HAL
open science

Local Distributed Decision and Verification

Heger Arfaoui

To cite this version:

Heger Arfaoui. Local Distributed Decision and Verification. Networking and Internet Architecture [cs.NI]. Paris Diderot University, 2014. English. NNT : . tel-01274154

HAL Id: tel-01274154

https://inria.hal.science/tel-01274154v1

Submitted on 15 Feb 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Université Paris Diderot - Paris 7
Laboratoire d'Informatique Algorithmique : Fondements et Applications

Thèse

présentée pour l'obtention du diplôme de

Docteur de l'Université Paris Diderot,
spécialité Informatique

à l'École doctorale de Sciences Mathématiques de Paris Centre

Local Distributed Decision and Verification

Par

Heger Arfaoui

Soutenue publiquement le 7 juillet 2014 *devant le jury constitué de :*

Olivier Bournez	Professeur - Ecole Polythechnique	*Rapporteur*
Jérémie Chalopin	Chargé de Recherche - CNRS	*Examinateur*
Pierre Fraigniaud	Directeur de Recherche - CNRS	*Directeur*
Cyril Gavoille	Professeur - Université de Bordeaux	*Rapporteur*
Frédéric Magniez	Directeur de Recherche - CNRS	*Examinateur*
Ioan Todinca	Professeur - Université d'Orléans	*Examinateur*

Résumé

Cette thèse s'inscrit dans le contexte du calcul distribué sur les réseaux, et, plus particulièrement, sur les aspects de *localité* qui apparaissent dans ce cadre. Par l'étude systématique des problèmes de décision, nous introduisons les classes de complexité ULD et UNLD pour la décision et la vérification locale, et présentons des résultats de séparation décrivant une hiérarchie impliquant d'autres classes relatives à la littérature de la décision locale. Ces résultats sont accompagnés de la classification de plusieurs problèmes distribués selon la hiérarchie introduite. Nous examinons et discutons dans ce cadre deux ingrédients ayant un rôle clé dans la décision et la vérification locale : la fonction d'interprétation des sorties, et l'identification des noeuds du réseau.

Nous isolons également dans cette thèse l'aspect de la localité en l'étudiant sous le prisme du modèle "non-signalling", qui bien que n'étant pas un modèle réaliste, offre des possibilités théoriques intéressantes, notamment sur la dérivation de bornes inférieures pour le calcul distribué quantique, sans avoir à manipuler les objets de cette théorie. Finalement, en nous plaçant à la limite extrême des contraintes de localité, nous considérons la classe particulière de jeux à deux joueurs *en l'absence de communication*, et examinons les limites du calcul distribué quantique pour cette classe de jeux.

Abstract

This thesis lays in the context of distributed computing on networks, and more particularly on the *locality* aspects that appear in that context. By the systematic study of decision problems, we introduce the complexity classes ULD and UNLD for local decision and verification respectively, and give separation results describing a hierarchy involving other classes of local decision in the literature. These results are accompanied by a classification of several distributed problems based on the hierarchy we introduce. We examine and discuss two key ingredients in local decision and verification: the interpretation function on the outputs, and node identification.

In this thesis, we also isolate the aspect of locality by studying it through the prism of the non-signaling model, which, even though not realistic, offers interesting theoretical possibilities, including the derivation of lower bounds for distributed quantum computing without having to manipulate objects of that theory. Finally, by placing ourselves at the extreme limit of locality constraints, we consider the particular class of two-player games *in absence of any communication* and examine the limits of quantum distributed computing for this class of games.

Contents

Introduction [FR] **7**
Contexte et motivations . . . 7
Contributions . . . 9

Introduction [EN] **13**
Context and motivation . . . 13
Contributions . . . 15

1 Distributed Computing **17**
1.1 Modeling distributed systems . . . 17
1.1.1 Definitions . . . 17
1.1.2 What is so special about distributed computing? . . . 19
1.2 The $\mathcal{LOCAL}$ model . . . 20
1.2.1 Terminology . . . 20
1.2.2 Construction problems . . . 22
1.3 Conclusion . . . 24

I Decision and verification in the $\mathcal{LOCAL}$ model **25**

2 Local Decision **27**
2.1 Introduction . . . 27
2.2 Decision problems . . . 29
2.2.1 Framework . . . 29
2.2.2 Role of the Interpreter $\mathcal{I}$ . . . 32
2.3 Unrestricted Local Decision (ULD) . . . 34
2.3.1 Definitions . . . 34
2.3.2 Role of identifiers . . . 35
2.3.3 Preliminary results . . . 37

3 Distributed verification **41**
3.1 A flavor of non-determinism . . . 41

3.2 Separation results 44
3.3 On the certificate size 49
3.3.1 Minimum certificate size for universal verification 49
3.3.2 Verifying trees with constant size certificates 52
3.4 Conclusion 54

II Causality and its algorithmic consequences 55

4 Non-Signalling Model: capturing the concept of causality. 57
4.1 Introduction 57
62section.4.2
4.3 Nonlocal computing 65
4.3.1 Non-signalling distributions 65
4.3.2 Non-classical computing 67
4.3.3 Examples and applications 70

5 Computational consequences of the Non-Signalling model 73
5.1 Two-player games 74
5.2 Equivalence classes of games 77
5.3 On the power of quantum correlations 79
5.4 Further work and open problems 85
5.4.1 n-player games 85
5.4.2 n-player boxes and the $\mathcal{LOCAL}$ model 87
5.5 Conclusion 88

Conclusion and Perspectives 91

Introduction

Penser globalement, Agir localement. Cette devise aurait pu constituer le titre de cette thèse. Notre travail s'attache en effet à étudier l'impact de l'information locale sur les performances globales des systèmes distribués. Le terme "système distribué" désigne un ensemble d'entités autonomes partageant peu de ressources et qui doivent collaborer afin d'accomplir une tâche globale.

Contrairement aux systèmes centralisés où les tâches sont exécutées les unes après les autres de manière *séquentielle* par une seule entité, il n'existe a priori pas de mécanisme centralisé dans les réseaux distribués qui puisse coordonner globalement les différents processus. Les différentes unités de calcul du système doivent alors, a priori sans connaissance du réseau dans lequel elles se trouvent, se coordonner pour exécuter une tâche collective, comme par exemple atteindre un consensus à propos d'une valeur de sortie commune. L'algorithmique distribuée cherche à déterminer quelles sont les tâches globales qui peuvent être obtenues dans de tels systèmes.

Contexte et motivations

L'étude des systèmes distribués s'est développée avec l'émergence des réseaux informatiques. Du web aux réseaux pair-à-pair, en passant par les réseaux téléphoniques sans fil ou les réseaux de capteurs aux applications industrielles, ces systèmes sont devenus ubiquitaires. L'aspect distribué s'est donc imposé comme un nouveau paradigme de calcul, qu'il soit inévitable, comme c'est le cas pour Internet par exemple (le réseau est construit *de facto* de cette manière), ou qu'il soit choisi pour augmenter les performances de calcul des processeur en parallélisant certaines tâches. En réalité, des réseaux similaires, ayant les mêmes propriétés, c'est-à-dire caractérisés par une struc-

ture complexe d'agents interconnectés, ont toujours *naturellement* existé. Le réseau social d'un groupe décrivant les relations d'amitiés entre ses personnes, ou le réseau décrivant les interactions possibles entre molécules sont des exemples de tels systèmes.

L'algorithmique distribuée a commencé à se détacher comme discipline à part en informatique depuis les années 70 - 80. Les travaux sur les réseaux dits *anonymes* (lorsque les noeuds sont indiscernables) ont été initié par D. Angluin[4], qui s'est penchée sur le problème d'établissement d'un *centre*, c'est-à-dire une configuration où exactement un seul processeur prend un état unique particulier, à partir d'une configuration de départ où tous les processeurs sont dans le même état (c'est le problème d'*élection de leader*). Elle a également posé la question de savoir quelles propriétés structurelles les processeurs peuvent-ils déterminer sur leur graphe sous-jacent. N. Linial [68] a donné les premiers résultats fondamentaux d'impossibilité dans les réseaux où les noeuds ont des identifiants, ce qui rend ces résultats particulièrrements forts, car l'absence d'identifiants introduisait une difficulté supplémentaire résultant de la symétrie entre les processeurs. Les travaux de [72, 27, 10] donnent les premiers résultats positifs liés aux réseaux avec identifiants. Depuis, plusieurs travaux sur des algorithmes efficaces de résolution de problèmes typiques des systèmes distribués ont été menés dans le cas des réseaux avec identifiants. Nous revenons sur quelques exemples dans le premier chapitre de cette thèse.

Bien que l'algorithmique distribuée soit aujourd'hui un domaine mature, elle souffre toutefois encore de l'absence d'une théorie de la complexité permettant de comparer systématiquement les problèmes en classifiant leur difficulté, comme il en est en algorithmique séquentielle. Ceci est la conséquence de multiples obstacles, dont nous citons en particulier :

La variété des modèles : La nature complexe des systèmes distribués fait qu'il est difficile d'en appréhender tous les aspects à la fois. Ainsi, il est apparu un grand nombre de modèles, chacun capturant un ou plusieurs de ces aspects. Nous citons à titre d'exemple les modèles à mémoires partagées, les modèles de calcul en réseaux, ou le calcul par entités mobiles (robots, agents logiciels, etc.).

La nature même des problèmes : Alors qu'en algorithmique séquentielle la plupart des problèmes étudiés se ramènent à des problèmes de décision, les problèmes en algorithmique distribuée ne se prêtent pas forcément à ce formalisme. Par

exemple, la tâche de diffuser une information dans tout le réseau ne se traduit pas naturellement en problème de décision.

À la lumière de ces observations, nous nous sommes fixés deux objectifs à travers cette thèse. Notre premier objectif est de dégager des analogies avec la complexité séquentielle afin d'essayer de contribuer au développement de définitions de classes de complexité distribuée. Nous essayerons pour cela d'établir un socle cohérent à tous les problèmes en nous ramenant tant que possible à l'étude de problèmes de décision. Nous tenterons d'appréhender cette problématique avec plusieurs approches comprenant le calcul déterministe classique mais en étudiant aussi l'impact du non-déterminisme et du calcul quantique. Pour atteindre cet objectif, en réponse à la première difficulté dégagée plus haut, nous avons choisi de nous intéresser au modèle $\mathcal{LOCAL}$ qui capture l'aspect de connaissance partielle du réseau par les agents. Cet aspect nous a semblé d'une grande importance dans la caractérisation des systèmes distribués : comment effectuer une tâche globale en agissant uniquement à l'échelle locale ? comment obtenir une connaissance totale à partir de connaissances réduites de son voisinage ? comment maintenir des temps d'exécution des tâches indépendants de la taille du réseau ? Par ailleurs, en considérant l'aspect de localité en tant que concept physique intrinsèquement lié au principe de causalité, comment est-il possible à des agents situés en des points spatiaux éloignés de collaborer dans la limite les lois de causalité relativiste ? Répondre aux dernières questions est notre deuxième objectif. Nous nous intéresserons à l'étude de la classe des problèmes *localement décidables*, et nous consacrerons une grande partie de nos travaux à isoler l'aspect de localité en tant que contrainte physique pour en étudier les conséquences d'un point de vue algorithmique et de communication.

Contributions

Les contributions de cette thèse s'articulent autour de deux axes intriqués et sont présentés en deux parties, précédées d'un chapitre introductif (Chapitre 1) qui définit la modélisation adoptée. Une partie des résultats de la Partie I est parue dans les actes de *SSS'13* (15th International Symposium on Stabilization, Safety, and Security of Distributed Systems) [6] et a obtenu le prix du meilleur article (Best Paper Award). Une partie des résultats de la Partie II va faire l'objet d'une publication dans les

Distributed Columns de l'*ACM SIGACT news*.

Dans la première partie, (Chapitres 2 et 3), nous procédons par analogie avec le calcul séquentiel et présentons un cadre qui met l'accent sur les problèmes de décision comme un candidat idéal pour l'étude de la complexité distribué locale. Un algorithme qui résout un problème de décision est un algorithme qui produit une sortie distribuée pouvant être projetée comme une réponse "oui" ou "non" à une question donnée. Toujours par analogie avec le calcul séquentiel, nous étudions l'impact du non-déterminisme sur la résolution de problèmes de décision. En d'autres termes, nous examinons dans quelle mesure les systèmes peuvent vérifier une solution donnée aux entités à l'aide d'un *certificat distribué*. Cette question est capturée par l'étude des problèmes de vérification.

Nos résultats qui se rapportent à ces deux types de problèmes distribués sont les suivants : d'abord, nous introduisons les classes des langages universellement localement décidables (ULD) et les langages universellement localement décidables de manière non-déterministe (UNLD), qui peuvent être vus dans une certaine mesure comme homologues distribués des classes séquentielles P et NP. Nous considérons divers exemples de problèmes pertinents et les plaçons dans leurs classes de décision et de vérification appropriées. Nous établissons aussi des résultats complets de séparation de ULD et UNLD avec les autres classes de la littérature de décision locale. Nous montrons également que, dans notre modèle, tous les langages distribués peuvent être vérifiés, et nous établissons une limite supérieure à la taille des certificats nécessaire pour la vérification. Notre borne est optimale, dans le sens que nous exhibons un problème qui nécessite des certificats de cette taille maximale afin d'être vérifié. Toujours concernant des certificats, nous prouvons que notre modèle a un grand impact sur leur taille en comparaison avec d'autres modèles de décision locale.

La deuxième partie de cette thèse (Chapitres 4 et 5) est dédiée à l'étude de la localité sous son aspect physique, en lien direct avec la notion de causalité. On s'intéressera au modèle dit *non-signalling* qui capture l'impossibilité de l'information à voyager plus vite que la lumière. Ce modèle est complètement décrit par des distributions de probabilités des sorties sachant les entrées. Il est donc fort et facile à manipuler à la fois. Une de ses principales caractéristiques est qu'il englobe les corrélations quantiques. Le principal objectif de cette partie sera donc de fournir des exemples de problèmes informatiques distribués pour lesquels il est possible de

trouver des bornes inférieures strictes pour des *algorithmes quantiques* sans avoir aucunement à manipuler des concepts de la mécanique quantique. Comme une étude de cas, nous abordons la classe des jeux à deux joueurs, où les joueurs doivent calculer une fonction de leurs entrées, en *l'absence de toute communication* entre les deux joueurs, mais *en présence de ressources partagées* telles que l'aléatoire partagé ou l'intrication quantique. Nous montrons que, en dehors d'une classe particulière de jeux, les corrélations quantiques ne sont d'aucune aide, en ce sens qu'il existe un protocole classique (utilisant l'aléatoire partagé) dont la probabilité de réussite est au moins aussi grande que celle de n'importe quel protocole utilisant les ressources quantiques. Ce résultat est vrai pour l'analyse au pire cas et en moyenne. Enfin, nous soulignons que ce dernier résultat met en évidence la pertinence d'adopter de nouvelles formes de décision locale, ce qui est le point principal des classes ULD et UNLD présentées dans la première partie .

Introduction

Think globally, act locally. This motto could have been the title of this dissertation. Our work focuses indeed on studying the impact of local information on the overall performance of distributed systems. The term "distributed system" refers to a set of autonomous computing entities sharing limited resources and aiming to work together to achieve a global task.

Unlike centralized systems where tasks are *sequentially* performed one after the other by a single entity, there is a priori no centralized mechanism in distributed networks that can globally coordinate the various processes. The computational units of the system must then, without a priori knowledge of the network in which they are, coordinate to perform a common task, such as reaching a consensus about a common output value. Distributed computing seeks to determine the global tasks that can be achieved in such systems.

Context and motivation

The study of distributed systems began with the emergence of computer networks. From the Web to peer-to-peer networks, including wireless telephone networks or industrial sensor networks, these systems have become ubiquitous. The distributed aspect has prevailed as a new computing paradigm, wether inevitable, as it is the case e.g. for the Internet (the network is *de facto* built that way), or chosen to increase CPU computing performance by parallelizing tasks. In fact, similar networks, having the same properties, i.e. characterized by a complex structure of interconnected entities, have always *naturally* existed. The social network of a group describing the friendship relations between its people, or the network describing the possible

interactions between molecules are examples of such systems.

Distributed computing began to emerge as a full field in computer science since the 70 - 80. Seminal work on the so-called *anonymous* networks (where nodes are indistinguishable) were initiated by D. Angluin [4], who focused on the problem of establishing a *center*, that is to say, a configuration where exactly one processor takes a particular single state, from a starting configuration where all processors are in the same state (this is the problem of *leader election*). She also raised the question of what structural properties processors can determine about their underlying graph. N. Linial [68] gave the first fundamental impossibility results in networks where nodes have IDs, which makes these results particularly strong, as the symmetry induced by absence of IDs introduces extra difficulty. [72, 27, 10] provide the first positive results related to networks with IDs. Since then, problems on efficient algorithms for solving typical problems of distributed systems has been investigated in the case of networks with identifiers. We give some examples in the first chapter of this dissertation.

Although distributed computing is now a mature field, it still suffers from the lack of a consistent complexity theory allowing to systematically compare the problems by classifying their difficulty, as it is the case with sequential algorithms. This is the result of multiple barriers, among which we mention:

The variety of models: The complex nature of distributed systems makes it difficult to understand all the aspects in one model. Thus, a large number of models has emerged, each capturing one or more of these aspects . We mention as examples the shared memory models, network models, or computing by mobile entities (robots, software agents, etc.).

The nature of the problems: Whereas most sequential algorithmic problems can be reduced to decision problems, problems in distributed computing are not necessarily easily translated to this formalism. For example, the task of broadcasting an information throughout the network does not translate naturally in a decision problem.

In the light of these observations, we set two goals we intent to meet throughout this thesis. Our first goal is to make use of analogies with the sequential complexity to contribute to the development of distributed complexity classes. We will try to establish a consistent basis for problem comparison by placing ourselves in a decision

problem setting each time it is possible. We will take several approaches, including classical deterministic computing, but also by studying the impact of non-determinism and quantum computing. To achieve this goal, and in response to the first challenge stated above, we have chosen to focus on the $\mathcal{LOCAL}$ model [75] that captures the aspect of partial knowledge the entities have about the network. This has seemed to us of a great importance in the characterization of distributed systems: how to perform a global task by acting only at a local scale? how to get a complete knowledge from reduced knowledge of its neighborhood? how to maintain scalability when the size of the network grows? Moreover, considering the aspect of locality as a physical concept intrinsically related to the principle of causality, how is it possible for spatially distant agents to collaborate within the laws of relativistic causality? Answering the latter questions is our second goal. We will focus on the study of the class of *locally decidable* problems, and we will devote a large part of our work to isolating the constraint of locality to study its consequences on an algorithmic and communication point of view.

Contributions

The contributions of this thesis are organized around two entangled axes, and are presented in two parts. An introductory chapter (Chapter 1) defines the adopted model. Some of the results presented in Part I appeared in *SSS'13* proceedings (15th International Symposium on Stabilization, Safety, and Security of Distributed Systems) [6] and obtained the Best Paper Award. Some of the results presented in Part II will soon be published in the *ACM SIGACT news Distributed Columns*.

In the first part, (Chapters 2 and 3), we proceed by analogy with sequential computing, and present a framework that focuses on *decision problems* as an ideal candidate for the study of local distributed complexity. An algorithm solving a decision problem is one that produces a distributed output that can be projected as an answer to a yes-or-no question. Also by analogy with sequential computing, we study the impact of non-determinism on solving decision problems. In other words, we examine to which extent systems can verify a given solution handed to the entities in the form of a *distributed certificate*. This question is captured by the study of *verification problems*.

Our results that relate to these two types of distributed problems are the following: first, we introduce the classes of *Universally Locally Decidable* languages (ULD) and *Universally Nondeterministic Locally Decidable* languages(UNLD), that can be seen to some extent as distributed counterparts of sequential classes P and NP. We consider various examples of relevant problems and place them in their appropriate decision and verification classes. We also establish complete separation results of ULD and UNLD with other classes from the literature of local decision. We then prove that, in our model, *all* distributed languages can be *verified*, and we establish an upper bound on the size of the certificate needed for verification. Our bound is tight in the sense that we exhibit a problem that needs a certificate of that maximal size in order to be verified. Still on the matter of certificates, we prove that our model has a tremendous impact on certificate size in comparison with other models of local decision.

The second part of this dissertation (Chapters 4 and 5) is dedicated to the study of locality and its relativistic-causality aspects. We focus on the *non-signaling* model that captures the no-faster-than-light communication law. This model is fully described by output probability distributions. It is strong and easy to handle at the same time. One of its main features is that it subsumes quantum correlations. The main objective of this part will be to provide examples of distributed computing problems for which it is possible to find tight lower bounds for *quantum algorithms* without having to manipulate concepts from quantum mechanics, at all. As a case study, we address the class of two player games, where players have to compute a function of their inputs, in *absence of any communication* between the two players, but in *presence of shared resources* such as shared randomness or quantum entanglement. We show that, apart from a particular class of games, quantum correlations do not help in solving such games in general, in the sense that there exists a classical protocol (using shared randomness) whose probability of success is at least as large as the one of any protocol using quantum resources. Our analysis holds for both the worst case and the average case. Finally, we point out that the latter result highlights the relevance of adopting novel forms of decision, which is the main point of our ULD and UNLD classes presented in the first part.

Chapter 1

Distributed Computing

1.1 Modeling distributed systems

1.1.1 Definitions

Definition 1 *A distributed system is a collection of autonomous computing entities, sharing some resources, and cooperating in order to perform a global task.*

Computing entities The computing entities are also referred to as *processors* (in the case of computer networks or multiprocessor systems) or *nodes* (often used when the system is modeled by a graph). They can be mobile in some settings and may be referred to as *robots* as well. They are autonomous, running the same distributed algorithm (possibly depending on their unique identifier), without some central entity intervening. When not specified, agents have no information about the system. Nevertheless, the amount of knowledge they have may be augmented in some settings with an input, the number of nodes, the topology of the system, or any other information depending on the model.

