Local Computations. *Information and Computation* **118**(1): 110–119 (1995)
31. Moni Naor, Merav Parter, Eylon Yogev: The Power of Distributed Verifiers in Interactive Proofs, CoRR, abs/1812.10917, 2018 (`http://arxiv.org/abs/1812.10917`).
32. M. Naor and L. Stockmeyer. What can be computed locally? *SIAM J. Comput.* **24**(6): 1259–1277 (1995)
33. B. H. Neumann. On Ordered Groups. *American Journal of Mathematics* **71**(1): 1–18 (1949)
34. D. Peleg. *Distributed Computing: A Locality-Sensitive Approach.* SIAM, 2000.
35. F. P. Ramsey. On a problem of formal logic. *Proceedings of the London Mathematical Society* **2**(1): 264–286 (1930)
36. A. C. Yao. Some Complexity Questions Related to Distributed Computing. In Proc. *11th ACM Symp. on Theory of Computing* (STOC), pages 209–213, 1979.