Shared ressources In order to cooperate, processors need to share some resources. Here, two distinct families of models appear in the literature. First, in the *shared memory* models, processors perform several read/write operations on a single shared memory. Problems that are addressed in this case are often related to atomic commit-

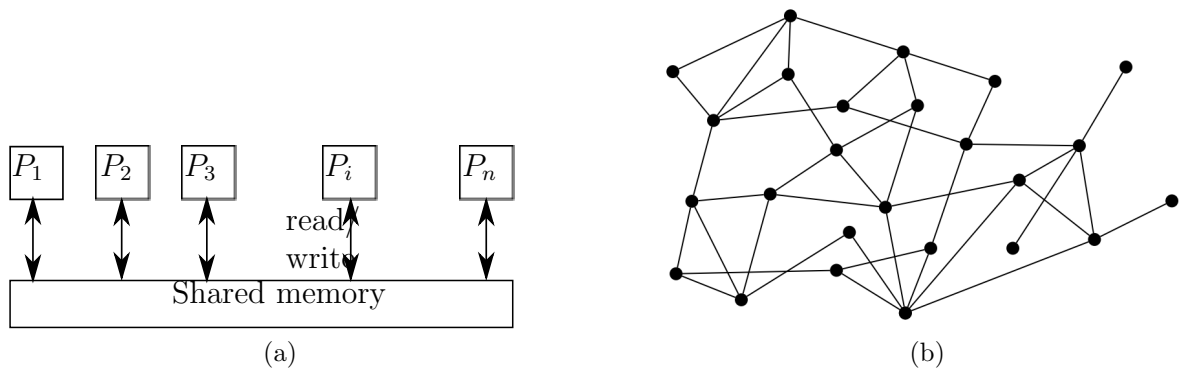


Figure 1-1: 1-1a represents the shared memory model where processors read and write on a single memory, whereas 1-1b represents the message passing model where processor has their own memory and communicate through communication links.

ments, agreements, transactional memory and so on. The second way to cooperate is to share information via communication links, hence the *message passing* models. Beyond this main dichotomy, processors my also share additional resources, such as a common clock for synchrony, randomness sources providing them with the same random bit string, or even an entangled quantum system of particles.

Global task Some examples of the so called global task are given below:

- Electing a leader among all the processors.
- Agreeing on a common output value.
- In the case of a networks, coloring the nodes such that non neighboring nodes have the same color.
- Broadcasting an information over the network.
- Sending a message from node i to node j.
- Answering the question *How many processors are there in the system?*
- Answering the question *Is the network a Tree?*

One can see from these examples that distributed tasks can be as various as computing a function at each node, construction tasks, decision problems, an so on. A formal definition of a global task is then quite difficult to establish because of

the very nature of distributed computing. We will see in the next chapter that we will focus on only one type of tasks in order to be able to compare the difficulty of distributed problems in a consistent framework.

1.1.2 What is so special about distributed computing?

Unlike sequential or centralized computing where a unique central entity linearly executes a given procedure, distributed settings rise many specific issues that need to be addressed. We list hereafter some of these issues.

Asynchrony Processors don't necessarily have a common clock and thus aren't *a priori* synchronized. This issue heavily impacts distributed computations and the question of how to implement consistent synchronous algorithms in asynchronous systems has been studied in, e.g., [7, 32, 80, 35, 9] and others.

Partial knowledge Whether it is due to constraint or to privacy concerns, agents often have only a partial knowledge of the system and/or of the input. It is the case in particular when the problem input is the system itself. For example, in the case of a network, processors can only gather information through communication links, i.e. from their neighborhood. However, they have to perform a task involving the whole network. Similarly, each processor is oblivious to the global state of the system. This issue makes it more complex in distributed computing to decide when to stop a computation.

Communication capacity Communication comes at a significant cost. It is indeed constrained by topological requirements (all nodes can't necessarily communicate with all others) and also by the capacity of communication links. We refer the reader to, e.g. [75] for insights in the $\mathcal{CONGEST}$ model that captures this issue and measures complexity in number and size of exchanged messages.

Symmetry Agents in distributed systems are equal, start from the same state, and run the same algorithm. This symmetry is one of the most challenging aspects of distributed computing and is at the core of many impossibility results ([57] and references therein). Many models assume a unique identification of agents and/or introduce randomness as a computational resource to achieve *symmetry breaking*.

Faults In the case of a failure of one component, distributed systems must be able to keep functioning. It is important to have models that capture the different types of failures that can occur: processor crash, communication link deletion or any byzantine behavior. See, e.g., [49, 81] for surveys on the design and analysis of fault-tolerant systems and recovery algorithms.

All these issues are hardly handled by a single model. This is why distributed computing is characterized by the variety of its models. Let us mention the $\mathcal{LOCAL}$ model, WAIT-FREE, MOBILE or $\mathcal{CONGEST}$ as examples. Each one of these models aims to capture one or some of the aspects stated above, and thus has a specific complexity measure, making it difficult to unify a proper complexity theory for distributed computing. In the rest of our work, our focus will mainly be on the $\mathcal{LOCAL}$ model. This model captures the essence of locality and constant time algorithms. Beyond the obvious advantage of scalability, locality is a good feature for the design of fault-tolerant protocols. Indeed, systems that run with local algorithms are resilient to errors as they would only propagate on an area of restricted size. Besides, in this model the cost of individual computations is free allowing us to focus purely on the value of local information and to study its impact on global knowledge.

1.2 The $\mathcal{LOCAL}$ model

1.2.1 Terminology

The $\mathcal{LOCAL}$ model is a standard network computing model for message-passing systems. It has been introduced in [72] and formalized in [75]. Formally, the system is modeled by an undirected connected graph $G = (V, E)$, where the set of nodes V represents the processors and the set of edges E represents the communication links. We recall that the distance $d(u, v)$ between two nodes u and v is the minimal number of edges between the two nodes, and that the *neighborhood* or *ball* of radius t of a node v, $B_G(v, t)$, is the subgraph of G formed by all the nodes at distance $d \leq t$ from v.

In this model, nodes are given unique Ids, and the algorithms should work properly for every possible identity assignment. The $deg(v)$ edges incident to node

$v \in V(G)$ are locally labeled by $deg(v)$ distinct labels, called port numbers. We assume that each node has the knowledge of its degree $\deg(v)$, the numbering of its ports, and its own identity Id. Initially, every node $v \in V$ has an input $\mathbf{x}_i$ and must output $out(v)$[1]. Nodes are woken up simultaneously and run the same algorithm $\mathcal{A}$ (possibly depending on the Id).

Computation proceeds in *fault-free synchronous rounds*. Let t be the number of computation rounds. At each round $i = 1, \ldots, t$, the following steps are performed in the following order:

1. Every node performs a local computation.
2. Every node v sends messages along the edges $1, \ldots, \deg(v)$.
3. Every node v receives a messages from edges $1, \ldots, \deg(v)$.

Eventually, after t rounds, every node $v \in V$ performs a final computation and announces $out(v)$ as an output. Due to the synchrony assumption, this model is equivalent (see, e.g. [72]) to the protocol where, in a single round:

1. Every node performs a local computation.
2. Every node v collects all possible information from nodes in $B_G(v, t)$, its neighborhood of radius t.
3. Every node v outputs $out(v)$ based on this information.

In addition, the model states that every processor exchanges messages of unlimited size with its neighbors, and performs arbitrary computations on its data without limitations on the computational capacity of the nodes.

Let D be the diameter of G, i.e. the largest distance between any two nodes. It is clear from the previous assumption that if $t = O(D)$ then any (Turing Machine computable) task can be handled by the processors since every processor can gather all available information about the network in t rounds, and designate a leader (for example, the node with the smallest Id) that will handle the task in a sequential centralized manner. That is why the study of the $\mathcal{LOCAL}$ model focuses on algorithms that run in $t = o(n)$, i.e. where outputs are computed based only on the

[1]depending on the problem and the notations, the output may also be noted b_v our $\mathbf{y}_v$ in the rest of this text.

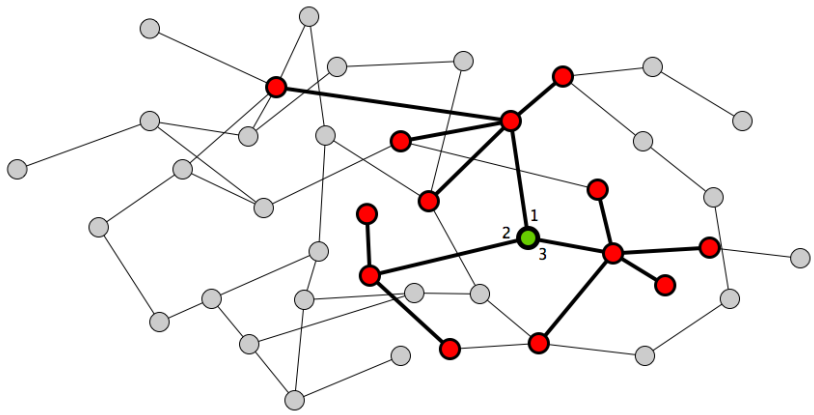


Figure 1-2: In this example, the green node performs its computation based only on the information it gathers from its neighborhood of radius $t = 2$, i.e. the red nodes. The same goes for all the other nodes.

information available in a relatively restrained local area. In our work, we are interested in algorithms (see the example on Figure 1-2) that run in time $t = O(1)$, i.e., local algorithms. Note that in many papers, the term "local" is used in a less strict manner, designating any algorithm running in the $\mathcal{LOCAL}$ model. We will adopt the following terminology

Definition 2 (Local algorithm) *A distributed algorithm $\mathcal{A}$ is local iff it runs in a constant number of rounds.*

1.2.2 Construction problems

Locality issues have been thoroughly studied in the literature, via the analysis of various *construction problems*. These problems require the output of each node to be part of a distributed solution that responds to particular specifications. For example, the Minimum Spanning Tree problem (MST) aims to construct a minimum weighted spanning tree in a weighted graph. A correct solution requires the output $out(v)$ of every node v to point to the edge that links to its parent in a tree T, such that T is of minimum weight. Other examples include coloring, maximal independent set (MIS), matching, dominating set, spanners, etc.

The question of what can be computed in a constant number of communication rounds was actually posed in the seminal work of Naor and Stockmeyer [72]. Con-

sidering only bounded degree networks, they introduce the class of *Locally Checkable Labeling* problems (LCL), which can be constructed and verified by local algorithms. One of their most celebrated results states that in order to solve LCLs, the actual numerical values of node identifiers are not necessary, but only their relative order needs to be known to the algorithm. They also present several nontrivial upper bounds on the running time of such algorithms.

We will not give an exhaustive survey of results related to such problems, and we refer the reader to [65] and [78] for more detailed an complete surveys. We also refer to the book [75] for an excellent introduction to local computing, providing pointers to the most relevant techniques for solving construction problems.

For the sake of illustration, we mention a typical example of construction problems, namely the Vertex Coloring problem, as we will go back to this problem later in our work. Solving the (distributed) Vertex Coloring problem entails assigning colors in $\{1, \cdots, c\}$ to the nodes of a graph G such that any two neighboring nodes have different colors. Properly coloring a graph has multiple applications, as it is used in modeling various scheduling problems, resource allocation problems, interference avoidance in wireless networks, etc. Cole and Vishkin [27] give an algorithm that uses the values of node identifiers to 3-color a ring of size n in a time[2] $O(\log^* n)$. Linial [68] showed that this bound is optimal: any algorithm 3-coloring the ring requires communication at distance $\Omega(\log^* n)$. For general graphs of maximum degree Δ, [68] also shows that $O(\Delta^2)$-coloring can be achieved within $O(\log^* n)$ communication distance. For $O(\Delta+1)$-coloring, the best known upper bound is in $O(2^{\sqrt{(\log n)}})$([10, 73]). There's a vast body of literature regarding distributed vertex coloring, and we refer to [12] for more insights on the matter.

It is worth mentioning that few construction problems are actually achievable locally, i.e. in a constant number of rounds. Many problems have indeed an inherently non-local nature, as is it the case for the Minimum Spanning Tree construction problem for example [78]. Kuhn et al.[66] also provide several impossibility results regarding approximation. For instance, minimum vertex cover, and minimum dominating set cannot be well locally approximated within a constant factor in general

[2]The $\log^*$ function (also called iterated logarithm) is defined as the number of iterations of the log function that needs to be applied to a number in order to obtain a result smaller than 1. It is an extremely slowly goring function.

graphs with no bound on the maximum degree.

Let us finally point out that a variant of the $\mathcal{LOCAL}$ model involves the use of oracles that provide extra information to nodes. This was in particular studied in the framework of *local computation with advice*. In this framework, MST construction was studied in [43], 3-coloring of cycles in [39], and broadcast and wake up in [42]. Finally, in [63] it is shown that, in the context of local computation, access to the oracle providing the number of nodes is not required for solving efficiently several central problems (e.g., $O(\Delta)$-coloring, MIS, etc.), while previous algorithms in the literature explicitly or implicitly assumed the use of this oracle.

1.3 Conclusion

In this introductory chapter we laid the formal ground for our work. We raised the issue of the variety of models in distributed computing, which is one of the obstacles on the way of the development of a proper complexity theory in this setting. We presented the model of our main interest, the $\mathcal{LOCAL}$ model, which will allow us to focus on the issues of partial knowledge and locality related constraints. Moreover, the focus on construction problems, even though general, is in a certain way too large and needs refining in order to compare the difficulty of problems in a consistent manner. The next chapter will introduce *decision problems* as an ideal candidate for the study of local distributed complexity.

Part I

Decision and verification in the $\mathcal{LOCAL}$ model

Chapter 2

Local Decision

2.1 Introduction

We showed in the previous chapter that the study of distributed algorithms has long been driven by construction problems. Despite its numerous results, this approach does not suit well the definition of proper complexity classes and thus the development of a distributed complexity theory. The main reason is that there is no common ground on which one can consistently compare two *construction problems* such as a broadcasting an information on a network versus the distributed computation of a function for example.

In the sequential setting on the other hand, the focus on *decision problems* has proven to be useful when it has come to building a consistent complexity theory. We recall that a (sequential) decision problem is defined as the question to determine whether, given a word ω and a language $\mathcal{L}$, if ω is in $\mathcal{L}$ or not. The complexity class P is then defined to be the set of all decision problems whose solution can be determined in a polynomial time by a deterministic Turing Machine (TM). Similarly, L is defined to be the class of decision problems that can be solved using a logarithmic amount of memory space.

This is why, inspired by sequential computing, we focus on *distributed decision problems* in the $\mathcal{LOCAL}$ model.

Our contributions In this chapter, we revisit distributed decision problems by introducing the class ULD of Universally[1] Locally Decidable languages. For this purpose, we start by describing in Section 2.2 the framework of local decision. We extensively discuss the role of the *interpretation function* as a key component in this framework, and highlight several arguments in favor of extending the interpretation to yet unusual operators. In Section 2.3, we formally define the class ULD, and we show that for a large class of graphs, solving decision problems doesn't require the actual values of node Ids, but only their relative order. Finally, we give some preliminary results regarding ULD. For instance, we show that it strictly contains the class LD, but does not contain all distributed languages.

Related Work Recently, several results were established concerning decision problems in distributed computing. For example, [29] and [59] study specific decision problems in the $\mathcal{CONGEST}$ model. Specifically, tight bounds are established in [59] for the time and message complexities of the problem of deciding whether a given subgraph is an MST of the network, and time lower bounds for many other subgraph-decision problems (e.g., spanning tree, connectivity) are established in [29]. Decision problems have recently received attention in the asynchronous setting too, in the framework of wait-free computing [47]. Similarly, decision problem have also received attention in the context of computing with mobile agents [46].

Regarding the decision in the $\mathcal{LOCAL}$ model, [45] define $\mathrm{LD}(t)$ as the class of decision problems that can be solved in t communication rounds with an AND (conjunction) interpretation of the outputs. LD is defined to be $\cup_{t\geq 0}\mathrm{LD}(t)$. We'll see that our main contribution of this chapter, the class ULD, is an extension of the class LD. [45] also study the impact of randomization on local decision, by defining the class $\mathrm{BPLD}(t,p,q)$, containing all languages for which there exists a randomized algorithm that runs in t rounds, that accepts correct instances with probability at least p, and rejects incorrect ones with probability at least q. They show that, restricted to hereditary languages, if $p^2+q>1$, then $\mathrm{BPLD}(O(t),p,q)=\mathrm{LD}(t)$.

The relaxation we introduce in our work with the *universal* interpretation is somehow close to the model used in [17]. The authors consider the setting where nodes communicate in the $\mathcal{CONGEST}$ model, and then send a message to a central

[1] the U in ULD may subsequently equally refer to Universal or Unrestricted.

entity, the *referee*, that must determine structural properties of the network based on the information it collects from all the nodes. The main differences with our setting are that nodes can only exchange messages of size $O(\log n)$, and that, on the other hand, they are allowed to send a message of size $O(\log n)$ to the referee, whereas in our setting, nodes can only output constant size outputs that will be handled by the interpretation function.

The impact of processors identifiers on the distributed decision has been thoroughly studied as well. In [41], the authors give several conditions under which identifiers are not needed, define the *Id-oblivious model* along with the corresponding class LD^*, and conjecture that identifiers are not needed in any decision problem, i.e., that $\mathrm{LD}^* = LD$. However, [40] disprove this conjecture and show that, under some critical assumptions on the computation model, there are graph properties for which local decision depends on node identification.

2.2 Decision problems

2.2.1 Framework

We start by reminding some definitions introduced by [45].

Definition 3 (Configuration) *Let $G = (V, E)$ be a connected undirected graph and $\mathbf{x} = \{\mathbf{x}_u, u \in V(G)\}$ denote the set of inputs given to the nodes (node u receives the binary string $\mathbf{x}_u$ as input). Such a pair $(G, \mathbf{x})$ is called a* configuration.
Note that in some cases the input may be empty, that is $\forall u, \mathbf{x}_u = \epsilon$, the empty binary string.

Definition 4 (Language) *A distributed language is a (TM-decidable) collection $\mathcal{L}$ of configurations $(G, \mathbf{x})$.*

Typical examples of distributed languages are the following:

- `IsColored` $= \{(G, \mathbf{x})$ s.t. $\forall u \in V(G), \forall v \in N(u), \mathbf{x}_u \neq \mathbf{x}_v\}$, where $N(u)$ denotes the neighborhood of u, that is, all nodes at distance exactly 1 from u.
- `Consensus` $= \{(G, (\mathbf{x}_1, \mathbf{x}_2))$ s.t. $\exists u \in V(G), \forall v \in V(G), \mathbf{x}_2(v) = \mathbf{x}_1(u)\}$. This

language consists of all instances in which all nodes agree on the value proposed by one of them.

- `Leader` $= \{(G, \mathbf{x})$ s.t. $\forall u, \mathbf{x}_u \in \{0, 1\}$ and $\sum_{u \in V(G)} \mathbf{x}_u = 1\}$ is the language formed by graphs where exactly one node is selected.
- `MIS` $= \{(G, \mathbf{x})$ s.t. $S = \{u \in V(G) | \mathbf{x}_u = 1\}$ forms a maximal independent set$\}$
- `Tree` $= \{(G, \epsilon)$ s.t. G is a tree$\}$

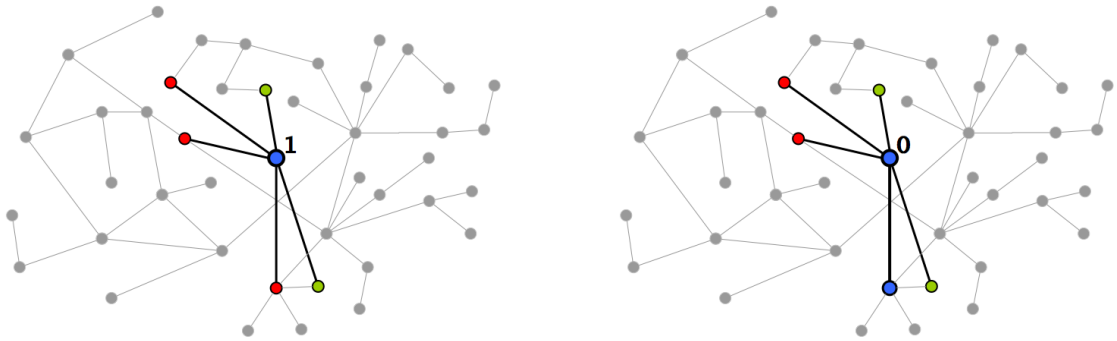
Definition 5 (Decision problem) *Given a configuration* $(G, \mathbf{x})$ *and a language* $\mathcal{L}$*, a* decision problem *consists in answering the yes-or-no question* "Is $(G, \mathbf{x})$ in $\mathcal{L}$?".

The latter definition is analogous the the definition of decision problems in the sequential setting. However, in our setting, the input $\mathbf{x}$ is distributed, and algorithms are distributed and local. Each node u performs the same algorithm $\mathcal{A}$ (possibly depending on its own identity) and outputs some value b_u in a constant number of rounds t, that is, after inspecting all nodes in the ball of radius t. This includes the structure of that ball, and the input values given to the nodes in this ball. However, this is not sufficient. Indeed, while in sequential computing one central entity decides by given one single output, distributed decision relies on an *interpretation function* $\mathcal{I}$ that translates the outputs of all the nodes to a single answer. Formally, $\mathcal{I}$ is a mapping from all possible output multisets to $\{\mathbf{yes}, \mathbf{no}\}$. We underline here that the interpreter acts on multisets, that is, it only depends on undistinguishable values of the outputs and their multiplicity.

In the setting defined by [45], this interpretation function is simply the conjunctive operator $\bigwedge_u b_u$ on 1-bit-per-node outputs (it is the boolean AND operator) . That is, if the input configuration is a valid instance of the language, then all nodes must output **yes**, and if it is not a valid one, then at least one processor will output **no**. The authors define the class LD as the class of language locally decidable with this conjunctive operator.

It is in general far from easy to tell if a given language is locally decidable or not. `IsColored` (see figure 2-1) however is a trivial example of a locally decidable language[2]. Indeed, given a graph colored with c different colors, i.e. a graph where

[2]Note that `IsColored` is the decision problem corresponding to the (non local) construction problem `Vertex Coloring` introduced in the previous chapter.



(a) Locally valid coloring (b) Locally forbidden coloring

Figure 2-1: In 1 round of computation, the local algorithm collects the colors of the neighbors. If the coloring is locally valid (2-1a), the node outputs **yes** (or **1**). Otherwise(in 2-1b one neighbor is also blue), the node outputs **no**. The conjunction of all the outputs equals **no** if and only if there are at least two neighboring nodes of the same color.

the input $\mathbf{x}$ is a vector of elements in $\{1, \cdots, c\}$, there is a simple local distributed algorithm that decides if G is in `IsColored`. In 1 round of computation, each node v collects the respective colors of all of its neighbors and compares it to its own colors $i \in \{1, \cdots, c\}$. If they all have a color different from i, the node outputs **yes**, otherwise, that is if at least one of v's neighbors has the color i, the node outputs **no**. The conjunctive operator AND applied to all the outputs equals **yes** if and only if no two neighboring nodes have the same color, i.e. the graph is a valid instance of `IsColored`.

Identifiers Due to the difficulty of symmetry-breaking, very few languages can be decided in anonymous graphs (see [57] and references therein, in particular [4]). Usually, nodes are given identifiers to overcome many impossibilities. An identification *Id* is an injective mapping from the set $V(G)$ to the set of integers. Nodes know their Ids and the Ids in their ball of radius t, but have in general no further information about Ids of other nodes. Since this identification isn't a part of the decision problem input, an algorithm locally deciding a language must be correct for any identification *Id* of the nodes. We will later show that in our framework and for the class of graphs of bounded maximal degree, if a language is locally decidable then it can be decided by an algorithm using only the relative order of the Ids and not their actual values.

2.2.2 Role of the Interpreter $\mathcal{I}$

The use of the conjunctive operator for the interpretation of the outputs is conceptually elegant, and is well motivated by practical applications. For instance, the output $b_u = 0$ at node u can be interpreted as node u *raising an alarm*. This alarm can be used by a central entity collecting data. It can also be used in a distributed manner. For instance, in the framework of *self-stabilzation*, the alarm at node u may correspond to the detection of an invalid state of the system, and yield the launch of a recovery procedure [11, 58]. Also, since the conjunctive operator is idempotent, commutative, and associative, it is easy to conceive a gossip procedure enabling all nodes to become aware of the global decision, by flooding in diameter rounds, where each round involves exchanging a single bit on each link (see, e.g., [29] for decision and verification in the $\mathcal{CONGEST}$ model).

Nevertheless, restricting ourselves to this interpretation is not a law carved in stone. We give hereafter several arguments against restricting the distributed decision setting to this specific operator.

Simultaneous decisions. The restriction to the AND operator does not fit with elementary algebraic operations on sets. Typically, one may be able to distributively decide locally two distributed properties $\mathcal{P}$ and $\mathcal{P}'$, and yet be unable to distributively decide $\mathcal{P} \vee \mathcal{P}'$. For instance, using the conjunctive operator, one cannot decide locally whether nodes are properly k-colored if the set of k colors is not specified (this holds even if this set of colors is specified as being either $\{\text{green}, \text{orange}, \text{red}\}$ or $\{\text{green}, \text{orange}, \text{blue}\}$).

Majority voting. Voting to the majority is a very common group decision making process. It is a natural and simple example of a distributed decision protocol where the global decision does not rely on an AND-interpretation of individual outputs (except for veto systems where, on the contrary, it is exactly equivalent to an AND-interpretation). Majority voting has thoroughly been studied from voting-theory and game-theoretic perspective [70, 22, 71, 19]. From a local distributed decision point of view, one can imagine a framework similar to [45] where decision is taken such that a configuration $(G, \mathbf{x})$ is accepted if and only if at least half the nodes output **yes**.

Sensor networks. Sensor networks are systems formed by spatially distributed agents whose role is to monitor an environment. Such networks can have industrial as well as civil and military applications that range from security surveillance to seismic monitoring, machine monitoring, fault detection, etc. In order to be exploited and treated by the user, the collected data is eventually gathered at a single control center or base unit. The design of many-to-one communication algorithms for data gathering and data fusion [2] has thus been a growing area of interest through the last years. Such requirements reinforce our interest in unrestricted interpretation functions for distributed decision. Indeed, should an event occur in the sensors environment, it may require more than one sensor to detect it. Alternatively, consider un error-prone network where events can be detected locally, but to avoid false alarms, an alarm is raised if and only if several consistent anomalies are reported. One can think of many other interpretations, however complex, of the gathered data. In other words, it is reasonable to consider that the AND-interpretation is not sufficient for the setting of sensor networks. Furthermore, note that since technological advances have enabled the development of inexpensive low-power micro-sensors, it has been a concern to design energy efficient and capacity efficient algorithms for such networks (see, e.g., [79, 56, 67]). The framework we present and develop further in this chapter suits well such constraints, as it only allows nodes to give bounded-size outputs.

Property testing. In the same spirit of minimizing resources, *property testing* [53] is aiming at identifying languages that can be decided without accessing the whole instance. In particular, for any fixed graph class $\mathcal{C}$, property testing on graphs [54] aims at designing algorithms that can decide whether or not any given n-node graph G belongs to $\mathcal{C}$, by querying only $o(n)$ (random) nodes of the graph. In essence, the result of querying a node u is a value $out(u)$ such as, e.g., the degree of u. The decision algorithm acts and decides depending on the set of values collected so far. There are no restrictions imposed to this algorithm, except that one aims at designing algorithms deciding in polynomial time.

We refer the interested reader to [53] for recent research and surveys about property testing.

Probabilistic decision. Last but not least, recent results in the framework of distributed local decision [44] reveal that restricting ourselves to the conjunction operator prevents us from "boosting" probabilistic decisions. That is, using the conjunction operator, and as opposed to, say, languages in BPP, there are classes of distributed languages that can be decided distributively with a fixed probabilistic guarantee, but which cannot be decided distributively with better guarantees. In fact, there are computational models (e.g., distributed quantum computing) in which the exclusive-disjunction operator is known to be far more practical and efficient than the conjunctive operator (see [5] as well as Part II of this work, and the references therein).

2.3 Unrestricted Local Decision (ULD)

To sum up the above discussion, it seems that while using the conjunctive operator in the framework of local distributed decision is well grounded for some settings, there are no reasons to stick to this specific operator in general. In this section, we revisit local decision by introducing the class ULD of Universally Locally Decidable languages and give some results regarding this class.

2.3.1 Definitions

As in classical local distributed decision, the interpretation is taken over all outputs. However, as in property testing and sensor networks, we allow any form of interpretation. Obviously, all distributed languages would be decidable in a single round in this setting if one did not impose restrictions on the values output by the nodes. We restrict every node to output a *constant* number k of bits. Hence, our framework is the extension of the model in [44, 45, 55, 72] where:

1. every node outputs a number of bits bounded by a language-specific constant (rather than only one bit), and
2. the interpretation of these outputs is allowed to be any binary valued function $\mathcal{I}$ whose arguments are the unordered multi-sets of outputs of all nodes (rather than only the conjunction operator applied to these outputs).

Note that, using the interpretation function $\mathcal{I}$, an instance is accepted or rejected on the basis of the number of outputs of each type, regardless by which node a given output is produced.

For a non-negative integer t, we define the class ULD(t) (for *Unrestricted Local Decision*) as the class of distributed languages that can be decided in t communication rounds in the $\mathcal{LOCAL}$ model, where decision is taken according to the rules specified above.

Definition 6 (ULD(t)) *A language $\mathcal{L}$ is in* ULD(t) *iff there exists a local algorithm $\mathcal{A}$ running in time t, and an interpreter $\mathcal{I}$ s.t. $\forall$ configurations with Ids $(G, \mathbf{x}, Id)$:*

$$\begin{array}{lcl} & (G,\mathbf{x}) \in \mathcal{L} & \Rightarrow \quad \mathcal{I}(out_{\mathcal{A}}(G,\mathbf{x},Id)) = \mathbf{yes} \\ \textit{and} & (G,\mathbf{x}) \notin \mathcal{L} & \Rightarrow \quad \mathcal{I}(out_{\mathcal{A}}(G,\mathbf{x},Id)) = \mathbf{no} \end{array}$$

We now define the class of main interest for the purpose of our work:

Definition 7 (ULD) ULD $= \cup_{t \geq 0}$ULD(t)

It is the class of all languages decidable in constant time, i.e. locally.

2.3.2 Role of identifiers

In this section, we restrict ourselves to the class of graphs of bounded maximal degree and give a result related to the role of identification in our framework. Although identification is necessary for symmetry-breaking, we show that for languages in ULD, it is always possible to design an algorithm that doesn't use the actual values of node IDs, but only their relative order. Our result is similar to the one by Naor and Stockmeyer [72], but the latter cannot be applied in our context because our instances are not necessary in the class LCL of so-called *locally checkable languages*. Nevertheless, we were able to proceed with an appropriate reduction by using the infinite version of Ramsey Theorem.

Definition 8 (Order-equivalent([72])) *Two identifications Id and Id$'$ are order-equivalent iff*

$$\forall u, v \quad Id(u) < Id(v) \Rightarrow Id'(u) < Id'(v)$$

Definition 9 (Order-invariant([72])) *An algorithm $\mathcal{A}$ is order-equivalent iff*

$$\forall u, Id, Id' \qquad Id \text{ and } Id \text{ are order-equivalent } \Rightarrow out_{\mathcal{A}}(u, Id) = out_{\mathcal{A}}(u, Id')$$

Theorem 1 *For the class of graphs of bounded maximal degree Δ, if there exists a local algorithm $\mathcal{A}$ that decides a* ULD *language $\mathcal{L}$, then there exists an* order-invariant *local algorithm $\mathcal{A}'$ that decides $\mathcal{L}$.*

Proof. For any set X, and any positive integer r, we denote by $X^{(r)}$ the set of all subsets of X with size exactly r. Let X be an countably infinite set, let r and s be two positive integers, and let $c : X^{(r)} \to [s]$ be a "coloring" of each set in $X^{(r)}$ by an integer in $[s] = \{1, \ldots, s\}$. Recall that (the infinite version of) Ramsey's Theorem states that 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). We make use of this theorem as follows.

Let us consider the collection $\mathcal{B}$ 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 Δ. There is a finite number β of pairwise non-isomorphic balls in $\mathcal{B}$. We enumerate these balls from 1 to β, and let n_i be the number of vertices in the ith ball, for $i = 1, \ldots, \beta$. For every i, the vertices of the ith ball can be ordered in $n_i!$ different manners, corresponding to the $n_i!$ permutations in Σ_{n_i}. We consider the $N = \sum_{i=1}^{\beta} n_i!$ ordered balls $B_{i,\sigma}$, for $i = 1, \ldots, \beta$, and $\sigma \in \Sigma_{n_i}$, and we enumerate these ordered balls as $L_1, \ldots, L_N$ in an arbitrary order. Using these balls, we define an infinite set U 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 $\mathcal{A}$ at the center of L_i is the same for all possible identity assignments to the nodes in L_i with values in X_i, and respecting the ordering of the nodes in L_i. We define the coloring $c : X_j^{(r)} \to [2^k]$ where r is the number of nodes in L_{j+1}, as follows: for each r-element set $I \in X_j^{(r)}$, assign r pairwise distinct identities to the nodes of L_{j+1} using the r values in I, and respecting the order of the nodes in L_{j+1}. Then, define $c(I)$ as the output of the algorithm $\mathcal{A}$ at the center of L_{j+1} under this identity assignment to the nodes of L_{j+1}. By Ramsey's Theorem, there exists an infinite set $Y_j \subset X_j$ such that all r-element set $I \in Y_j^{(r)}$ are given the same color. We set $X_{j+1} = Y_j$. We proceed that way until we exhaust all balls L_i, $i = 1, \ldots, N$, and

we set $U = X_N$.

By construction, the set U 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 U 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 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 U, in the order specified by σ, and output what would have outputted $\mathcal{A}$ if nodes were given these identities. Note that $\mathcal{A}'$ is well defined, as nodes can be provided with the $\nu = \sum_{i=0}^{t} d^i$ smallest integers in the set U. (They do not need to know the entire set U, but only a finite number of values in U). Note also that, by construction, $\mathcal{A}'$ is order-invariant.

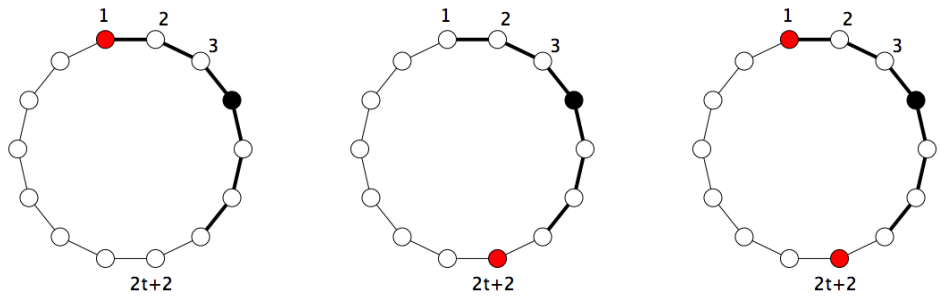
To establish that $\mathcal{A}'$ is correct, let us consider some n-node input graph G, with nodes provided with pairwise distinct identities in U, and let $out = \{out(v), v \in V(G)\}$ be the output of $\mathcal{A}$ in this context. This multi-set is precisely the multi-set outputted by $\mathcal{A}'$ in G. Indeed, every node v relabels its radius-t ball with identities in U, respecting the order induced by the original identities in U, and U 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 U. Hence, since $\mathcal{A}$ is correct, it follows that $\mathcal{A}'$ is correct as well. $\square$

2.3.3 Preliminary results

We give here some preliminary results related to the class ULD. Further results, including separation results with other classes, are given in the next chapter.

Fact 1 LD $\subsetneq$ ULD.

Proof. Recall that LD is the class of locally decidable languages by the setting defined in [45], i.e. by using the conjunctive operator as an interpreter. By definition, LD $\subseteq$ ULD.



(a) G_1 (b) G_2 (c) H

Figure 2-2: Illustration of `Leader` $\notin$ LD for $t = 3$. Red nodes are leaders. The black node has the same view in H as in G_1. Since $\mathcal{A}$ is correct for G_1, the black node outputs **yes** in H. The same goes for all other nodes.

Recall now the language

$$\texttt{Leader} = \{(G, \mathbf{x}) \text{ s.t. } \forall u \in V(G), \mathbf{x}_u \in \{0,1\}, \text{ and } \sum_{u \in V(G)} \mathbf{x}_u = 1\}.$$

We have `Leader` $\notin$ LD. Indeed, suppose for the sake of contradiction that there is an algorithm $\mathcal{A}$ running in time t deciding `Leader` with the conjunctive operator. Consider two cycles G_1 and G_2, both of size $4t + 2$. Give the nodes identifiers form 1 to $4t + 2$ in the same ordering in both cycles. Let node 1 be the leader in graph G_1 and node $2t + 2$ the leader in G_2. With such inputs, G_1 and G_2 are both valid instances of `Leader` and $\mathcal{A}$ outputs **yes** for all nodes in both graphs. Consider now the cycle H of size $4t + 2$, with the same identification as the previous graphs. Let both nodes 1 and $2t + 1$ have input 1 (two leaders). Each node in H sees the same ball of radius t as in one of the two other graphs G_1 or G_2, with the same structure, same input and same identifiers. Each node thus gives the same output as in G_1 or G_2, which contradicts the fact that $H \notin$ `Leader`.

To establish that `Leader` $\in$ ULD, we describe a local distributed algorithm enabling each node to output a constant number of bits, with the associated interpretation. The algorithm performs in zero rounds: every node u simply returns the single bit $b_u = \mathbf{x}_u$. 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 $\mathcal{I} = \bigvee_{i=1}^{n} \left(b_i \wedge \bigwedge_{j \neq i} \overline{b_j} \right)$ 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. $\square$

The following result states that ULD doesn't contain all languages. To establish that, we consider the following language

$$\texttt{Biconnected} = \{(G, \mathbf{x}) \text{ s.t. } \forall u, v \in V, \text{there are two disjoint paths between } u \text{ and } v\}.$$

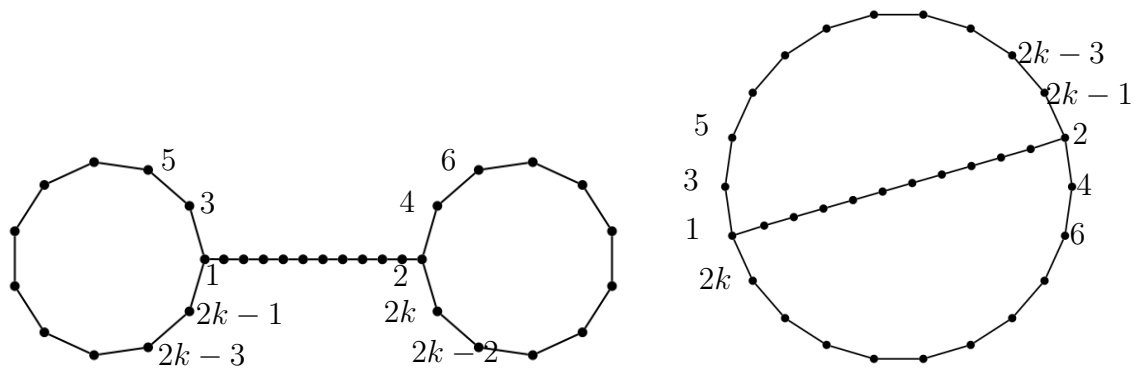
Biconnectivity is a desirable feature for many communication networks where connectivity robustness to vertex removal is required. We show that it is not a locally decidable property, even with unrestricted interpretation.

Fact 2 `Biconnected` $\notin$ ULD

Proof. Consider the graphs G_1 and G_2 of figure 2-3. G_1 is formed by 2 cycles of size k each, connected by a path of length k. Let u and v denote the two 3-degree nodes. Give to u the Id 1, and to all nodes that are in the same cycle as u the odd Ids in $[1, \cdots, 2k]$ in a clockwise ordering. Similarly, give to node v the Id 2, and let all the nodes in the same cycle as v have even Ids in $[2, \cdots, 2k]$ in the same ordering. Nodes on the path have Ids in $[2k+1, 3k]$ in a growing order, from u to v. Now, consider G_2, a graph of size $3k$, formed by a cycle of size $2k$ and a path of length k, where the two 3-degree nodes haves Ids 1 and 2. Consider the three segments between the 3-degree nodes and give to nodes in the first segment (resp. second, third) Ids from odd integer in $[1, \cdots, 2k]$ (reps. even integers in $[2, \cdots, 2k]$, in $[2k+1, 3k]$).

Assume that there exists an algorithm $\mathcal{A}$ that decides `Biconnected`. By Theorem 1, there exists an order-invariant algorithm $\mathcal{A}'$ that also decides `Biconnected`. Provided that $k > 2t$, one can check that for every node i, $B_{G_1}(i, t)$ and $B_{G_2}(i, t)$ are order equivalent. As a consequence, $\mathcal{A}'$ produces the same output for i in G_1 and G_2, for all i in $[1, \cdots, 3k]$. However, G_2 is a biconnected graph, while G_1 is not, as the removal of any node of the central path disconnects the graph. This contradicts the assumption of local decidability of `Biconnected`. $\square$

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



(a) G_1 (b) G_2

Figure 2-3: G_2 is biconnected while G_1 is not. However, to any node, the graphs look locally the same.

Chapter 3

Distributed verification

3.1 A flavor of non-determinism

In sequential computing, the ability of verification is tightly related to the notion of non-determinism, as NP and NL, the non-deterministic versions of P and L, can be interpreted as the classes of languages *verifiable* using polynomial computation time and logarithmic memory space, respectively, with small certificates.

In this chapter, we introduce the class of locally verifiable languages UNLD, the non-deterministic version of ULD. In distributed verification, every node $u \in V(G)$ is given a *certificate* $\mathbf{y}_u$, in addition to the input $\mathbf{x}_u$. Each certificate is an arbitrary binary string. A distributed language $\mathcal{L}$ is locally verifiable if there exists a pair $(\mathcal{A}, \mathcal{I})$, where $\mathcal{A}$ is a distributed algorithm performing in a constant number of rounds, and $\mathcal{I}$ is an interpretation of the outputs produced by $\mathcal{A}$ at all nodes, such that the following holds:

- if $(G, \mathbf{x}) \in \mathcal{L}$ then there exists a collection of certificates $\mathbf{y} = \{\mathbf{y}_u, u \in V(G)\}$ satisfying that $\mathcal{A}$ running in G, with the pair $(\mathbf{x}_u, \mathbf{y}_u)$ given to each node u, returns $out(u)$ at every node u such that $\mathcal{I}$ accepts the multi-set $\{out(u), u \in V(G)\}$;
- if $(G, \mathbf{x}) \notin \mathcal{L}$ then for any collection of certificates $\mathbf{y} = \{\mathbf{y}_u, u \in V(G)\}$, $\mathcal{A}$ running in G, with the pair $(\mathbf{x}_u, \mathbf{y}_u)$ given to each node u, returns $out(u)$ at every node u such that $\mathcal{I}$ rejects the multi-set $\{out(u), u \in V(G)\}$.

For a non-negative integer t, we define the class UNLD(t) (for *Unrestricted Nondeterministic Local Decision*) as the class of distributed languages that can be verified in t communication rounds in the $\mathcal{LOCAL}$ model, where the verification is performed according to the rules specified above. We then define our second class of interest: $\text{UNLD} = \cup_{t \geq 0} \text{UNLD}(t)$.

Observe again that, for both decision and verification, the global outcome should not depend on the identities assigned to the nodes. In particular, the certificate $\mathbf{y}$ for a legal instance $(G, \mathbf{x})$, i.e., for an instance $(G, \mathbf{x}) \in \mathcal{L}$, enabling the interpretation to accept $(G, \mathbf{x})$ should not depend on the identity assignment to the nodes. This is in accordance to distributed verification, as studied in [44, 45], but should not be mixed up with *proof-labeling schemes* [55, 62] in which the certificates can possibly depend on the identity assignment. The theory of proof labeling schemes [55, 60, 62] was designed to tackle the issue of locally verifying (with the aid of a "proof", i.e., a certificate, at each node) solutions to problems that cannot be decided locally. Investigations in this framework mostly focus on the minimum size of the certificate necessary so that verification can be performed in a single round [55, 60, 62], or in t rounds [61]. Hence, the model of proof labeling schemes has some resemblance to our definition of the class UNLD. The notion of proof labeling schemes also has interesting similarities with the notions of local detection [1], local checking [8], or silent stabilization [33], which were introduced in the context of self-stabilization.

Our results

We first establish a set of classification and separation results, by placing various languages in their appropriate decision and verification classes. These results are summarized in Figure 3-1, where the classes LD and NLD are the classes of locally decidable languages, and of non-deterministic locally decidable languages, respectively, defined in [45]. These classes can alternatively be defined as the restriction of ULD and UNLD 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.

We then prove that, in our universal decision and verification model, as opposed to classical distributed decision, *all* distributed languages can be *verified*. More specifically, all distributed languages on n-node networks with k-bit input per node can

be verified using certificates of $O(n^2 + kn)$ bits, by having each node inspecting its neighborhood at distance 1 only, with just 1-bit-per-node outputs. Hence, in other words, UNLD = All. This result is essentially obtained by proving that the problem `Cover`, known to be a "hardest" local decision problem up to local reduction [45], is in UNLD. (Formally, `Cover` is BPNLD-complete[1] [45]).

The above upper bound on the certificate size enabling to put any language in UNLD is tight. Indeed, we prove that there are languages which require $\Omega(n^2 + kn)$ bits to be verified, even if the nodes are allowed to perform an arbitrarily large number of communication rounds, and even if each node can output an arbitrarily large number of bits.

From the fact that all distributed languages can be *verified*, it results that, as for *proof-labeling scheme*, one major issue in our setting is minimizing the size of the certificates. We prove that just enabling two output bits per node instead of just one, and just enabling a slightly more complex interpretation that the conjunction operator, has a tremendous impact on the size of the certificates. For instance, it is known that verifying trees using the logical conjunction operator, on 1-bit-per-node outputs, requires certificates of $\Omega(\log n)$ bits. (This holds even in the proof-labeling setting, i.e., when the certificates can possibly depend on the identity assignment). One of our perhaps most surprising results is a proof that, by simply using the conjunction and the disjunction operators together, on only 2-bit-per-node outputs, one can verify trees using certificates of only $O(1)$ bits.

Importantly, several of our positive results use interpretations of the outputs that have desirable properties. In particular, they are idempotent, commutative and associative. As a consequence, all nodes can become aware of the decision result by a simple gossip protocol performing in $O(\log n)$ time whenever such a mechanism can be implemented on top of the network. Alternatively, the global decision can be computed by all nodes in $O(D)$ time in the $\mathcal{CONGEST}(1)$ model [75], where D denotes the diameter of the network. Our universal verifier, used to establish UNLD = All, does not satisfy the idempotence property. Nevertheless, the global decision can still be computed by all nodes in $O(D)$ time in the $\mathcal{CONGEST}(\log n)$ model.

[1]BPNLD stands for Bounded-error Probabilistic Non-deterministic Local Decision.

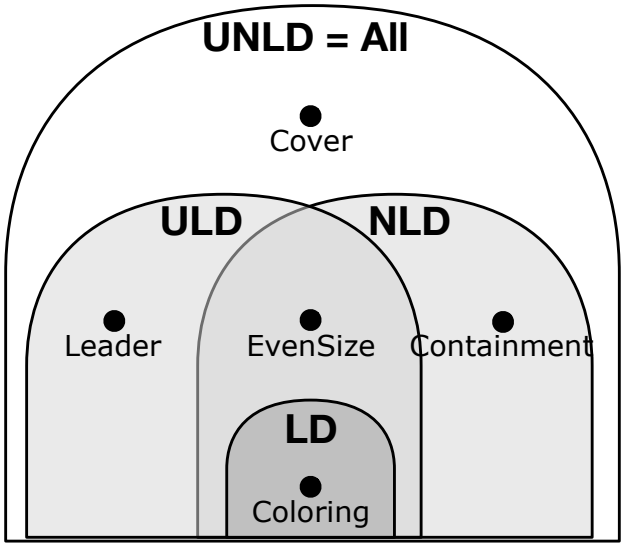


Figure 3-1: *Four distributed decision and verification classes, with representatives*

3.2 Separation results

Recall that the classes LD and NLD, defined in [45], are the respective restrictions of ULD and UNLD to the setting in which each node can output a single bit, and the interpretation is the result of the conjunction (AND) operator on these outputs. Hence, by definition, LD $\subseteq$ ULD, and NLD $\subseteq$ UNLD. Also, by definition, ULD $\subseteq$ UNLD. The purpose of this section is to show that these inclusions are strict (the strict inclusions LD $\subset$ NLD and LD $\subset$ NLD are established resp. in [45] and the previous chapter), and to study the relationship between ULD and NLD. The following result is illustrated in Figure 3-1.

Theorem 2 ULD $\setminus$ NLD $\neq \emptyset$*,* NLD $\setminus$ ULD $\neq \emptyset$*, and* LD $\subset$ (ULD $\cap$ NLD) $\subset$ (ULD $\cup$ NLD) $\subset$ UNLD = All*.*

The proof of the above theorem is direct by combining the following four lemmas, including Lemma 2 which, in addition, provides an upper bound on the size of the certificates enabling to place every language in UNLD.

Lemma 1 ULD $\setminus$ NLD $\neq \emptyset$ *and* LD $\subset$ ULD $\cap$ NLD*.*

Proof. Recall that $\texttt{Leader} = \{(G, \mathbf{x})$ s.t. $\forall u \in V(G), \mathbf{x}_u \in \{0,1\}$, and $\sum_{u \in V(G)} \mathbf{x}_u = 1\}$. We have $\texttt{Leader} \notin$ NLD because this language is not closed under lift (see [41] for the characterization of NLD in term of lifts). On the other hand, we showed in **??** that $\texttt{Leader} \in$ ULD. This establishes that ULD $\setminus$ NLD $\neq \emptyset$.

Let $\texttt{EvenSize} = \{(G, \mathbf{x})$ s.t. G has an even number of nodes$\}$. This language is in NLD because it is closed under lift (see [41]). To establish that $\texttt{EvenSize} \in$ ULD, consider the algorithm performing in zero rounds consisting, for each node u, in outputting the single bit $b_u = 1$. The decision is then made by applying the operator $\mathcal{I} = 1 - \bigoplus_{i=1}^{n} b_i$ to the collection $\{b_i \in \{0,1\}, i \in [n]\}$ of output bits, where $\oplus$ denotes the exclusive-disjunctive operator. The value of $\mathcal{I}$ is equal to 1 if and only if the graph has an even number of nodes. Now, we also have $\texttt{EvenSize} \notin$ 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 for C being 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 proves LD $\subset$ ULD $\cap$ NLD, which completes the proof. $\square$

Let us consider the language

$$\texttt{Cover} \;=\; \{(G, (\mathbf{e}, \mathbf{S})) \mid \exists v \in V(G),\ \exists S \in \mathbf{S}_v \text{ s.t. } S = \{\mathbf{e}_u : u \in V(G)\}\}$$

introduced in [45]. This language is formed by all configurations $(G, \mathbf{x})$ with $\mathbf{x}_u = (\mathbf{e}_u, \mathbf{S}_u)$, where $\mathbf{e}_u$ is an element of some universe U, and $\mathbf{S}_u = \{S_1, \ldots, S_{k_u}\}$ is a collection of sets with elements in U, such that there exists a node v whose collection $\mathbf{S}_v$ contains a set S that is equal to the set formed of all the elements $\mathbf{e}_u$ for all $u \in V(G)$. We have $\texttt{Cover} \in$ UNLD as a consequence of the combined observations that (1) by providing every node with an oracle deciding $\texttt{Leader}$, all distributed languages are in NLD, and (2) $\texttt{Leader} \in$ ULD. The first claim is implicit in [45], and the second has been established in the proof of Lemma 1. In other words, $\texttt{Cover} \in$ UNLD simply because UNLD $= \text{NLD}^{\texttt{Leader}} =$ All. We provide a complete proof of UNLD $=$ All below, for the purpose of completeness and further references in the text, and refer to [45] for more details on the impact of using oracles on the

theory of local decision.

Lemma 2 *Every TM-decidable distributed language is in* UNLD*. Moreover, the verification of languages on n-node networks with k-bit input per node can be achieved using certificates of $O(n^2 + kn)$ bits, by having each node inspecting its neighborhood at distance* 1*, and with 1-bit-per-node outputs.*

Proof. Let $\mathcal{L}$ be a language. We describe a 1-round nondeterministic verification scheme $(\mathcal{A}, \mathcal{I})$ for $\mathcal{L}$. The certificate $\mathbf{y}$ of an instance $(G, \mathbf{x}) \in \mathcal{L}$ is a $n \times n$ adjacency matrix M of G, with vertices indexed arbitrarily by distinct integers in $[1, n]$, plus a n-dimensional vector I where I_i is the input of vertex $i \in [1, n]$. In addition, every node v receives the index $\lambda(v) \in [1, n]$ corresponding to v in M and I. More formally, the certificate at node v is $\mathbf{y}_v = ((G', \mathbf{x}'), i)$, where G' is an isomorphic copy of G with nodes labeled by λ from 1 to n, $\mathbf{x}'$ is an n-dimensional vector such that $\mathbf{x}'_{\lambda(u)} = \mathbf{x}_u$ 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.

The local algorithm $\mathcal{A}$ executed on an instance $(G, \mathbf{x})$ with certificate $\mathbf{y}$ outputs one bit c_u at every node u. Let us first describe an algorithm with two bits a_u and b_u at every node u, and then we will show how to reduce these two bits into just one. Every node u with index $\lambda(u) = 1$ sets $a_u = 1$. The others set $a_u = 0$. For computing b_u, every node performs a single round of communication. First, every node u checks that it has received the input as specified by $\mathbf{x}'$, i.e., u checks whether $\mathbf{x}'_{\lambda(u)} = \mathbf{x}_v$, and set $b_u = 0$ if this does not hold. Second, each node u communicates with its neighbors to check that (1) they all got the same graph G' and the same input vector $\mathbf{x}'$, and (2) they are indexed the way they should be according to the map G'. If some inconsistency is detected by a node, 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 $(G', \mathbf{x}') \in \mathcal{L}$, and to 0 otherwise. All nodes u output the pair (a_u, b_u). The decision is then made according to the collection $\{(a_i, b_i) \in \{0, 1\}^2, i \in [n]\}$ of outputs, by applying the operator

$$\mathcal{I} = \left(\bigvee_{i=1}^{n} \left(a_i \wedge \bigwedge_{j \neq i} \overline{a_j} \right) \right) \wedge \left(\bigwedge_{i=1}^{n} b_i \right)$$

which is 1 if and only if $(G, \mathbf{x}) \in \mathcal{L}$. To see why, 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 $\mathbf{x}'$. 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 $(G', \mathbf{x}')$ is either identical to $(G, \mathbf{x})$ or to a lift of it[2]. It follows that, if all bits b_u are 1, then $(G', \mathbf{x}') = (G, \mathbf{x})$ if and only if there exists exactly one node $v \in G$, whose index $\lambda(v) = 1$. This is precisely the `Leader` problem, which is decided using the a_us.

Now, we reduce the two bits a_u and b_u into just one bit c_u. This reduction is based on the observation that if any node u detects some inconsistencies, then at least one of it neighbors also detects the same inconsistencies. As a consequence, if some node "raises an alarm" (i.e., set $b_u = 0$), then at least another node does the same. Thus, every node u sets $c_u = a_u \vee \overline{b_u}$ and output 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

$$\mathcal{I}' = \bigvee_{i=1}^{n} \Big(c_i \wedge \bigwedge_{j \neq i} \overline{c_j} \Big)$$

which is 1 if and only if $(G, \mathbf{x}) \in \mathcal{L}$. Indeed, $c_u = 1$ if and only if u detects some inconsistencies (i.e., $b_u = 0$) or $\lambda(u) = 1$ (i.e., $a_u = 1$). However, if u has detected some inconsistencies, then one of its neighbors u' has also detected the same inconsistencies, which guarantees $c_{u'} = 1$ for u' as well. Thus $\mathcal{I}' = 0$ if $(G, \mathbf{x}) \notin \mathcal{L}$. (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 UNLD = All. $\square$

Lemma 3 ULD $\cup$ NLD $\subset$ UNLD.

Proof. It is known that `Cover` $\notin$ NLD [45]. We prove that `Cover` $\notin$ ULD by contradiction, using arguments from communication complexity. Assume that there exists a local algorithm $\mathcal{A}$ and an interpretation $\mathcal{I}$ of the individual outputs produced by $\mathcal{A}$ enabling to decide `Cover`. In particular, $(\mathcal{A}, \mathcal{I})$ must decide the restricted version of `Cover`, defined on paths $P = (v_1, \ldots, v_n)$ with $U = \{0,1\}^k$, defined as follows. Let $\bar{0}$ denote the k-bit string formed by k consecutive 0s. We set

$$\mathbf{e}_1 = x, \mathbf{e}_n = y, \text{ and } \mathbf{e}_i = \bar{0} \text{ for } 1 < i < n,$$

[2]A graph H is a lift of a graph G if there exists a homomorphism from H to G preserving the neighborhood of each node.

and $\mathbf{S}_i = \{S_i\}$ for $i = 1, \ldots, n$ with

$$S_1 = \{\bar{0}, x\}, S_n = \{\bar{0}, y\} \text{ and } S_i = \emptyset \text{ for } 1 < i < n \ .$$

Such a configuration is in `Cover` if and only if $x = y$. We show that, using $(\mathcal{A}, \mathcal{I})$, one could solve the communication complexity problem "Equality" between Alice and Bob, by exchanging less than k bits. Assume $\mathcal{A}$ performs in t rounds. Then, given x as input, Alice simulates the algorithm $\mathcal{A}$ applied at the $n - t - 1$ nodes $v_1, \ldots, v_{n-t-1}$, while, given y as input, Bob simulates $\mathcal{A}$ applied to the $t + 1$ nodes $v_{n-t}, \ldots, v_n$. Assume that $\mathcal{A}$ 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 $(t+1)B = O(1)$ bits to Alice so that she can apply $\mathcal{I}$ on these bits together with her own $(n - t + 1)B$ bits to determine whether $x = y$ or not. This holds for any $x, y \in \{0, 1\}^k$. This is a contradiction, whenever $k > (t + 1)B$ because "Equality" requires k bits to be exchanged between Alice and Bob for being solved. Hence `Cover` $\notin$ ULD $\cup$ NLD, which completes the proof. $\square$

Lemma 4 NLD $\setminus$ ULD $\neq \emptyset$.

Proof. Let us consider the following language, similar to `Cover`:

$$\texttt{Containment} \ = \ \{(G, (\mathbf{e}, \mathbf{S})) \mid \exists v \in V(G), \ \exists S \in \mathbf{S}_v \text{ s.t. } S \supseteq \{\mathbf{e}_u : u \in V(G)\}\}$$

The two languages `Cover` and `Containment` differ only in the fact that `Cover` asks for $S = \{\mathbf{e}_u : u \in V(G)\}$ while `Containment` simply asks for $S \supseteq \{\mathbf{e}_u : u \in V(G)\}$. It is known [45] that `Containment` $\in$ NLD. Now, by the same arguments as for proving `Cover` $\notin$ ULD, one can show `Containment` $\notin$ ULD as well. $\square$

Remark. Lemma 2 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 [62]. However, while proof-labeling schemes allows certificates to depend on the identity assignment, our verification algorithm uses certificates that are independent of the identity assignment.

3.3 On the certificate size

3.3.1 Minimum certificate size for universal verification

By Lemma 2, 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 1-bit outputs. The following theorem proves that this bound is tight, in the sense that, for every k, there exist languages with k-bit inputs which require certificates of size $\Omega(n^2 + nk)$ bits to be verified in t rounds for b-bit outputs, for all t and b.

Theorem 3 *There exist languages with k-bit inputs that require certificates of size $\Omega(n^2 + nk)$ bits in n-node networks to be verified locally (i.e., to be placed in* UNLD*).*

Proof. We define the language `Symmetry` as follows. Given a graph G with k-bit input $\mathbf{x}_u$ per node u, an *input-preserving* automorphism ϕ of G is an automorphism satisfying $\mathbf{x}_u = \mathbf{x}_{\phi(u)}$ for every node u. Let

$$\texttt{Symmetry} = \{(G, \mathbf{x}) : \text{ there is a non-trivial input-preserving automorphism for } G\}.$$

The proof that `Symmetry` requires $\Omega(n^2 + nk)$ bits to be verified in n-node networks with k-bit inputs is based on a construction used in [55] to prove a lower bound on the size of the certificates when using the conjunction operator. We extend the arguments from [55] so that they apply to languages with inputs (and not only to graph properties), and apply to all possible operators for interpreting b-bit outputs (and not only the conjunction operator for 1-bit outputs).

Let $\mathcal{F}_{n,k}$ be the family of configurations $(G, \mathbf{x})$ where G is a non-symmetric graph with n-nodes, and $|\mathbf{x}_u| = k$ for every node u of G. More precisely, by labeling the nodes of G from 1 to n in 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 [55] 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 configurations $(F_1, \mathbf{x}_1)$ and $(F_2, \mathbf{x}_2)$ in $\mathcal{F}_{n,k}$, let $(G, \mathbf{x}) = (F_1, \mathbf{x}_1) + (F_2, \mathbf{x}_2)$ be the configuration formed by a copy of F_1 together with its inputs $\mathbf{x}_1$, a copy of F_2 together with its inputs $\mathbf{x}_2$, and a path P of $4t + 1$ nodes (without inputs), connecting the node with label 1 in F_1 to the node with label 1 in F_2. The number of nodes in G is $2n + 4t + 1 = \Theta(n)$. Let

$$\mathcal{C} = \{(G, \mathbf{x}) = (F_1, \mathbf{x}_1) + (F_2, \mathbf{x}_2) \ : \ (F_1, \mathbf{x}_1) \in \mathcal{F}_{n,k} \text{ and } (F_2, \mathbf{x}_2) \in \mathcal{F}_{n,k}\}.$$

We show that even verifying `Symmetry`-membership for configurations 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 $(G, \mathbf{x}) \in \mathcal{C}$, we have $(G, \mathbf{x}) \in \texttt{Symmetry}$ if and only if $(F_1, \mathbf{x}_1) = (F_2, \mathbf{x}_2)$. (Recall that the graphs in $\mathcal{F}_{n,k}$ are labeled, and thus equality here means the existence of a label-preserving input-preserving isomorphism between F_1 and F_2). Let $\mathcal{C}_{\text{sym}}$ be the subset of $\mathcal{C}$ consisting of symmetric graphs in $\mathcal{C}$, i.e., $\mathcal{C}_{\text{sym}} = \mathcal{C} \cap \texttt{Symmetry}$. We have:

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

Note that $|\mathcal{C}_{\text{sym}}| = |\mathcal{F}_{n,k}| \geq 2^{c(n^2+nk)}$ for some constant $c > 0$ and for big enough values of n. Assume now, for the sake of contradiction, that one can verify `Symmetry` in t rounds with certificates of size $s = o(n^2 + nk)$ bits per node, using algorithm $\mathcal{A}$ with interpretation $\mathcal{I}$. Then, for every configuration in $\mathcal{C}$, the path P includes $4t + 1$ certificates, for a total of $(4t+1)s$ bits, that is still $o(n^2 + nk)$ bits since t is constant. Therefore, there are at least

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

graphs in $\mathcal{C}_{\text{sym}}$, that have the same collection of certificates on their respective paths P, for some c', $0 < c' < c$. 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 upper bounded. If $\left(\!\binom{x}{y}\!\right)$ denotes the multinomial coefficient "x multichoose y", then this number is:

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

Therefore

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

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

Since R is very large compared to N^2, there exist two configurations $(G_1, \mathbf{x}_1) = (F_1, \mathbf{x}'_1) + (F_1, \mathbf{x}'_1)$ and $(G_2, \mathbf{x}_2) = (F_2, \mathbf{x}'_2) + (F_2, \mathbf{x}'_2)$ 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 $(F_1, \mathbf{x}'_1)$ and $(F_2, \mathbf{x}'_2)$ 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 $(F_1, \mathbf{x}'_1)$ and $(F_2, \mathbf{x}'_2)$ connected to the nodes with identities $2n+3t+1, \ldots, 4t+1$ on P. Let us denote by M_0 the multi-set of produced produced by $\mathcal{A}$ on the $2t+1$ nodes at the middle of P in both configuration $(G_1, \mathbf{x}_1)$ and $(G_2, \mathbf{x}_2)$.

Now, consider the following configuration $(G, \mathbf{x})$ formed by "cutting and gluing" $(G_1, \mathbf{x}_1)$ and $(G_2, \mathbf{x}_2)$. More precisely, $(G, \mathbf{x})$ is formed by connecting $(F_1, \mathbf{x}_1)$, $(P, \varnothing)$, and $(F_2, \mathbf{x}_2)$, with identities in $[1, n]$ for F_1, in $[n+1, 2n]$ for F_2, and, as usual, in $[2n+1, 2n+4t+1]$ for P. Let us provide these nodes with the certificates inherited from these respective copies of $(F_1, \mathbf{x}_1)$, and $(F_2, \mathbf{x}_2)$. Each node with identities $\{1, ..., n\} \cup \{2n+1, \ldots, 2n+t\}$ (resp., with identities in $\{n+1, \ldots, 2n\} \cup \{2n+3t+1, \ldots, 2n+4t+1\}$) has the same local view of radius t in $(G, \mathbf{x})$ as in $(F_1, \mathbf{x}_1)$ (resp., $(F_2, \mathbf{x}_2)$). Moreover, nodes in the middle part of the path, with identities in $[2n+t+1, 2n+3t]$ have the same view in $(G, \mathbf{x})$ as in $(G_1, \mathbf{x}_1)$ and $(G_2, \mathbf{x}_2)$. Therefore, the verification algorithm $\mathcal{A}$ outputs the same multi-set $M_0 \cup M_1 \cup M_2$ for the illegal configuration $(G, \mathbf{x})$, as it does for the legal configurations $(G_1, \mathbf{x}_1)$ and $(G_2, \mathbf{x}_2)$, yielding the desired contradiction. $\square$

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 $c < 1$, and, by the construction of the accepted illegal configuration, even for verification algorithms performing in time up to $o(n)$ rounds.

3.3.2 Verifying trees 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 [62] that verifying `Tree` using the conjunction operator requires $\Omega(\log n)$-bit certificates for n-node trees. 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 conjointly the conjunction and disjunction operators, on 2-bit outputs, enables to verify `Tree` in one round, using certificates of only $O(1)$ bits. Moreover, as we can see in the proof of this result, the decision is made according to the application of a 2-bit logical operator $\mathcal{I}$ 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 the $\mathcal{CONGEST}(1)$ model.

Theorem 4 `Tree` *can be 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 of `Tree`, i.e., for a tree T. The certificate assigned to node v is a pair $\mathbf{y}_v = (r(v), d(v))$, 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_0 of T, and set u_0 as the root of T. Set $r(u_0) = 1$, and $r(v) = 0$ for every node $v \neq u_0$. For every $v \in V(T)$, let $d(v) = dist_T(v, u_0) \bmod 3$, where $dist_T(x, y)$ 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 performs in just one round, during which every node v sends its certificate $\mathbf{y}_v$ 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 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:

$$\mathcal{I} = \Big(\bigwedge_{i=1}^{n} a_i\Big) \wedge \Big(\bigvee_{i=1}^{n} b_i\Big).$$

By construction, if T is a tree, then $\mathcal{I} = 1$. Indeed, all tests are passed successfully, and thus the (unique) node v with $r(v) = 1$ returns $(1, 1)$ while all the other nodes return $(1, 0)$.

Establishing that $\mathcal{I} = 0$ whenever T is not a tree, independently from 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 $\mathcal{I} = 0$. To see why this is indeed the case, assume that the current input (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 $\mathcal{I} = 0$, 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 in 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. Since G is not a tree, there is a cycle C in G. Since $a_i = 1$ for all i, it must be the case that all edges of the cycle are consistently oriented along C. That is, each node in C has exactly one outgoing edge in C and one incoming edge in C. In particular, all edges incident to C are entering C. As a consequence, there is a unique cycle in G. Indeed, if there were two node-disjoint cycles, then one could not guarantee consistency of the edge orientation along a path connecting these two cycles. The same holds if the two cycles would share one or more nodes. So, G is an "octopus". That is, it consists of a cycle C to which are attached a collection of trees, whose edges are all consistently oriented toward the cycle. Therefore, every node has an 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 $\mathcal{I} = 0$, which completes the proof of the theorem. $\square$

3.4 Conclusion

The focus on *unrestricted* decision and verification enables a new approach to distributed computing that highlights the role of the interpreter $\mathcal{I}$ as a key component in the decision and verification process. Our framework borrows indeed from property testing and sensor networks the way the decision is taken by the unconstrained interpretation function applied to the set of outputs produced by individual queries or sensors. Although a universal interpreter may seem too powerful, it is thus well motivated by several setting, and it opens new perspectives in the study the interpretation function. It seems reasonable to consider that the AND interpretation is far less constrained than the XOR interpretation[3] for example, the former giving the same decision whether one or many nodes output 1, the latter requiring a specific parity on the number of nodes outputting 1. Is it then possible to establish a hierarchy over all possible interpretation functions? And if the answer to this question is yes, then, is it possible to refine the classification inside ULD and UNLD according to the power of $\mathcal{I}$?

On a wider perspective, our approach allows us to set ground for a complexity theory for the $\mathcal{LOCAL}$ model. The separation results we have established raise the question of characterization. [72] show that it is in general undecidable whether a language in LCL has an algorithm. What about languages in ULD and UNLD? Our results also raise open questions about other settings in the $\mathcal{LOCAL}$ model, such as the *approximate decision* or *probabilistic decision*. For instance, one could ask if there's an unrestricted version to the classes BPLD and BPNLD [44] that capture the use of randomization as a computational resource, and if so, how do they relate to the classes presented here? In the second part of this thesis, we introduce a model that captures the use of extra computational resources such as randomization but also the use of quantum resources. We study the impact of the locality requirement as a concept governed by physical laws and its consequences on distributed computing.

[3] It is the exclusive OR. It corresponds, for n outputs in $\{0, 1\}$ to the parity of nodes outputting 1.

Part II

Causality and its algorithmic consequences

Chapter 4

Non-Signalling Model: capturing the concept of causality.

4.1 Introduction

One of the most celebrated results in the context of network computing is Linial's $\Omega(\log^* n)$ lower bound [68] on the number of rounds required for 3-coloring the n-node ring distributively. In essence, this lower bound states that even if nodes can communicate an arbitrary large amount of data between neighbors at every communication round, and even if nodes can perform an arbitrarily large amount of computation between every two communication rounds, 3-coloring the n-node ring in a distributed manner requires $\Omega(\log^* n)$ communication rounds. In other words, 3-coloring the ring requires some information to flow between nodes at distance larger than any constant. This lower bound is very robust. In particular, it holds even for Las Vegas algorithms where the nodes have access to a shared source of randomness.

In this chapter, we question the *universality* of results such as Linial lower bound. A lower bound (or an impossibility result) established for a distributed computing model offering very specific features has indeed little *conceptual* interests. (It may however have a significant *practical* interest if the model reflects a widely used technology). Instead, if the model is generic enough to capture a large amount of frameworks, then the lower bound is quite significant conceptually. This is the case of Linial lower

bound, but up to some extent only. Indeed, on the one hand, the formal distributed computing model used in [68] – the $\mathcal{LOCAL}$ model – is very liberal, and therefore the lower bound remains valid in very many contexts. Still, the $\mathcal{LOCAL}$ model is based on classical physics, while there are several physical evidences[1] indicating that we may not be living in a world governed by classical physics. A natural question is therefore to ask whether, for example, the $\Omega(\log^* n)$ lower bound for 3-coloring the ring still holds if nodes are able to store, manipulate, and exchange resources such as, e.g., quantum bits (qubits).

At this point, we want to point out that the question of whether quantum computers able to manipulate a large number of qubits will one day exist is still open, and our work is not aiming at arguing in favor nor against this existence. Nevertheless, the practical efficiency of quantum effects in the context of distributed computing has already been demonstrated. One preeminent example is the establishment of long-distance quantum cryptographic channels between two parties[2]. Hence, while it is not completely clear whether connecting a large number of powerful quantum computers able to exchange very many qubits is achievable, it is a fact that the presence in a network of a handful of computers capable of manipulating just a few qubits may radically change the computational power of the network [21, 24, 30]. It is therefore of the utmost importance to determine whether or not the known limitations of network computing can be overcame by using quantum mechanic effects, and if so, to which extend.

Our contributions

Tackling the above question may not necessarily require any knowledge regarding quantum physics and/or quantum computing. One purpose of our work is in fact to point out to the reader that, for several frameworks, and for several problems, lower bounds for quantum distributed computing can be derived *without manipulating any concepts related to quantum mechanics*. Indeed, there exist models that offer the same flavor as classical models, but subsume quantum distributed computing. Here is why: roughly, (classical) distributed algorithms using shared randomness produce outputs

[1] John F. Clauser, Alain Aspect, and Anton Zeilinger received the Wolf Prize in Physics in 2010 for, in particular, their increasingly sophisticated series of tests of Bell's inequalities.

[2] There are currently companies offering commercial quantum key distribution systems.

that are statistically distributed according to some specific kinds of distributions, and the same holds for the outputs produced by quantum distributed algorithms. Let us denote by $\mathcal{D}_{random}$ and $\mathcal{D}_{quantum}$ the set of distributions produced by the former and the latter, respectively. We have $\mathcal{D}_{random} \subset \mathcal{D}_{quantum}$, but the structure of $\mathcal{D}_{quantum}$ is quite difficult to handle for whom is not familiar with quantum computing. The good news is that there is a larger set of distributions, that is easier to handle than $\mathcal{D}_{quantum}$, but restricted enough so that non-trivial lower bounds can be designed for it. This latter set of distributions is called *non-signaling*, and is denoted here by $\mathcal{D}_{nonsignaling}$.

Informally, as the distributions in the sets $\mathcal{D}_{random}$ and $\mathcal{D}_{quantum}$ are constrained by classical and quantum mechanics, respectively, the distributions in the set $D_{nonsignaling}$ are only constrained by *relativistic causality*. The assumption of relativistic causality states that effects belong to the light cone of their causes, or, alternatively, *causal influences* do not travel faster than the speed of light. Hence, this type of distributions captures not only classical distributions, but also distributions that appear due to quantum effects. The reason why quantum correlations belong to the set $D_{nonsignaling}$ of non-signaling correlations may seem unclear to the reader unfamiliar with quantum computing. Indeed, some quantum phenomena seem to be in contradiction with relativistic causality. A typical case of such an apparent contradiction shows up when considering a system of a pair of infinitely distant particles that have been pre-set in entangled states, for which the measure of one particle's state gives immediate knowledge about the other particle's state. This phenomenon, which violates the classical concept of *local realism*, is known as quantum *nonlocality*. However, it is crucial to note that this phenomenon does not violate relativistic causality. Indeed, quantum nonlocality is just the expression of some particular probability distributions. As an example, the EPR paradox [36] involves two particles a and b, each one having a *spin* $+1$ or -1, such that the joint probability distribution of the pair of spins satisfies:

$$\begin{cases} \Pr[a=+1,\ b=-1] &= \Pr[a=-1,\ b=+1] &= \frac{1}{2} \\ \Pr[a=+1,\ b=+1] &= \Pr[a=-1,\ b=-1] &= 0. \end{cases}$$

In this case, a measurement performed on a gives immediate information on b, although no signals propagate from b to a. However, the marginal distributions of a and b are independent from each other (both are uniform in $\{-1,+1\}$). Prior to mea-

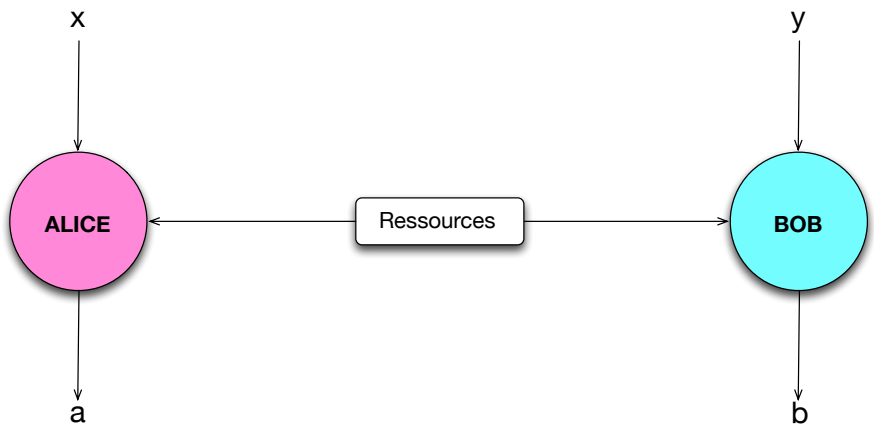


Figure 4-1: *Alice receives boolean x as input, while Bob receives boolean y. Alice and Bob are separated and cannot communicate. However, they had access to a common source of resources (e.g., random bits, intricate qubits, etc.) before being separated, and before getting their inputs. In the (δ, f) game, they have to compute boolean outputs a and b, respectively, such that $\delta(a, b) = f(x, y)$.*

surement, both have equal probabilities of being 0 or 1. The fact that the marginal distributions are independent of each other is the evidence that the above joint distribution is non-signaling. More generally, it has been proved that quantum correlations are nonlocal, but cannot be of any use for transmitting signals faster than light [52], and thus quantum correlations are non-signaling.

Non-signaling distributions form a stronger model than quantum computing. It is however unlikely to be a realistic model as argued in [28]. (If every non-signaling distribution could be realized in a physical experiment, then non plausible distributed computations could be achieved, like computing the scalar product of two vectors located at remote places by exchanging only one bit of information). Nevertheless, this chapter is interested in impossibility results, and, as we will show, the known chain of strict containments

$$D_{random} \subset D_{quantum} \subset D_{nonsignaling}$$

enables to establish lower bounds on the power of distributed quantum computing, for several problems at least.

Related work

The paper [51] inspired us very much. In that paper, the authors define different extensions of the $\mathcal{LOCAL}$ model, by enabling processors to manipulate qubits, and they establish several separation results between these extensions. They also point out that lower bounds for distributed quantum computing can be obtained by considering a stronger model, called φ-$\mathcal{LOCAL}$ in [51]. It turns out that this latter model is noting else than the non-signaling model considered in our previous chapters. We prefer using the terminology "non-signaling" because it is the standard terminology used in the physics literature and in our work.

There is a huge literature on design and analysis of algorithms in the $\mathcal{LOCAL}$ model of distributed computing, and we refer the reader to the first part of this dissertation, particularly Chapter 1 and references therein. As far as distributed graph coloring is concerned, we already mentioned the lower bound $\Omega(\log^* n)$ on the number of rounds for $(\Delta+1)$-coloring of n-node graphs [68], where Δ denotes the maximum degree. So far, the best known upper bound for deterministic algorithms is $2^{O(\sqrt{\log n})}$ rounds in [74], while the best known upper bound for randomized (Las vegas) algorithms is $O(\log n)$ expected number of rounds [3, 69]. These bounds can be improved for bounded degree graphs. Specifically, $(\Delta+1)$-coloring can be randomly computed in expected $O(\log\Delta+\sqrt{\log n})$ communication rounds (see [77]) recently improved to $O(\log\Delta+e^{O(\sqrt{\log\log n})})$ rounds in [14]. In contrast, the best known deterministic algorithm performs in $O(\Delta+\log^* n)$ rounds [13, 64].

The same way Monsieur Jourdain has been speaking prose without knowing it, many lower bounds in the literature are using arguments that can be directly applied to distributed quantum algorithms. As mentioned in [38], this is typically the case of "limited sight" arguments, since quantum mechanics respect causality. For instance, [51] noticed that the lower bounds for Maximal Independent Set [66], Locally-Minimal Coloring [50], and Sparse Connected Subgraph [31, 37] also hold for quantum computing. However, proofs based on other kinds of arguments, like, typically, Linial's lower bound [68], do not extend trivially to quantum computing models. (We shall come back to this issue later in the text).

Regarding distributed quantum computing, one can already cite a few contributions (see, e.g., the surveys [21, 24]). For instance [30] lists a series of papers aiming

at solving leader election in several variants of distributed quantum computing. Recently, [38] tackles several network problems (connectivity, MST, etc.) in a quantum computing model extending the $\mathcal{CONGEST}$ model [75] to capture quantum effects. Closer to our work are all contributions to multi-player "pseudo-telepathy" games (i.e., games solvable in absence of communications, using entanglement). We refer to the surveys [20] for the analysis of several such games. In particular, the fact that the CHSH game [26] can be solved with probability $\cos^2(\pi/8)$ has been established in [18, 23], while [25] showed that this is the best success probability that can be achieved by a quantum strategy. Two-player games have also been investigated using the concept of boxes. A box is a conceptual device which receives pairs of inputs, and returns pairs of outputs distributed according to some non-signaling probability distribution which characterizes the box. The PR-box, solving the CHSH game with probability 1, has been introduced in [76], and is further studied in [15, 16]. It is known [16] that PR-boxes can be used to simulate any binary-output games. However, they cannot alone simulate any multiple-output games [34].

4.2 Local[3] computing

Let us consider a distributed algorithm $\mathcal{A}$ performing in networks modeled as simple connected undirected graphs, under a synchronous message passing model [75]. In this model, processors have pairwise distinct identities. They are woken up simultaneously, and computation proceeds in fault-free synchronous rounds during which every processor exchanges messages of unlimited size with its neighbors, and performs arbitrary computations on its data.

More specifically, the n nodes of network $G = (V, E)$ are given pairwise distinct identities in $[n] = \{1, \ldots, n\}$, and we denote by $Id(u)$ the identity of node u. Let $\mathbf{x}$ be n-dimensional vector denoting the inputs to the nodes, where $\mathbf{x}_i \in \{0,1\}^*$ denotes the input binary string given to the node with identity i, for $i = 1, \ldots, n$. Similarly, let us denote by $out_i \in \{0,1\}^*$ the output of the node with identity i, and let $out = (out_i)_{i \in [n]}$. Let $t = t(G, \mathbf{x}, Id)$ be the running time of $\mathcal{A}$ for the

[3]In Chapters 4 and 5, some of the terminology is borrowed from Physics. Hence, the term "local" here doesn't designate the $\mathcal{LOCAL}$ model as presented in the previous chapters, but it rather refers to *local realism*. Similarly, the term "nonlocal" refers to phenomena that violate local realism.

input configuration $(G, \mathbf{x}, Id)$ where G is an n-node network with nodes identified by $Id : V \to [n]$, and where the node with identity i receives input $\mathbf{x}_i$, for every $i \in [n]$. That is, the output out_i of the node with identity i is computed by $\mathcal{A}$ based solely on the ball $B_{G,\mathbf{x},Id}(i,t)$ of radius t in G, centered at node labeled i, including the structure of the subgraph of G induced by all nodes at distance at most t from i, together with the inputs and the identities of these nodes. In the sequel, when it is clear from the context, the subscripts G, Id, and $\mathbf{x}$ will omitted, and $B_{G,\mathbf{x},Id}(i,t)$ simply denoted by $B(i,t)$.

A *task* T is defined by an input-output relation that, given a triple $(G, \mathbf{x}, Id)$, specifies all valid output vectors *out* for graph G with nodes identified by *Id*, and input $\mathbf{x}$. For instance, in the case of the k-coloring task, given $(G, \mathbf{x}, Id)$, one must have (1) $out_i \in \{1, \ldots, k\}$ for every $i \in [n]$, and (2) $out_i \neq out_j$ whenever the two nodes with respective identities i and j are adjacent in G.

We are interested in the different resources that can be accessed by an algorithm solving a task T in the above message synchronous passing model.

If $\mathcal{A}$ is deterministic, then $\mathcal{A}$ is simply a mapping from D to $\{0,1\}^*$, where D is the *domain* of $\mathcal{A}$, that is, D is the set of all possible balls $B(i,t)$ that can be formed by legal inputs to $\mathcal{A}$. (E.g., if $\mathcal{A}$ is designed for planar graphs with boolean inputs given to nodes, then a ball $B(i,t)$ is legal if and only if it is planar, with $\mathbf{x}_j \in \{0,1\}$ for every node j in this ball). Therefore, the output vector $out = (out_i)_{i \in [n]}$ for some input configuration $(G, Id, \mathbf{x})$ is simply defined by its n coordinates $out_i = \mathcal{A}(B(i,t))$ for $i = 1, \ldots, n$. This can be denoted as:

$$out \mid (G, Id, \mathbf{x}) = \Big(\mathcal{A}(B(i,t)\Big)_{i \in [n]} . \tag{4.1}$$

Randomization provides additional power to the algorithm. In particular, assuming that every node is given a private source of random values ω in some probabilistic space Ω, then, for every $y \in \{0,1\}^*$, the probability that the node with identity i output y depends on $B(i,t)$ as well as on the collection of random values at the nodes in this ball. In this context, algorithm $\mathcal{A}$ yields a probability distribution on the possible outputs *out*. That is, the output vector is now a random variable $\mathbf{Y}$, and

$$\mathbf{Y} \mid (G, \mathbf{x}, Id) = \Big(\mathbf{Y}_i \mid B(i,t)\Big)_{i \in [n]} \tag{4.2}$$

where $\mathbf{Y}_i$ is the ith coordinate of $\mathbf{Y}$, that is the random variable corresponding to the output at node i. Typically, if the nodes do not exchange their private random values, and if every node i acts by, first, collecting the data in $B(i,t)$, and, second, computing its (random) output out_i based on these data and on its private random coins, then the distribution of the output $\mathbf{Y}$ can be expressed, for any fixed $out = (out_i)_{i \in [n]}$, as:

$$\Pr[\mathbf{Y} = out \mid (G, \mathbf{x}, Id)] = \prod_{i=1}^{n} \Pr[\mathbf{Y}_i = out_i \mid B(i,t)] \ .$$

In the above expression, $\Pr[out_i \mid B(i,t)]$ denotes the distribution of the output out_i at node i applying Algorithm $\mathcal{A}$, knowing $B(i,t)$. This distribution depends in turn of the distribution of the private random values $\omega \in \Omega$ used at node i.

Obviously distributions that can be described by Eq. (4.2) subsume those that can be defined by Eq. (4.1). Shared randomness enlarges the spectrum of possible distributions even further. In the context of shared randomness, nodes have collectively access to a common source of random values λ in some probabilistic space Λ, in addition to possible private sources of randomness. The distribution of the output $\mathbf{Y}$ can then be expressed as:

$$\mathbf{Y} \mid (G, \mathbf{x}, Id) = \Big(\mathbf{Y}_i \mid B(i,t) \wedge \lambda\Big)_{i \in [n]} \text{ with probability } \Pr[\lambda], \text{ for every } \lambda \in \Lambda. \tag{4.3}$$

In the above expression, $\Pr[\lambda]$ denotes the distribution of the random value $\lambda \in \Lambda$, while $\mathbf{Y}_i \mid B(i,t) \wedge \lambda$ denotes the distribution of the output at node i applying Algorithm $\mathcal{A}$, knowing $B(i,t)$ and λ, which may also depend on the distribution of private values $\omega \in \Omega$. Again, obviously, the above expression subsumes the one in Eq. (4.2). Note also, that if the nodes proceed in first collecting all data in their ball of radius t, and then using their private values to compute their outputs, then

$$\Pr[\mathbf{Y} = out \mid (G, \mathbf{x}, Id)] = \sum_{\lambda \in \Lambda} \prod_{i=1}^{n} \Pr[\mathbf{Y}_i = out_i \mid B(i,t) \wedge \lambda] \cdot \Pr[\lambda] \ .$$

All the previous expressions for $\mathbf{Y}$ in Equations (4.1), (4.2), and (4.3) capture locality

in the sense that distant events can only be weakly correlated. That is, the outputs at two nodes i and j such that $B(i,t) \cap B(j,t) = \emptyset$ are only correlated according to the distributions generated by shared randomness. Nevertheless, it is worth pointing out that causality is not captured in its full generality by these equations. Indeed, roughly speaking, causality expresses the fact that the *distribution* of outputs at node i should not depend on events taking place at far away nodes, i.e., at nodes lying in G at distance greater than t. The expression of $\mathbf{Y}$ yielded by shared randomness satisfies this constraint. However, *there are distributions that satisfy causality which cannot be expressed as Eq.* (4.3). To see why, we need to formally define the notion of *non signaling* distributions, as introduced in the next section.

4.3 Nonlocal computing

4.3.1 Non-signalling distributions

Let us consider a probability distribution on the output vectors $out = (out_i)_{i\in[n]}$, $out_i \in \{0,1\}^*$, described by a random variable $\mathbf{Y}$ conditioned over all n-node graphs with nodes labeled with pairwise distinct identities in $[n]$, where the node with identity i is given input $\mathbf{x}_i \in \{0,1\}^*$, for every $i \in [n]$. More precisely, $\mathbf{Y}$ is defined conditionally to all possible triples $(G, \mathbf{x}, Id)$ where Id is the identity function for G, and $\mathbf{x} = (\mathbf{x}_i)_{i\in[n]}$, $\mathbf{x}_i \in \{0,1\}^*$, denotes the inputs given to the nodes. To describe $\mathbf{Y}$, we are thus given a collection of conditional distributions

$$\{\mathbf{Y} \mid (G, \mathbf{x}, Id) \text{ for all triples } (G, \mathbf{x}, Id)\}. \tag{4.4}$$

where $\sum_{out} \Pr[\mathbf{Y} = out \mid (G, \mathbf{x}, Id)] = 1$ for every configuration $(G, \mathbf{x}, Id)$. Typical examples of such a distribution are those distributions as in Eq. (4.1), (4.2), and (4.3). For a fixed configuration $(G, \mathbf{x}, Id)$, the distribution $\mathbf{Y} \mid (G, \mathbf{x}, Id)$ yields the marginal distributions $\mathbf{Y}_i \mid (G, \mathbf{x}, Id)$ defined over all binary strings in $\{0,1\}^*$, $i = 1, \ldots, n$. This distribution is corresponding to the way the i-th coordinate of the output $\mathbf{Y}$ distributes whenever the whole vector $\mathbf{Y}$ distributes according to Eq. (4.4). More generally, for every subset $I \subseteq [n]$, the distribution $\mathbf{Y} \mid (G, \mathbf{x}, Id)$ yields the marginal

distributions

$$\mathbf{Y}_I \mid (G, \mathbf{x}, Id)$$

defined over all $|I|$-dimensional vectors with coordinates in $\{0,1\}^*$. It is the distribution corresponding to the way the vector $\mathbf{Y}_I = (\mathbf{Y}_i)_{i \in I}$ distributes whenever the whole vector $\mathbf{Y}$ distributes according to Eq. (4.4). The set of marginal conditional distributions

$$\{\mathbf{Y}_I \mid (G, \mathbf{x}, Id), \text{ for all triples } (G, \mathbf{x}, Id) \text{ and all } I \subseteq [n]\}$$

enables to characterize whether or not the distribution given by Eq. (4.4) transmits "signals" at distance larger than t. Informally, if the set of balls of radius t, centered at nodes with identities in I, is the same for two configurations $(G, \mathbf{x}, Id)$ and $(G', \mathbf{x}', Id')$, then the marginal distributions of the outputs of the nodes in I should be identical in both configurations. This is formally captured by the following definition, derived from [15, 51]. For any two random variables U and U', let $U \sim U'$ denotes the fact that U and U' are identically distributed.

Definition 10 *Let $t \geq 0$. A distribution $\mathbf{Y}$ described by its conditional distributions as in Eq.* (4.4) *is* non-signaling *at distance more than t if, for every positive integer n, for every two n-node graphs G and G' with respective identity assignments Id and Id$'$, and respective inputs $\mathbf{x}$ and $\mathbf{x}'$ given to their nodes, and for every $I \subseteq [n]$, the marginal distribution $\mathbf{Y}_I$ satisfy the following:*

$$B_{G,\mathbf{x},Id}(i,t) = B_{G',\mathbf{x}',Id'}(i,t) \textit{ for all } i \in I \implies \mathbf{Y}_I \mid (G, \mathbf{x}, Id) \sim \mathbf{Y}_I \mid (G', \mathbf{x}', Id'). \tag{4.5}$$

In other words, to be non-signaling at distance more than t, the output distribution $\mathbf{Y}$ must satisfy that if $B_{G,\mathbf{x},Id}(i,t) = B_{G',\mathbf{x}',Id'}(i,t)$ for every $i \in I$, then, for every $|I|$-dimensional vector $\mathbf{z}$ with coordinate in $\{0,1\}^*$, we must have

$$\Pr[\mathbf{Y}_I = \mathbf{z} \mid (G, \mathbf{x}, Id)] = \Pr[\mathbf{Y}_I = \mathbf{z} \mid (G', \mathbf{x}', Id')].$$

In particular, from the definition above, if $B_{G,Id,x}(i,t) = B_{G',Id',x'}(i,t)$ for some $i \in [n]$, then the non-signaling condition states that the distributions of the outputs at node i must be identical in $(G, \mathbf{x}, Id)$ and $(G', \mathbf{x}', Id')$. The non-signaling condition is actually more constrained by requesting that the property should hold not only

at every individual node, but also for all possible scales of computation, that is, for every subset $I \subseteq [n]$, to capture the case of pairwise independent events that are not mutually independent. The condition $B_{G,\mathbf{x},Id}(i,t) = B_{G',\mathbf{x}',Id'}(i,t)$ for every $i \in I$ states that, for every node with identity $i \in I$, all the information this node can gather in t rounds is identical in both instances $(G, \mathbf{x}, Id)$ and $(G', \mathbf{x}', Id')$. The non-signaling condition states that whenever this is the case, the marginal distributions of the outputs at the nodes with identities in I must be identical in both instances $(G, \mathbf{x}, Id)$ and $(G', \mathbf{x}', Id')$, which is expressed in Eq. (4.5).

Remark. Consider a distribution $\mathbf{Y}$ that violates Eq. (4.5). It means that there exists a set I, and two distinct instances $(G, \mathbf{x}, Id)$ and $(G', \mathbf{x}', Id')$ such that $B_{G,\mathbf{x},Id}(i,t) = B_{G',\mathbf{x}',Id'}(i,t)$ for all $i \in I$ while $\mathbf{Y}_I \mid (G, \mathbf{x}, Id)$ distributes differently from $\mathbf{Y}_I \mid (G', \mathbf{x}', Id')$. In other words, the "behavior" of the nodes with identities in I differ in $(G, \mathbf{x}, Id)$ and $(G', \mathbf{x}', Id')$, as witnessed by the fact that the output vector $\mathbf{Y}_I$ is not distributed the same in both instances, whereas the "view" at distance t of the nodes in I are identical in both instances. As a consequence, for such a distribution, it means that some "signal" from nodes at distance greater than t reaches some nodes with identities in I, in no more than t rounds. This is not possible, unless the distribution violates causality.

4.3.2 Non-classical computing

By definition, every output distribution resulting from the execution of a distributed algorithm using shared randomness, and performing in at most t rounds, is non-signaling at distance greater than t. This is because, as illustrated by Eq. (4.3), the output of every node i is precisely conditioned on $B(i,t)$, and on the value of the shared random variable λ, and the output $\mathbf{Y}$ is simply the joint distribution of the marginal outputs $\mathbf{Y}_i$. Hence, if $B_{G,\mathbf{x},Id}(i,t) = B_{G',\mathbf{x}',Id'}(i,t)$ for all $i \in I$, then we get the desired non-signaling equality $\mathbf{Y}_I \mid (G, \mathbf{x}, Id) \sim \mathbf{Y}_I \mid (G', \mathbf{x}', Id')$.

Bell's inequalities (cf. the next section) precisely states that no *classical* distributed algorithms (i.e., algorithms based on classical physics) can yield all distributions to be observed in quantum mechanics [18], and such distributions have been experimentally observed (see [48] for the first experimental test of the violation of

Bell's inequalities), demonstrating that we may not live in a world governed by classical physics. In particular, it is known that quantum effects generate correlations enabling to go beyond the distribution of Eq. (4.3). A very basic example explored exhaustively in this chapter is the scenario in which two processes Alice and Bob have access to some variables u and v, respectively, correlated in a way consistent with quantum physics. This correlation may not be captured by shared randomness. Thus, an algorithm run by Alice and Bob with access to the variable u at A, and v at B, might be able to solve tasks quicker, or with higher success probability than when the algorithm is restricted to the access to private and/or shared random variables only. How to exploit the power of probabilistic correlations beyond the ones resulting from using shared randomness is the the purpose of *quantum distributed computing*.

Quantum distributed computing is rather in its infancy, but we have already mentioned a set of quite significant contributions. Designing efficient quantum distributed algorithms is a challenge, and requires a very fine and deep skill in the manipulation of quantum correlations. There are good news though: designing *lower bounds* for quantum distributed computing may not be as hard as designing algorithms, at least in some frameworks. More precisely, there are several tasks for which lower bounds or impossibility results designed for classical algorithms translate easily to the context of quantum computing. This statement of facts is thanks to non-signaling distributions, as defined in Definition 10.

Indeed, consider a distribution $\mathbf{Y}$ that is non-signaling at distance greater than t. Define the success probability of $\mathbf{Y}$ for a task T as follows. The success probability of $\mathbf{Y}$ for $(G, \mathbf{x}, Id)$ is the sum, over all *valid* outputs *out* for T with respect to the configuration $(G, \mathbf{x}, Id)$, of the probability of *out*. That is:

$$\Pr[\mathbf{Y} \text{ succeeds for } (G, \mathbf{x}, Id)] = \sum_{\text{valid } out} \Pr[\mathbf{Y} = out \mid (G, \mathbf{x}, Id)].$$

Then, the success probability of $\mathbf{Y}$ for the task T is the infimum, taken over all configurations $(G, \mathbf{x}, Id)$, of the success probability of $\mathbf{Y}$ for $(G, \mathbf{x}, Id)$. Now, quantum physics does not violate causality, and all distributions generated by quantum distributed algorithms communicating at distance no more than t are non-signaling at distance greater than t. Hence, given a task T, proving that there are no distributions non-signaling at distance greater than t with success probability greater than p for T

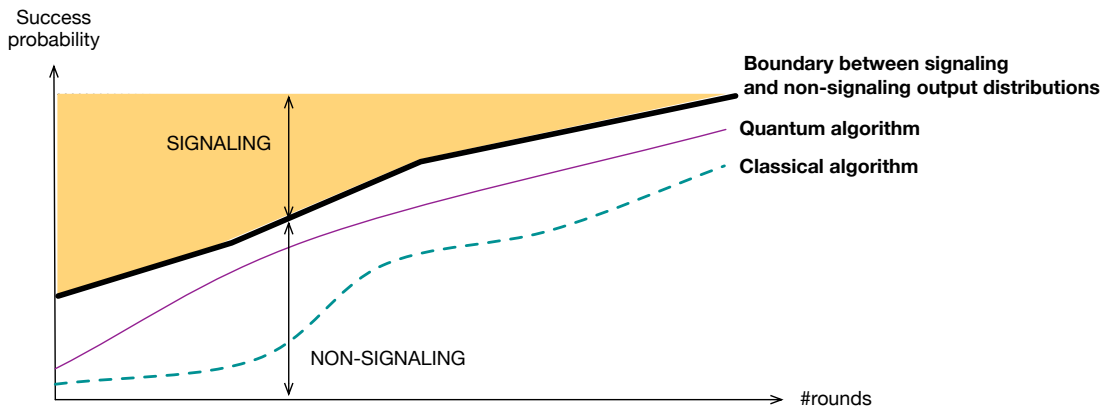


Figure 4-2: *Quantum resources enable to design distributed algorithms that are at least as efficient as any classical algorithm (even those using shared randomness), but the output distribution of any distributed quantum algorithm cannot offer better tradeoff between success probability and number of rounds than the best non-signaling distribution.*

enables to prove that every distributed quantum algorithm solving T with probability at least p runs in at least t rounds.

The following result is folklore. It simply states that quantum mechanics does not violate causality.

Theorem 5 (see, e.g., [51]) *For any $t \geq 0$, any output distribution produced by a quantum distributed algorithm performing t communication rounds is non-signaling at distance greater than t.*

As a consequence, we immediately get:

Corollary 1 *Let T be a task, and let $t \geq 0$. Assume that, for every distribution $\mathbf{Y}$ that is non-signaling at distance greater than t, the success probability of $\mathbf{Y}$ for T is at most p. Then, there are no quantum distributed algorithms enabling to solve task T with probability more than p in t communication rounds.*

Figure 4-2 displays an abstract graphical representation of the tradeoff between time complexity and success probability for various kinds of algorithms. We stress the fact that the world "quantum" in the statement of Corollary 1 can be replaced by any distributed computing model M, whether it be weaker, stronger, orthogonal, or inconsistent with quantum computing, as long as the model M is non-signaling.

4.3.3 Examples and applications

We give in this section a first illustrative example of our model. Further examples and applications are given in the next chapter.

The following example is borrowed from [51], by the courtesy of the authors: Cyril Gavoille, Adrian Kosowski, and Marcin Markiewicz. Consider the task of 2-coloring the nodes in a ring C_n with an even number n of nodes. Linial [68] has proved that, with a classical algorithm, $n/2-1$ rounds are necessary and sufficient for solving that task with probability 1. (The lower bound holds even for an algorithm using shared randomness, while the upper bound holds deterministically). Let us show that, as pointed out in [51], quantum algorithms cannot do much better in term of number of rounds, in the sense that at least $n/4-1$ rounds are required by such algorithms for 2-coloring rings with an even number of nodes. Moreover, this holds even if one just asks for a success probability greater than 1/2. To establish this claim, we use Corollary 1, and prove that any distribution $\mathbf{Y}$ that is non-signaling at distance greater that $n/4-1$ cannot solve 2-coloring with probability greater than 1/2.

Let $n = 4(t+1)$ for $t \geq 1$. Hence, two antipodal nodes in the n-node ring C_n (at distance $n/2$) satisfies that the balls of radius t centered at these nodes do not intersect, and are in fact separated by two nodes which belong to none of these two balls. (See Figure 4-3). We consider two configurations $(C_n, \varnothing, Id)$ and $(C_n, \varnothing, Id')$ (the coloring problem assumes no inputs). In $(C_n, \varnothing, Id)$, the nodes of the rings are labeled consecutively from 1 to n. In $(C_n, \varnothing, Id')$, nodes are also labeled consecutively from 1 to n, apart from one identity, $3(t+1)+1$, which is placed between $t+1$ and $t+2$, and identities $3(t+1)$ and $3(t+1)+2$ which are adjacent. Now, in the two configurations, the balls $B(1,t)$ are identical. Similarly, the balls $B(n/2+1,t)$ are identical in the two configurations. However, nodes 1 and $n/2+1$ are at even distance $2(t+1)$ in $(C_n, \varnothing, Id)$ while they are at odd distance $2t+1$ in $(C_n, \varnothing, Id')$.

Let $\mathbf{Y}$ be a non-signaling distribution with success probability p. It must be the case that, with probability at least p, nodes 1 and $n/2+1$ are colored the same in $(C_n, \varnothing, Id)$, but with different colors in $(C_n, \varnothing, Id')$. So, let $I = \{1, n/2+1\}$, and let us focus on $\mathbf{Y}_I$ conditioned on the two configurations $(C_n, \varnothing, Id)$ and $(C_n, \varnothing, Id')$. These distributions act on 2-dimensional boolean vectors $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2)$. Because of

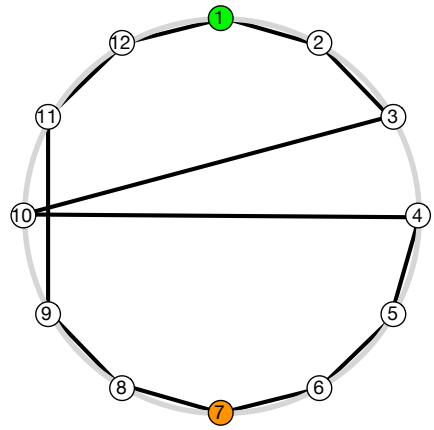


Figure 4-3: *Two rings on 12 nodes (in black, and in light grey). For $t = 2$, the balls $B(1, t)$ and the balls $B(7, t)$ are identical in both rings. In a proper 2-coloring, nodes 1 and 7 must have different colors in one ring, while they must have identical color in the other ring.*

the way $\mathbf{z}$ is distributed according to $\mathbf{Y}_I \mid (C_n, \varnothing, Id)$, we must have $\Pr[\mathbf{z}_1 = \mathbf{z}_2] \geq p$ because 1 and $n/2+1$ are colored the same in $(C_n, \varnothing, Id)$, with probability at least p. Similarly, because of the way $\mathbf{z}$ is distributed according to $\mathbf{Y}_I \mid (C_n, \varnothing, Id')$, we must have $\Pr[\mathbf{z}_1 \neq \mathbf{z}_2] \geq p$.

Now, the non-signaling condition actually states that these two conditional distributions $\mathbf{Y}_I \mid (C_n, \varnothing, Id)$ and $\mathbf{Y}_I \mid (C_n, \varnothing, Id')$ are identical. Therefore, $1 - p \geq p$, i.e., $p \leq 1/2$. Thus, by Corollary 1, no quantum algorithms can successively 2-color the ring of even size with probability greater than $1/2$ in less than $n/4 - 1$ rounds. It is worth noticing that the above lower bound exploits Corollary 1 maximally, in the sense that there exists a distribution $\mathbf{Y}$ that is non-signaling at distance greater that $n/4$, with success probability 1 for 2-coloring rings with even number of nodes (see [51]).

In the next chapter, we use the concepts introduced in this one, and specifically Corollary 1, to exhaustively study 2-player games, involving two parties that do not communicate, but share resources (e.g., random bits, quantum bits, etc.), and explore further computational consequences.

Chapter 5

Computational consequences of the Non-Signalling model

The main objective of this chapter is to provide illustrative examples of distributed computing problems for which it is possible to design tight lower bounds for *quantum* algorithms without having to manipulate concepts from quantum mechanics, at all. This is achieved by considering the *non-signaling model*, that is stronger than distributed quantum computing, but involves no concepts from quantum mechanics.

To demonstrate the consequences of this model on settings involving many agents, we start by addressing the following class of 2-player problems (see Figure 4-1). Alice (resp., Bob) receives a boolean x (resp., y) as input, and must return a boolean a (resp., b) as output. A *game* between Alice and Bob is defined by a pair (δ, f) of boolean functions. The objective of Alice and Bob playing game (δ, f) is, for every pair (x, y) of inputs, to output values a and b, respectively, satisfying

$$\delta(a, b) = f(x, y)$$

in *absence of any communication* between the two players. However, the two players have access to common resources such as a source of random bits, or a source of entangled quantum bits (qubits). The ability of solving a given game (δ, f) thus depends on the type of resources shared by Alice and Bob. It is known [26] that, for

the so-called CHSH game

$$a \oplus b = x \wedge y \ ,$$

the ability for the players to use entangled qubits helps. We show that, apart from the CHSH game, and games equivalent to the CHSH game[1], quantum correlations do not help. That is, for every game non equivalent to the CHSH game, there exists a classical protocol (using shared randomness) whose probability of success is at least as large as the one of any protocol using quantum resources. This result holds for both worst case and average case analysis.

5.1 Two-player games

We consider the following 2-player problem. Alice (resp., Bob) receives a boolean x (resp., y) as input, and must return a boolean a (resp., b) as output. A *game* between Alice and Bob is defined by a pair (δ, f) of boolean functions. The objective of Alice and Bob playing game (δ, f) is, for every inputs x and y, to output values a and b satisfying

$$\delta(a, b) = f(x, y)$$

in *absence of any communication* between the two players. Obviously, the game is trivial whenever there exist two boolean functions α and β such that $\delta(\alpha(x), \beta(y)) = f(x, y)$ for every pair $(x, y) \in \{0, 1\}^2$. Indeed, for such games, there exists a deterministic distributed protocol solving the game, with Alice returning $\alpha(x)$ on input x, and Bob returning $\beta(y)$ on input y. Non-trivial games may still be solved, but only under some probabilistic guarantees. A game (δ, f) is said to be solvable with probability p if there exists a (randomized) distributed protocol such that Alice outputs a, and Bob outputs b, with

$$\Pr(\delta(a, b) = f(x, y)) \geq p \tag{5.1}$$

for every input pair $(x, y) \in \{0, 1\}^2$.

Different sources of randomness can then be considered. Classical sources of randomness (i.e., not using quantum effects) include the case where each of the two

[1] E.g., the game $a \oplus b = \bar{x} \wedge \bar{y}$ is equivalent to the CHSH game, as Alice and Bob just have to complement their respective inputs, and solve the CHSH game on these complemented inputs.

players are provided with individual independent sources of random bits. They also include shared randomness where, in addition to individual independent sources of random bits, the two players have access to a common source of random bits. Rewriting Eq. (4.3) in the context of 2-player games, shared randomness enables to produce outputs satisfying

$$\Pr(a, b|x, y) = \sum_{\lambda \in \Lambda} \Pr(a|x, \lambda) \cdot \Pr(b|y, \lambda) \cdot \Pr(\lambda) \tag{5.2}$$

where the random variable λ is drawn from some probability space Λ, and $\Pr(a, b|x, y)$ denotes the probability that Alice outputs a and Bob outputs b, given the fact that Alice receives x as input, and Bob receives y as input. It is known [18] that correlations on quantum entangled states enable to derive protocols whose output distribution cannot be modeled as Eq. (5.2). One evidence of this fact is the CHSH game [26]:

$$a \oplus b = x \wedge y$$

where $\oplus$ denotes the exclusive-or operator. CHSH can be solved with probability $\cos^2(\pi/8) > \frac{3}{4}$ with a quantum protocol (i.e., a protocol in which Alice and Bob have access to entangled quits) [25], while every protocol using classical shared randomness cannot solve CHSH with probability more than $\frac{3}{4}$. In fact, the literature dealing with 2-player games [23] refers to objects called *boxes*. A box B is characterized by the probabilities $\Pr(a, b|x, y)$ of outputting pair (a, b) given the input pair (x, y), for all $a, b, x, y \in \{0, 1\}$. A box B is thus described by a set

$$\{\Pr(\cdot, \cdot|x, y),\ (x, y) \in \{0, 1\}^2\} \tag{5.3}$$

of four probability distributions on $\{0, 1\}^2$, one for each pair $(x, y) \in \{0, 1\}^2$. Hence, there are infinitely many boxes, with different computational powers.

The absence of communication between the two players along with the assumption of causality are captured by the class of *non-signaling* boxes, defined according to Definition 10. A box B is non-signaling if and only if it satisfies that the marginal output distributions for Alice and Bob depend only on their respective inputs. Formally,

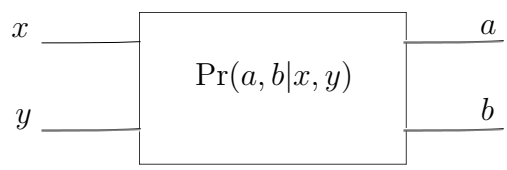


Figure 5-1: Non-signalling 2-player box where Pr is a non-signalling probability distribution.

a 2-player non-signaling box satisfies:

$$\begin{aligned} &\forall a, x,\ \sum_b \Pr(a,b|x,0) = \sum_b \Pr(a,b|x,1), \\ \text{and}\quad &\forall b, y,\ \sum_a \Pr(a,b|0,y) = \sum_a \Pr(a,b|1,y). \end{aligned} \tag{5.4}$$

Non-signaling boxes satisfying Eq. (5.2) are called *local*, where "locality" is referring here to the physical science concept of *local hidden variables* [18, 36]. Boxes that do not satisfy Eq. (5.4) are *signaling*.

The set of all boxes has a geometric interpretation [15], because it forms a 12-dimensional convex polytope corresponding to the four sets in Eq. 5.3, where each set is entirely described by three values (as the 4th is the sum of the first three). This polytope includes the convex polytope of non-signaling boxes, which includes in turn the convex local polytope. Fig. 5-2 provides an abstract representation of the non-signaling polytope. It is known [16] that each of the extremal vertices of the non-signaling polytope is equivalent (up to individual reversible transformations on the inputs and outputs) to the so-called PR box [25, 76], that is described by the distribution:

$$\Pr(a,b|x,y) = \begin{cases} \frac{1}{2} & \text{if } a \oplus b = x \wedge y \\ 0 & \text{otherwise.} \end{cases}$$

Notice that the PR box satisfies $\Pr(a \oplus b = x \wedge y) = 1$ for every input pair x, y. So, in particular, it solves the CHSH game with probability 1. Each of the extremal vertices of the local polytope can be implemented by a deterministic protocol: they are "equivalent" to the identity box ID described by $\Pr(a,b|x,y) = 1$ if and only if $a = x$ and $y = b$. Every non-extremal box B is a linear combinations of extremal boxes: $B = \sum_{i=1}^{k} \beta_i B_i$ where B_i is an extremal box, $\sum_{i=1}^{k} \beta_i = 1$, and $\beta_i > 0$ for every $i = 1, \ldots, k$. On Fig. 5-2, the doted line represents the limit of the class of

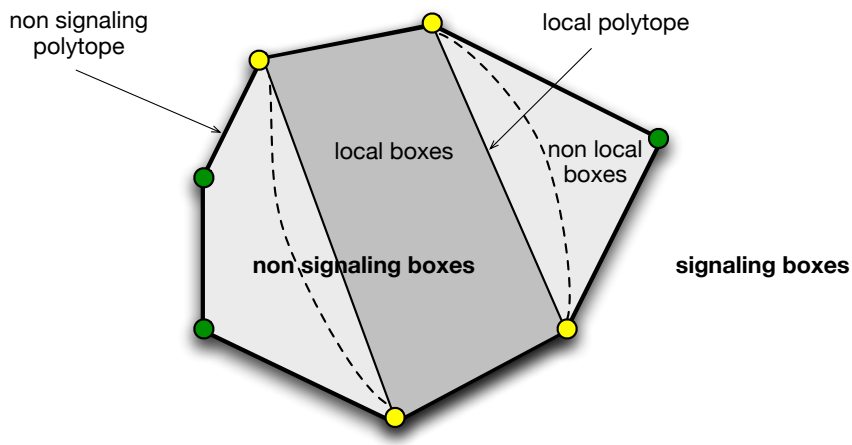


Figure 5-2: *Abstract representation of the non-signaling polytope, including the polytope of local boxes. The dotted line indicates the boundary between boxes which can be implemented by a quantum algorithm, and boxes which cannot.*

boxes implementable by a quantum protocol. This latter class strictly contains the local boxes, and is strictly included in the class of non-signaling boxes, as witnessed by the CHSH game.

Our objective of identifying the games for which quantum correlations help can be reformulated as follows. Given a box implementable by a quantum protocol, which games can be efficiently solved using this box? Stated differently, given a game, what are the boxes implementable by a quantum protocol that enable to solve that game with better guarantees than any local boxes?

5.2 Equivalence classes of games

As introduced in the previous section, a game between Alice and Bob is described by a pair (δ, f) of boolean functions on two variables. Playing the game means for Alice (resp. Bob) to receive a boolean x (resp., y) as input, and to return a boolean a (resp., b) as output such that $\delta(a, b) = f(x, y)$ without communication between the two players. Examples of games are

$$\text{EQ}: \quad a \wedge b = \overline{x \oplus y} \quad \text{and} \quad \text{NEQ}: \quad a \wedge b = x \oplus y.$$

Another example of a game is :

$$\text{AMOS}: \quad a \wedge b = \overline{x \wedge y}.$$

In these three examples, one can view the games as Alice and Bob respectively deciding whether the equality $x = y$ holds, whether the non-equality $x \neq y$ holds, and whether there is "at most one selected" player (a selected player has input 1). Here, "deciding" means that if the answer is "yes" then both players should output "yes", while if the answer is "no" then at least one player should output "no". In fact, the three games EQ, NEQ, and AMOS, are AND-games, in the sense that δ is the conjunctive operator. However, all games are not of that type. In particular, we shall see that the already mentioned CHSH game

$$a \oplus b = x \wedge y$$

is not an AND-game, since it is not equivalent to any game (δ, f) where δ is the conjunctive operator. More precisely, for any game (δ, f), both functions δ and f can be rewritten as:

$$\delta(a,b) = \alpha_{1,1}ab + \alpha_{1,0}a + \alpha_{0,1}b + \alpha_{0,0} \text{ and } f(x,y) = \beta_{1,1}xy + \beta_{1,0}x + \beta_{0,1}y + \beta_{0,0}$$

where the $+$ symbol denotes the exclusive-or operator $\oplus$, the (omitted) $\cdot$ symbol denotes the and-operator $\wedge$, and all coefficients are in $\{0,1\}$. We say that two games (δ, f) and (δ', f') are equivalent if

$$\delta(a,b) = \delta'(A,B) \text{ and } f(x,y) = f'(X,Y)$$

where A (resp., B, X, Y) is a degree-1 polynomial in a (resp., b, x, y) with coefficients in $\{0,1\}$. Whenever two games are equivalent, any protocol solving one of the two games can be used for solving the other game, by performing individual reversible transformations on the inputs and outputs. The probability of success for the two games will be identical. The same notion of equivalence can be defined for boxes. Now we can state formally that the CHSH game is not equivalent to any of the three AND-games: EQ, NEQ, or AMOS. This is because, as we will see further in the text, none of these latter games can be solved with probability 1 by a non-signaling box (as

opposed to the CHSH game which can be solved with probability 1 by the PR box). Instead, EQ and NEQ are equivalent games. Indeed, for NEQ, $f(x, y) = x + y$, while, for EQ, $f(x, y) = x + (y + 1)$.

Definition 11 *A game (δ, f) is an* XOR*-game if and only if it is equivalent to a game (δ', f') where $\delta'(a, b) = a \oplus b$.*

5.3 On the power of quantum correlations

In this section, we establish our main result, stating that correlations on quantum entangled states do not help for solving 2-player games that are not equivalent to an XOR-games. In fact we establish a stronger result by showing that non-signaling boxes do not help for those games.

Theorem 6 *Let (δ, f) be a 2-player game that is not equivalent to any* XOR*-game. Let p be the largest success probability for (δ, f) over all local boxes. Then every box solving (δ, f) with probabilistic guarantee $> p$ is signaling.*

Proof. The proof is straightforward for games (δ, f) where δ does not depend on both a and b. Indeed, on the one hand, if δ is constant, say $\delta(a, b) = \alpha$ for some $\alpha \in \{0, 1\}$, for all (a, b), then the game is either impossible (whenever $\exists x, y : f(x, y) \neq \alpha$) or trivial (whenever $\forall x, y, f(x, y) = \alpha$). And, on the other hand, if δ depends on only one of its two variables, say $\delta(a, b) = a + \alpha$ for some $\alpha \in \{0, 1\}$, then the game is again either trivial, or equivalent to a single-player game where the player must compute a 2-variable function $f(x, y)$ knowing only one of the variables. Games of that latter class are equivalent to either the game $a = y$ or the game $b = x$. Non-signaling boxes do not help for such games: the best probability of success is $\frac{1}{2}$ (which is achievable by a classical algorithm using randomization). Indeed, consider the game $a = y$, and, for that game, consider the instance $x = 0$ and $y = 0$. Then, a box B with success probability p satisfies that $\Pr[\text{success for input } (0, 0)] \geq p$. Now,

$$\Pr[\text{success for input } (0, 0)] = \sum_b \Pr(0, b|0, 0),$$

and the non-signaling condition in Eq (5.4) states that $\sum_b \Pr(0, b|0, 0) = \sum_b \Pr(0, b|0, 1)$. The latter sum is the probability of failure for the input $(0, 1)$, which is at most $1 - p$.

Therefore, $p \leq 1 - p$, and thus $p \leq \frac{1}{2}$.

Therefore, we focus now on "true" 2-player games, i.e., games (δ, f) where δ depends on both a and b. First, we show that every true 2-player game (δ, f) which is not equivalent to an XOR-game is either deterministic, or equivalent to NEQ or AMOS. To establish this claim, observe that if f is constant, or depends on only one of the the two inputs, then the game (δ, f) can be solved with probability 1, by a deterministic protocol. Indeed, assume, without loss of generality, that f depends only on x. (The case f constant is straightforward). Then Alice and Bob can agree beforehand on a fixed value b^* for b. It follows that, knowing b^*, f, and δ, Alice can output a such that $\delta(a, b^*) = f(x)$.

We now come to the interesting case, that is, when both δ and f depend on their two inputs. Any 2-variable boolean function g can be rewritten as :

$$g(u, v) = U + V \quad \text{or} \quad g(u, v) = UV \quad \text{or} \quad g(u, v) = UV + 1$$

where U (resp., V) is a polynomial in u (resp., v) of degree at most 1, with coefficients in $\{0, 1\}$. Given that fact, we rewrite any game (δ, f) using two expressions from the above, one for δ, and the other for f. We thus get nine different types of games, which can be narrowed down to five types by noticing that games like $A + B = XY + 1$ are the same as games like $A' + B' = X'Y'$, up to the (reversible) transformation $B' = B + 1$. These five types of games are the following:

$$\begin{aligned}
\delta(a, b) = A + B &= f(x, y) = X + Y \\
\delta(a, b) = A + B &= f(x, y) = XY \\
\delta(a, b) = AB &= f(x, y) = X + Y \\
\delta(a, b) = AB &= f(x, y) = XY \\
\delta(a, b) = AB &= f(x, y) = XY + 1
\end{aligned}$$

Since f (resp., δ) depends on both x and y (resp., both a and b), all polynomials A, B, X, and Y in the above five types of games are of degree exactly 1, hence making all transformations reversible. Therefore, if two games can be rewritten into the same type, then they are equivalent. Table 5.1 describes the equivalence classes over the set of games formed by the five types above, and provides a representative for each

class.

	Form of the class	Representative of the class
Deterministic	$AB = XY$	PROD $a \wedge b = x \wedge y$
	$A + B = X + Y$	SUM $a \oplus b = x \oplus y$
Not deterministic	$A + B = XY$	CHSH $a \oplus b = x \wedge y$
	$AB = X + Y$	NEQ $a \wedge b = x \oplus y$
	$AB = XY + 1$	AMOS $a \wedge b = \neg(x \wedge y)$

Table 5.1: Equivalence classes for 2-player games depending on both inputs. The first two classes of games are deterministic, i.e., can be solved by a deterministic protocol. Instead, the last three classes are not deterministic (no deterministic protocols can solve any of the games in these three classes).

The theorem holds for games PROD and SUM since both of them can be solved by a deterministic protocol. Every game that is neither deterministic nor equivalent to an XOR-game is equivalent to the AND-game NEQ or AMOS. We now show that non-local boxes fail to solve AMOS or NEQ with higher probabilistic guarantee than what can be achieved with local boxes.

Let us first examine AMOS. We start by showing that any box that solves AMOS with probabilistic guarantee $p > \frac{2}{3}$ is signaling. Suppose that there exists a non-signaling box B, defined by the correlation $\Pr(a, b|x, y)$, that solves AMOS with probability p. On the one hand, for any probability distribution $\pi = \{\pi_{xy}|(x, y) \in \{0, 1\}^2\}$ of the inputs, we have

$$\sum_{xy} \pi_{xy} \Pr(\text{success for input } (x, y)) \geq p$$

On the other hand, we have

$$\sum_{xy} \pi_{xy} \Pr(\text{success for input } (x, y)) = \sum_{xy} \pi_{xy} \sum_{ab} \mathbb{1}_{\{a \wedge b = \neg(x \wedge y)\}} \Pr(a, b|x, y)$$

where $\mathbb{1}_{\{a\wedge b=\neg(x\wedge y)\}}$ denotes the boolean indicator function of whether $a\wedge b=\neg(x\wedge y)$ is true or not. Let us consider the following distribution π^*:

$$\pi^*_{00}=0 \quad \text{and} \quad \pi^*_{xy}=\frac{1}{3} \text{ for all } (x,y)\neq(0,0)$$

Let $p_{abxy}=\Pr(a,b|x,y)$ for box B. The probability of success with the input distribution π^* satisfies

$$\begin{aligned}\sum_{xy}\pi^*_{xy}\Pr(\text{success for }(x,y)) &= \frac{1}{3}\sum_{xy\neq 00}\mathbb{1}_{\{a\wedge b=\neg(x\wedge y)\}}p_{abxy}\\ &= \frac{1}{3}\left(p_{1101}+p_{1110}+\sum_{ab\neq 11}p_{ab11}\right)\\ &= \frac{1}{3}\left(p_{1101}+p_{1110}+p_{0011}+p_{0111}+p_{1011}\right) \quad (5.5)\end{aligned}$$

The non-signaling conditions (cf., Eq. (5.4)) require that, for every a,b,x,y,

$$\begin{aligned} p_{a0x0}+p_{a1x0} &= p_{a0x1}+p_{a1x1}\\ \text{and}\quad p_{0b0y}+p_{1b0y} &= p_{0b1y}+p_{1b1y}\end{aligned}$$

This gives a bound on the first two terms of Eq. (5.5):

$$\begin{aligned} p_{1101} &= p_{1111}+p_{0111}-p_{0101} \leq p_{1111}+p_{0111}\\ \text{and}\quad p_{1110} &= p_{1111}+p_{1011}-p_{1010} \leq p_{1111}+p_{1011}\end{aligned}$$

The probability p of success is therefore bounded by :

$$\begin{aligned} p &\leq \frac{1}{3}\left(p_{1111}+p_{0111}+p_{1011}+p_{1111}+p_{0011}+p_{0111}+p_{1011}\right)\\ &\leq \frac{1}{3}\Big(2\sum_{ab}p_{ab11}-p_{1111}\Big)\\ &\leq \frac{2}{3}\end{aligned}$$

Indeed, $\sum_{ab}p_{abxy}=1$ for any fixed (x,y), and $p_{1111}\geq 0$. Therefore, every non-signaling box solves AMOS with success at most $\frac{2}{3}$.

Regarding NEQ, we observe that with distribution π^* that discards the input $(0,0)$, AMOS and NEQ become the same games:

$$f_{\text{AMOS}}(x,y) = f_{\text{NEQ}}(x,y)$$

for all $(x,y) \neq (0,0)$. As a consequence, the same bound $\frac{2}{3}$ also holds for NEQ: every non-signaling box solves NEQ with success at most $\frac{2}{3}$.

We now show that the bound $\frac{2}{3}$ for AMOS and NEQ can be reached by local boxes. For this purpose, we describe a protocol using solely shared randomness, and reaches success probability $\frac{2}{3}$. Let a_0 and a_1 (resp., b_0 and b_1) be the outputs of Alice (resp. Bob) on the respective input $x = 0$ and $x = 1$ (resp., $y = 0$ and $y = 1$). AMOS translates into solving the system:

$$\begin{cases} a_0 \cdot b_0 = 1 \\ a_0 \cdot b_1 = 1 \\ a_1 \cdot b_0 = 1 \\ a_1 \cdot b_1 = 0 \end{cases} \tag{5.6}$$

and NEQ translates into :

$$\begin{cases} a_0 \cdot b_0 = 0 \\ a_0 \cdot b_1 = 1 \\ a_1 \cdot b_0 = 1 \\ a_1 \cdot b_1 = 0 \end{cases} \tag{5.7}$$

The second and third equations of the system for AMOS as well as for NEQ imply that $a_0 = a_1 = b_0 = b_1 = 1$, resulting in the last equation impossible to be satisfied in both games. Hence the fourth equation of each system cannot be simultaneously satisfied with those two equations. Instead, if one chooses to ignore one of them, then one can find a solution to the game. Playing any one of the two games using shared randomness, we allow Alice and Bob to have access, before knowing their inputs, to a shared random variable λ uniformly distributed in $\{1,2,3\}$, designating the equation to be ignored among the last three ones. Alice and Bob will fail to solve the game with probability at most $\frac{1}{3}$ (when the ignored equation is precisely the one corresponding to the actual inputs), making the success probability for any input (x,y) equal to $\frac{2}{3}$. This completes the proof of the theorem. $\square$

It turns out that even relaxing the constraints placed on solving the game, by considering average case analysis, does not allow non-signaling boxes to perform better than local boxes on games not equivalent to XOR-games.

Theorem 7 *Let (δ, f) be a 2-player game that is not equivalent to any* XOR*-game. Let p be the largest* average *success probability for (δ, f) over all local boxes. Then every box solving (δ, f) with average probabilistic guarantee $> p$ is signaling.*

Proof. Using the same arguments as in the proof of Theorem 6, we limit the analysis to AMOS and NEQ. For average case analysis, we consider these two games with input probability distribution $\pi_{xy} = \frac{1}{4}$ for every $(x, y) \in \{0, 1\}^2$. The success probability for Alice and Bob with this input distribution is then given by:

$$\Pr(\text{success}) = \frac{1}{4} \sum_{x,y} \sum_{a,b} \mathbb{1}_{\{\delta(a,b)=f(x,y)\}} \Pr(a, b|x, y)$$

First, we show that the protocol described in the proof of Theorem 6 for solving AMOS and NEQ has average success probability $\frac{3}{4}$. Indeed, the success probability of that protocol can be written as:

$$\Pr(\text{success}) = \frac{1}{4} \sum_{x,y} \Pr(\text{success}(x, y)) = \frac{1}{4}\Big(1 + \frac{2}{3} + \frac{2}{3} + \frac{2}{3}\Big) = \frac{3}{4}$$

because, the protocol always satisfies the first equation of both games, and satisfies each of the three other equations (of both games) with probability $\frac{2}{3}$.

Next, we show that a non-local box cannot solve AMOS or NEQ with average success probability greater than $\frac{3}{4}$. Indeed, we have

$$\begin{aligned} \Pr(\text{success}) &= \frac{1}{4}\Big[\Big(\sum_{(x,y)\neq(0,0)} \sum_{a,b} \mathbb{1}_{\{\delta(a,b)=f(x,y)\}} \Pr(a, b|x, y)\Big) \\ &+ \Big(\sum_{a,b} \mathbb{1}_{\{\delta(a,b)=f(0,0)\}} \Pr(a, b|0, 0)\Big)\Big] \end{aligned}$$

The first term is the same as the one analyzed in the proof of Theorem 6, where it was proved to be at most 2. The second term is at most $\sum_{ab} \Pr(a, b|0, 0) \leq 1$. Therefore, the average success probability for non-signaling boxes is at most $\frac{3}{4}$. $\square$

The practical interest of the previous two theorems comes from their consequence to distributed quantum computing. By Corollary 1, we get:

Corollary 2 *Quantum correlations does not help for solving 2-player games that are not equivalent to any* XOR*-game. This limitation holds for both worst case, and average case analysis.*

5.4 Further work and open problems

5.4.1 n-player games

One obvious generalization of the 2-player games is to consider games with more than two players, with IDs from 1 to $n \geq 2$. In the n-player game (δ, f), Player i receives boolean x_i as input, and must return a boolean a_i such that

$$\delta(a_1, \ldots, a_n) = f(x_1, \ldots, x_n)$$

in absence of communication between the players. As for two players, two classes of games deserve specific interest:

- XOR-games: $\delta(a_1, \ldots, a_n) = a_1 \oplus \cdots \oplus a_n$, for they generalize the CHSH game, and for they can be solved by a non-signaling box implementable by a circuit of PR boxes (see [16]);
- AND-games: $\delta(a_1, \ldots, a_n) = a_1 \wedge \cdots \wedge a_n$ for they correspond to the standard decision mechanism in the distributed computing literature (see, e.g., [45, 72]).

In particular, the n-player variant of AMOS is:

$$\bigwedge_{i=1}^{n} a_i = \bigwedge_{i \neq j} (\overline{x_i \wedge x_j}).$$

There exists a randomized protocol (see [45]), that is using individual random coins, and solves AMOS with success guarantee $\frac{\sqrt{5}-1}{2} \geq 0.61 > 1/2$. In this protocol, every selected player (i.e., with input 1) outputs 1 with probability p, and 0 with probability $1 - p$, where p is to be fixed later. Every non-selected player (i.e., with input 0)

systematically outputs 0. Hence, if no players are selected, then the protocol always outputs the right answer. If one player is selected, then the protocol fails with probability $1-p$, while if two or more players are selected then the protocol fails with probability at most p^2. Solving $p^2 = 1-p$ results in picking the optimal probability $p^* = \frac{\sqrt{5}-1}{2}$.

On the other hand, we have seen in this paper that AMOS can be solved with success guarantee $\frac{2}{3} > p^*$ by two players applying a probabilistic protocol using shared randomness. One can actually show that the same guarantee can be achieved with three players, by analyzing the following system

$$\begin{cases} a_0 \cdot b_0 \cdot c_0 = 1 \\ a_1 \cdot b_0 \cdot c_0 = 1 \\ a_0 \cdot b_1 \cdot c_0 = 1 \\ a_0 \cdot b_0 \cdot b_1 = 1 \\ a_1 \cdot b_1 \cdot c_0 = 0 \\ a_1 \cdot b_0 \cdot c_1 = 0 \\ a_0 \cdot b_1 \cdot c_1 = 0 \\ a_1 \cdot b_1 \cdot b_1 = 0 \end{cases}$$

which lists the eight equations for AMOS corresponding to the eight possible inputs of the games. Consider the protocol which solves that system after ignoring the second and seventh equations with probability $\frac{1}{3}$, the third and sixth with probability $\frac{1}{3}$, and the fourth and fifth with probability $\frac{1}{3}$. This protocol has success probability at least $\frac{2}{3}$ for every triple of inputs.

Unfortunately, the protocols for two and three players do not seem to extend easily to a higher number of players. For four players, we have designed an ad hoc probabilistic protocol using shared randomness, with success probability $\frac{9}{14} > \frac{\sqrt{5}-1}{2}$, but we failed to design a local protocol with success probability $\frac{2}{3}$. For more than four players, the ad hoc protocol could be generalized, but we have not identified a general pattern for it.

Instead, the lower bound $\frac{2}{3}$ on the probability of success for solving AMOS with non-signaling boxes established in this paper trivially extends to n players. We thus conclude by stating the following problem.

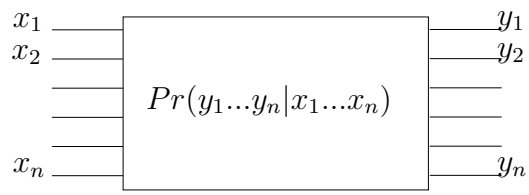


Figure 5-3: A non-signalling box is a device shared by n players defined by a non-signalling probability distribution.

Open problem 1: Prove or disprove the existence of a shared-randomness probabilistic protocol that solves the n-player AMOS game with success probability $\frac{2}{3}$, for all $n \geq 2$.

5.4.2 n-player boxes and the $\mathcal{LOCAL}$ model

Since non-signalling captures the very impact of locality constraints, another generalization is to consider the computational power of non-signalling boxes in the $\mathcal{LOCAL}$ model. We need for that to consider n-player boxes shared by the n nodes of the system. In non-signalling boxes for n-players (Figure 5-3) each player i hands an input x_i in $\{0,1\}^*$ to the box, and receives an output y_i such that the probability of the global output $\mathbf{Y}$ respects the no-communication constraint between the players. A non-signalling probability distribution is thus defined according to Definition 10 for $t = 0$, i.e., for n players with an input vector $\mathbf{x}$ and a random output vector $\mathbf{Y}$, where $\mathbf{Y}$ distributes such that for all $I \subseteq [n]$, for all $|I|$-dimensional vector $\mathbf{z}$ and for all input vectors $\mathbf{x}$ and $\mathbf{x}'$:

$$\text{if} \quad \forall i \in I, \quad \mathbf{x}_i = \mathbf{x}'_i \quad \text{then} \quad \Pr[\ \mathbf{Y}_I = \mathbf{z} \mid \mathbf{x}\ ] = \Pr[\ \mathbf{Y}_I = \mathbf{z} \mid \mathbf{x}'\ ].$$

The most general framework of $\mathcal{LOCAL}$ computations with non-signalling resources that we could think of is probably a framework where nodes have access to arbitrarily many n-player non-signalling boxes at each round of calculation. However, such a framework turns out to be far more challenging than the classical $\mathcal{LOCAL}$ model as it seems unclear whether algorithms could benefit from a simulation similar to the one described in Section 1.2.1, where the computation part is postponed after the communication part. In other words, let us define, say, a *postfix form* of protocols using non-signalling boxes as the protocols where nodes perform a local

algorithm $\mathcal{A}$ in time t as described in 1.2.1 , and after t rounds, have access to a single non-signalling box. We could then ask the following question:

Open problem 2: Could any non-signalling local protocol be simulated by a postfix non-signalling protocol ?

If the answer to the previous question is yes, that would open new perspectives for the design of lower bounds for non-signalling computations, and, by extension for quantum algorithms. For instance, we have seen in Section 4.3.3 that quantum resources would not be of much help as far as 2-coloring the even-size ring C_{2n} distributively is concerned. As pointed out in [51], the situation appears to be much more intriguing for 3-coloring the ring C_n. Indeed, the arguments used in [68] for establishing the $\Omega(\log^* n)$ lower bound do not seem to extend to non-signaling distributions. However, an approach using postfix non-signalling protocols seems to be promising, and will be at the core of our future work. Until then, it is still not clear whether or not there exists a quantum Las Vegas algorithm for 3-coloring C_n performing in $o(\log^* n)$ rounds. We believe that it is not the case. On the other hand, it may be the case that quantum resources would help in designing Monte Carlo algorithms with better success probabilities than classical (shared-randomness) algorithms.

Open problem 3: What is the complexity of 3-coloring C_n using n-player non-signalling boxes?

5.5 Conclusion

The results in this paper open new perspectives in term of distributed *checking*, a.k.a. distributed *verification*, which consists in having a set of, say, n processes deciding whether their global state (defined as the union of the local state of every individual process) satisfies some prescribed property, or not. The literature on this latter topic (see, e.g., [45, 47, 62, 72]) assumes a *decision* function δ which is applied to the set of individual decisions produced by the processes. Typically, each process should output a boolean b_i, and the global interpretation of the outputs is computed by

$$\delta(b_1, \ldots, b_n) = \bigwedge_{i=1}^{n} b_i \in \{\text{“yes”}, \text{“no”}\} \ .$$

The use of the AND operator is motivated by the requirement that the global state is valid if and only if all processes agree on some (local) validity condition. If this condition is locally violated somewhere in the system, then at least one process "rises an alarm" by outputting 0. However, recent advances in the theory of distributed checking [6, 47] demonstrate that using other decision functions δ significantly increases the power of the "checker", or "verifier". Our results show that some functions δ, in particular the classical AND operator, do not enable to use the power of quantum computing efficiently, compared to shared randomness, at least for 2-player games. In contrast, the exclusive-or operator is known to offer high potential, as far as distributed quantum computing is concerned. In particular, [16] proved that every boolean function f on n independent players can be implemented by a circuit of PR boxes that output booleans b_i, $i = 1, \ldots, n$, satisfying

$$\bigoplus_{i=1}^{n} b_i = f(x_1, \ldots, x_n) \ .$$

The results in this paper give one more evidence of the impact of the decision function δ on the ability of "deciding" boolean predicates f.

Conclusion

Conclusion

The results presented in this thesis bring new insights on the implications of locality constraints on the overall performance of network systems. Inspired by sequential complexity classes, we have participated by our work to extend the complexity theory for the $\mathcal{LOCAL}$ model initiated by Fraigniaud *et al.* [45] by placing ourselves in the context of *universal* local decision and verification, which seems to us as an ideal ground for comparing problems difficulty. This allowed us to establish full classification and separation results. In our definition of ULD and UNLD classes, we brought to light the role of the interpretation function as a key component in distributed decision. In particular, many of our positive results use interpretation that have desirable properties: they are idempotent, commutative and associative, which can be used to achieve global decision in the $\mathcal{CONGEST}$ model by a simple gossip protocol. We also discussed the role of identification and extended the famous results of Naor and Stockmeyer [72] regarding order-invariant algorithms, showing that for a large class of graphs, solving decision problems doesn't require the actual values of node Ids, but only their relative order. Finally, we point out that our framework allows a considerable economy on certificate size when it comes to verification problems, as showed by our example on Tree verification, where, by allowing only one extra-bit on the outputs, one can achieve verification with a constant size certificates, instead of $\Omega(\log n)$-bits size in other frameworks.

The other important contribution of this thesis remains in the attempt to isolate locality as a physical concept, and to study its impact regardless of other factors (such as structural, communicational or computational constraints). To do so, we have placed ourselves in a "super-model", namely the non-signalling model, admittedly non-realistic by its extremal aspects, but sufficiently large and general enough to subsume other realistic models, including classical local computing, but also models

that use quantum mechanics as a computational resource. The main advantage of the non-signalling model is its relative simplicity as it is fully characterize by linear probability constraints. One of our major contributions has thus been the design of tight lower bounds for quantum algorithms without having to manipulate concepts from quantum mechanics, as demonstrated by our case study of two-player games. We showed indeed that, for interpretation functions different from the exclusive-or operator, quantum correlations do not help in achieving better success probabilities for those games. It is important to point out that this result has been an additional motivation for us to consider universal decision and verification as done in the first part of this thesis. Indeed, the classical frameworks of distributed verification are sticked to the usage of interpretation functions based on the and-operator, hence potentially preventing us from using the potential benefit of quantum effects.

Therefore, the two sides of our work, namely the study of decision problems on one hand, and the non-signalling model on the other hand, complement each other. The definition of ULD and UNLD classes places the focus on the value of information located on fixed radius areas of a graph, which is in turn captured by the non-signalling probability distributions, where, roughly, the marginal distribution of the outputs only depends on the information located on a fixed neighborhood.

Perspectives

A natural extension of our work would be to enrich our classification results by considering several other problems and placing them in their appropriate decision and verification classes. For instance, one of our ongoing work at the time of writing involves studying the Tree decision problem. It would also be interesting to examine the possibility of extending the class hierarchy we present by considering the universal counterparts of BPLD and BPNLD [44], the randomized versions of LD and NLD. Also, as our classes rely on a universal interpretation, it would be relevant to consider refining the ULD and UNLD classes based on a possible hierarchy on the interpretation functions.

On another aspect, our work focuses on locality and abstracts all other types of issues. This leaves open perspectives regarding other aspects of distributed computing in networks. For instance, one could consider a more realistic model where commu-

nication comes at a cost. Such a work would extend the theory of the $\mathcal{CONGEST}$ model, and would relate to communication complexity theory. Besides, our framework relies on an unlimited computational power of the nodes. A natural open question would be to study a more constrained model on nodes computational power, thus linking local complexity and computational complexity theories.

As for our contributions in the non-signalling model, they make up a first step for future work about generalizations to multiplayer games, as defined in our last chapter. They also open new perspective to the study of a *non-signalling* $\mathcal{LOCAL}$ *model* where each round of computation is enhanced by the access to a multiplayer non-signalling box. How to describe such a model? How to set up relevant reduction techniques? and how to derive lower bounds for distributed quantum computing theory? These are some of the open questions that we will consider in our future work.

Bibliography

[1] Yehuda Afek, Shay Kutten, and Moti Yung. The local detection paradigm and its applications to self-stabilization. *Theoretical Computer Science*, 186:199 – 229, 1997.

[2] Kemal Akkaya, , and Mohamed Younis. A survey on routing protocols for wireless sensor networks. *AD HOC NETWORKS*, 3:325–349, 2005.

[3] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7:567–583, 1986.

[4] Dana Angluin. Local and global properties in networks of processors (extended abstract). In *STOC '80: Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 82–93. ACM, 1980.

[5] Heger Arfaoui and Pierre Fraigniaud. What can be computed without communication. In *Proceedings of the 19th International Colloquium on Structural Information and Communication Complexity*, SIROCCO'12, pages 135–146. Springer, 2012.

[6] Heger Arfaoui, Pierre Fraigniaud, and Andrzej Pelc. Local Decision and Verification with Bounded-Size Outputs. In *SSS*, pages 133–147, France, 2013.

[7] Eshrat Arjomandi, Michael J. Fischer, and Nancy A. Lynch. Efficiency of synchronous versus asynchronous distributed systems. *J. ACM*, 30:449–456, 1983.

[8] B. Awerbuch, B. Patt-Shamir, and G. Varghese. Self-stabilization by local checking and correction. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, FOCS'91, pages 268–277, 1991.

[9] Baruch Awerbuch. Complexity of network synchronization. *J. ACM*, 32:804–823, 1985.

[10] Baruch Awerbuch, M. Luby, A.V. Goldberg, and Serge A. Plotkin. Network decomposition and locality in distributed computation. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 364–369, 1989.

[11] Baruch Awerbuch, Boaz Patt-Shamir, George Varghese, and Shlomi Dolev. *Self-stabilization by local checking and global reset*, volume 857, pages 326–339. Springer Berlin Heidelberg, 1994.

[12] L. Barenboim and M. Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2013.

[13] Leonid Barenboim and Michael Elkin. Distributed $(\delta+1)$-coloring in linear (in δ) time. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 111–120. ACM, 2009.

[14] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 321–330, 2012.

[15] Jonathan Barrett, Noah Linden, Serge Massar, Stefano Pironio, Sandu Popescu, and David Roberts. Nonlocal correlations as an information-theoretic resource, 2005.

[16] Jonathan Barrett and Stefano Pironio. Popescu-rohrlich correlations as a unit of nonlocality, 2005.

[17] Florent Becker, Martin Matamala, Nicolas Nisse, Ivan Rapaport, Karol Suchan, and Ioan Todinca. Adding a referee to an interconnection network: What can(not) be computed in one round. In *Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium*, IPDPS '11, pages 508–514. IEEE Computer Society, 2011.

[18] John Stewart Bell. On the einstein-podolsky-rosen paradox. *Physics*, 1(3):195–200, 1964.

[19] Yitzhak Birk, Liran Liss, Assaf Schuster, and Ran Wolff. *A Local Algorithm for Ad Hoc Majority Voting via Charge Fusion*, volume 3274, pages 275–289. Springer Berlin Heidelberg, 2004.

[20] Gilles Brassard, Anne Broadbent, and Alain Tapp. Quantum pseudo-telepathy. *Foundations of Physics*, 35(11):1877–1907, 2005.

[21] Anne Broadbent and Alain Tapp. Can quantum mechanics help distributed computing? *SIGACT News*, 39(3):67–76, 2008.

[22] J.J. Buckley and T.E. Westen. The majority rule game. *International Journal of Game Theory*, 4(2):105–112, 1975.

[23] Harry Buhrman, Richard Cleve, Serge Massar, and Ronald de Wolf. Non-locality and communication complexity, 2009.

[24] Harry Buhrman and Hein Röhrig. *Distributed Quantum Computing*, volume 2747, pages 1–20. Springer Berlin Heidelberg, 2003.

[25] B. S. Cirel'son. Quantum generalizations of bell's inequality. *Letters in Mathematical Physics*, 4(2):93–100, 1980.

[26] J F Clauser, M A Horne, A Shimony, and R A Holt. Proposed experiment to test local hidden-variable theories. *Physical Review Letters*, 23(15):880–884, 1969.

[27] Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Inf. Control*, 70(1):32–53, 1986.

[28] Wim Dam. Implausible consequences of superstrong nonlocality. *Natural Computing*, 12(1):9–12, 2013.

[29] Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 363–372. ACM, 2011.

[30] Vasil S. Denchev and Gopal Pandurangan. Distributed quantum computing: a new frontier in distributed systems or science fiction? *SIGACT News*, 39(3):77–95, 2008.

[31] Bilel Derbel, Cyril Gavoille, David Peleg, and Laurent Viennot. On the locality of distributed sparse spanner construction. In *Proceedings of the Twenty-seventh ACM Symposium on Principles of Distributed Computing*, PODC '08, pages 273–282. ACM, 2008.

[32] Danny Dolev, Cynthia Dwork, and Larry Stockmeyer. On the minimal synchronism needed for distributed consensus. *J. ACM*, 34(1):77–97, 1987.

[33] Shlomi Dolev, Mohamed G. Gouda, and Marco Schneider. Memory requirements for silent stabilization. *Acta Informatica*, 36(6):447–462, 1999.

[34] Frédéric Dupuis, Nicolas Gisin, Avinatan Hasidim, André Allan Méthot, and Haran Pilpel. No nonlocal box is universal. *Journal of Mathematical Physics*, 48(8), 2007.

[35] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988.

[36] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete?". *Phys. Rev.; Physical Review*, 47(10):777–780, 1935.

[37] Michael Elkin. A near-optimal distributed fully dynamic algorithm for maintaining sparse spanners. In *Proceedings of the Twenty-sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '07, pages 185–194. ACM, 2007.

[38] Michael Elkin, Hartmut Klauck, Danupon Nanongkai, and Gopal Pandurangan. Can quantum communication speed up distributed computation?, 2012.

[39] Pierre Fraigniaud, Cyril Gavoille, David Ilcinkas, and Andrzej Pelc. Distributed computing with advice: information sensitivity of graph coloring. *Distributed Computing*, 2009.

[40] Pierre Fraigniaud, Mika Göös, Amos Korman, and Jukka Suomela. What can be decided locally without identifiers? In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, PODC '13, pages 157–165. ACM, 2013.

[41] Pierre Fraigniaud, MagnúsM. Halldórsson, and Amos Korman. On the impact of identifiers on local decision. In *Principles of Distributed Systems*, volume 7702 of *OPODIS '12*, pages 224–238. Springer Berlin Heidelberg, 2012.

[42] Pierre Fraigniaud, David Ilcinkas, and Andrzej Pelc. Communication algorithms with advice. *J. Comput. Syst. Sci.*, 76(3-4):222–232, 2010.

[43] Pierre Fraigniaud, Amos Korman, and Emmanuelle Lebhar. Local mst computation with short advice. In *Proceedings of the Nineteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '07, pages 154–160. ACM, 2007.

[44] Pierre Fraigniaud, Amos Korman, Merav Parter, and David Peleg. Randomized distributed decision. *Distributed Computing*, 7611:371–385, 2012.

[45] Pierre Fraigniaud, Amos Korman, and David Peleg. Local distributed decision. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 708–717. IEEE Computer Society, 2011.

[46] Pierre Fraigniaud and Andrzej Pelc. Decidability classes for mobile agents computing. In *Proceedings of the 10th Latin American International Conference on Theoretical Informatics*, LATIN'12, pages 362–374. Springer-Verlag, 2012.

[47] Pierre Fraigniaud, Sergio Rajsbaum, and Corentin Travers. Locality and checkability in wait-free computing. *Distributed Computing*, pages 223–242, 2013.

[48] Stuart Freedman and John F. Clauser. Experimental test of local hidden-variable theories. *Physical Review Letters*, 28(14):938–941, 1972.

[49] Felix C. Gärtner. Fundamentals of fault-tolerant distributed computing in asynchronous environments. *ACM Comput. Surv.*, 31(1):1–26, 1999.

[50] Cyril Gavoille, Ralf Klasing, Adrian Kosowski, Kuszner, and Alfredo Navarra. On the complexity of distributed graph coloring with local minimality constraints. *Netw.*, 54(1):12–19, 2009.

[51] Cyril Gavoille, Adrian Kosowski, and Marcin Markiewicz. What can be observed locally? round-based models for quantum distributed computing. In *Proceedings of the 23rd international conference on Distributed computing*, DISC '09, pages 243–257. Springer-Verlag, 2009.

[52] G. C. Ghirardi, A. Rimini, and T. Weber. A general argument against superluminal transmission through the quantum mechanical measurement process, 1980.

[53] Oded Goldreich, editor. *Property Testing: Current Research and Surveys*. Springer-Verlag, Berlin, Heidelberg, 2010.

[54] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 406–415. ACM, 1997.

[55] Mika Göös and Jukka Suomela. Locally checkable proofs. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, PODC '11, pages 159–168. ACM, 2011.

[56] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8 - Volume 8*, HICSS '00. IEEE Computer Society, 2000.

[57] Ralph E. Johnson and Fred B. Schneider. Symmetry and similarity in distributed systems. In *Proceedings of the Fourth Annual ACM Symposium on Principles of Distributed Computing*, PODC '85, pages 13–22. ACM, 1985.

[58] Shmuel Katz and Kenneth Perry. Self-stabilizing extensions for message-passing systems. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, PODC '90, pages 91–101. ACM, 1990.

[59] Liah Kor, Amos Korman, and David Peleg. Tight bounds for distributed mst verification. In Thomas Schwentick and Christoph Dürr, editors, *STACS*, volume 9, pages 69–80, 2011.

[60] Amos Korman and Shay Kutten. Distributed verification of minimum spanning trees. *Distributed Computing*, 20:253–266, 2007.

[61] Amos Korman, Shay Kutten, and Toshimitsu Masuzawa. Fast and compact self stabilizing verification, computation, and fault detection of an mst. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '11, pages 311–320. ACM, 2011.

[62] Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Computing*, 22(4):215–233, 2010.

[63] Amos Korman, Jean-Sébastien Sereni, and Laurent Viennot. Toward more local-

ized local algorithms: Removing assumptions concerning global knowledge. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '11, pages 49–58. ACM, 2011.

[64] Fabian Kuhn. Weak graph colorings: Distributed algorithms and applications. In *Proceedings of the Twenty-first Annual Symposium on Parallelism in Algorithms and Architectures*, SPAA '09, pages 138–144. ACM, 2009.

[65] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *CoRR*, 2010.

[66] Fabian Kuhn, Thomas Moscibroda, and Rogert Wattenhofer. What cannot be computed locally! In *Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing*, PODC '04, pages 300–309. ACM, 2004.

[67] Stephanie Lindsey, Cauligi Raghavendra, and Krishna M. Sivalingam. Data gathering algorithms in sensor networks using energy metrics. *IEEE Trans. Parallel Distrib. Syst.*, 13(9):924–935, 2002.

[68] Nathan Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.

[69] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986.

[70] Kenneth O. May. A set of independent necessary and sufficient conditions for simple majority decision. *Econometrica*, 20(4):680–684, 1952.

[71] Nicholas R. Miller. Graph-Theoretical Approaches to the Theory of Voting. *American Journal of Political Science*, 21:769–803, 1977.

[72] Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM J. Comput.*, 24(6):1259–1277, 1995.

[73] Alessandro Panconesi and Aravind" Srinivasan. The local nature of -coloring and its algorithmic applications. *Combinatorica*, 15(2):255–280, 1995.

[74] Alessandro Panconesi and Aravind Srinivasan. On the complexity of distributed network decomposition. *Journal of Algorithms*, 20(2):356 – 374, 1996.

[75] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Mono-

graphs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2000.

[76] Sandu Popescu and Daniel Rohrlich. Quantum nonlocality as an axiom. *Foundations of Physics*, 24(3):379–385, 1994.

[77] Johannes Schneider and Roger Wattenhofer. A new technique for distributed symmetry breaking. In *Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '10, pages 257–266. ACM, 2010.

[78] Jukka Suomela. Survey of local algorithms. *ACM Comput. Surv.*, 45(2):24:1–24:40, 2013.

[79] Hüseyin Özgür Tan and Ibrahim Körpeoğlu. Power efficient data gathering and aggregation in wireless sensor networks. *SIGMOD Rec.*, 32(4):66–71, 2003.

[80] Jennifer Lundelius Welch. Simulating synchronous processors. *Information and Computation*, 74(2):159 – 171, 1987.

[81] Naixue Xiong, Yan Yang, Ming Cao, Jing He, and Lei Shu. A survey on fault-tolerance in distributed network systems. In *Computational Science and Engineering, 2009. CSE '09. International Conference on*, volume 2, pages 1065–1070, 2009.